LONG RUNNING TRANSACTIONS WITHIN ENTERPRISE RESOURCE PLANNING SYSTEMS

PhD Thesis

Abdullah Bajahzar

This thesis is submitted in partial fulfilment of the requirements for the

Degree of Doctor of Philosophy

Software Technology Research Laboratory (STRL)

Faculty of Technology

De Montfort University

March 2014

DEDICATION

To

my mother,

who deserves special recognition

for her endless support throughout my life.

To

my wives,

for their encouragement and great support

during this time of challenge.

Without them,

nothing would have ever happened.

To

my lovely children,

for their love and encouragement.

To

my sisters,
for their encouragement and their endless love.

DECLARATION

I declare that the work presented in this thesis is original work undertaken by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degrees or qualifications. It is submitted for the degree of Doctor of Philosophy, at the Software Technology Research Laboratory, Faculty of Technology, at De Montfort University.

ACKNOWLEDGMENTS

In the name of Allah, the Most Merciful and the Most Gracious, first and foremost, my deepest thankfulness goes to Almighty ALLAH for all his bounties and blessings, and for giving me the ability to complete this thesis. This research report could not have been completed without the recommendations, suggestions and advice of many people. Their contributions, guidance and support are highly appreciated.

My thanks and appreciation go to my great supervisor, Professor Hussein Zedan, and Professor Duska Rosenberg who have encouraged my autonomy and offered guidance, support, valuable comments, constructive criticism, academic support, guidance and suggestions throughout my studies and given me invaluable instruction, without which this thesis could not have been produced in its present form. I am deeply indebted to him for his insights and suggestions on the research topic and for his invaluable supervision at various stages of my work. He has helped me in many ways to develop my academic thinking and problem solving which will be very helpful in my future work and research. I feel extremely lucky to have had him as my supervisor, as he remains always dedicated to the success of his students. My thanks also go to my second supervisor, Dr. Kath Garnett.

I would like to express my thanks to my family, especially my mother and my wives, my children and my sisters, who have always supported me during my studies and encouraged me throughout my research. They truly share in this achievement.

I would also like to thank the staff and officials at the School of Computer Sciences and Engineering at De Montfort University, and especially offer thanks to all the members of the Software Technology Research Laboratory (STRL), which without whose knowledge and assistance this study would not have been successful.

I would like to thank all of my colleagues, especially Dr. A. Alqahtani, Dr. A. Al-Ajlan, Dr. A. Al-bayati, Dr. N. Al-alwan, Dr. M. Al-Sammaraei, Dr. O. Al-Shathri, Dr. O. Al-Hassan, Dr. K. Aldrawiesh, Dr. F. Al-Shathri, H. Aldabbas, Dr. Y. Al-Saway, Dr.A. Alhussain, Faisal Al-Ghamdi, Dr. A. Al-Gamdi, Dr. A. Al-Marghilani, Dr. Bassam Zafar, Dr. A. Al-Zahrani, Dr. M. Al-Awairdhi, Dr. S. Al-Otaibi and Dr. M. Taye, at STRL, for their valuable suggestions and discussions during the study, which made working with them very enjoyable

I would also like to express my great thanks to Dr. A. Al-Ghamdi, Dr. H. Al-Askar, Dr. Z. Al-Zamil, Dr. R. Madani and Mr. A. Al-Otaibi for their assistance and support throughout the whole duration of my research.

ABSTRACT

Recently, one of the major problems in various countries is the management of complicated organisations to cope with the increasingly competitive marketplace. This problem can be solved using Enterprise Resource Planning (ERP) systems which can offer an integrated view of the whole business process within an organisation in real-time. However, those systems have complicated workflow, are costly to be analysed to manage the whole business process in those systems. Thus, Long Running Transaction (LRTs) models have been proposed as optimal solutions, which can be used to simplify the analysis of ERP systems workflow to manage the whole organiational process and ensure that completed transactions in a business process are not processed in any other process.

Practically, LRTs models have various problems, such as the rollback and check-pointing activities. This led to the use of Communication Closed Layers (CCLs) for decomposing processes into layers to be analysed easily using sequential programs. Therefore, the purpose of this work is to develop an advanced approach to implement and analyse the workflow of an organisation in order to deal with failures in Long Running Transaction (LRTs) within Enterprise Resource Planning (ERP) systems using Communication Closed Layers (CCLs). Furthermore, it aims to examine the possible enhancements for the available methodology for ERP systems based on studying the LRT suitability and applicability to model the ERP workflows and offer simple and elegant constructs for implementing those complex and expensive ERP workflow systems.

The implemented model in this thesis offers a solution for two main challenges; incompatibilities that result from the application of transitional transaction processing concepts to the ERP context and the complexity of ERP workflow. The first challenge is addressed based on offering new semantics to allow modelling of concepts, such as

rollbacks and check-points through various constraints, while the second is addressed through the use of the Communication Closed Layer (CCL) approach.

The implemented computational reconfigurable model of an ERP workflow system in this work is able to simulate real ERP workflow systems and allows obtaining more understanding of the use of ERP system in enterprise environments. Moreover, a case study is introduced to evaluate the application of the implemented model using three scenarios. The conducted evaluation stage explores the effectiveness of executable ERP computational models and offers a simple methodology that can be used to build those systems using novel approaches.

Based on comparing the current model with two previous models, it can be concluded that the new model outperforms previous models based on benefiting from their features and solving their limitations which make them inappropriate to be used in the context of ERP workflow models.

PUBLICATIONS

1. BAJAHZAR, A., ALQAHTANI, A. & BASLEM, A., A SURVEY STUDY OF THE ERP SYSTEM, *ADVANCED COMPUTER SCIENCE APPLICATIONS AND TECHNOLOGIES (ACSAT)*, PP. 156-160. IEEE, 2012.

2. BAJAHZAR, A., ALQAHTANI, A. & BASLEM, A., SUCCESSFUL IMPLEMENTATION OF ERP, *ADVANCED COMPUTER SCIENCE APPLICATIONS AND TECHNOLOGIES (ACSAT)*, PP. 246-252. IEEE, 2012.

3. BAJAHZAR, A., BASELM, A., ALHAJ, H., 'A NEW DATA MANAGEMENT APPROACH FOR LONG RUNNING TRANSACTION: RECONSIDERING PP/T', *THE 7TH SAUDI INTERNATIONAL CONFERENCE*, IN PROCEEDINGS OF ICT WORKSHOP IN EDINBURGH UNIVERSITY, EDINBURGH UK, 2014.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ACID	Atomicity, Consistency, Isolation and Durability
AME	Automatic Mutual Exclusion
BER	Bit Error Rate
BERT	Bit Error Rate Test
BPR	Business Process Reengineering
CA	co-ordination agent
CAD	Computer-Aided Design
CAM	Client Association Managing
CASE	Computer-Aided Software Engineering
CCLs	Communication closed Layers
CMOT	Communal Manufacture of Technology
DBMS	database management system
EOU	Ease Of Use
ERP	Enterprise Resource Planning
FIFO	First In First Out
GUI	Guide User Interface
HR	Human Resource
ISs	Information Systems
IT	Information Technology
KM	Knowledge Management
LRTs	Long Running Transactions
MRP	Material Requirements Planning
OTP	Online Transaction Processing
Pout	Probability of outage
SCM	Supply Chain Management
SCOT	Social Construction of Technology

SIS	Strategic Information Systems
SNR	Signal to Noise Ratio
TAM	Technology Acceptance Model
TRS	Technology Receipt Scheme
TTF	Task-Technology Fit

CHAPTER 1

INTRODUCTION

This chapter offers an overview of Enterprise Resource Planning (ERP) systems, explain the motivation for ERP systems, explore the scope of the thesis and research questions, outline the research methodology, determine the main components of the measure of success and outline the thesis structure.

1.1 INTRODUCTION

Generally, Information Technology (IT) is introduced for its ability to endorse managerial performance by improving both innovation and operational competence. To date enterprise resource planning (ERP) systems have represented the most essential improvements in IT commercial utilisation. These systems allow companies to obtain enhanced performance through normalising data, as well as allowing them to incorporate several company functions such as marketing, Human Resource (HR), sales, manufacturing and finance. Thus, ERP systems are understood to be valuable as the most all-encompassing systems for dynamically delineating company conditions and characteristics [1, 2].

In the 1960s, the focus was on designing software packages for the purpose of holding records for management without taking into consideration additional demands. By the 1970s, companies had shifted their focus towards customer demand. Thus, at this time Material Requirements Planning (MRP) software packages were brought to market. Market conditions vary daily and are continually shifting, but this knowledge did not prompt the development of advanced technologies. Over the next decade, it was found that knowledge of production did not impact on income. Thus, several additional issues were introduced into resources planning systems; including considerations of economics, planning capability and requirements. Thus, both economic accounting and economic management systems were incorporated in conjunction with development and materials management systems [3, 4].

In the 1990s, competition among the companies in the marketplace was at its height. During this decade, companies were seeking out innovative methods in order to obtain an aggressive benefit for remaining active in the market, since MRP software packages did not satisfy their main demand. Thus, companies began utilising ERP systems, since they were known to effectively provide assistance to direct decision making and

institutional management, by maintaining information and offering both information precision and consistency [5, 6].

ERP systems were credited with the development of both solid performance and market worthiness, through competence and usefulness gains. In recent years, similar systems have been developed using advanced systems, such as Client Association Managing (CAM) system and Supply Chain Management (SCM) systems. Figure 1.1 below shows ERP systems development [5].

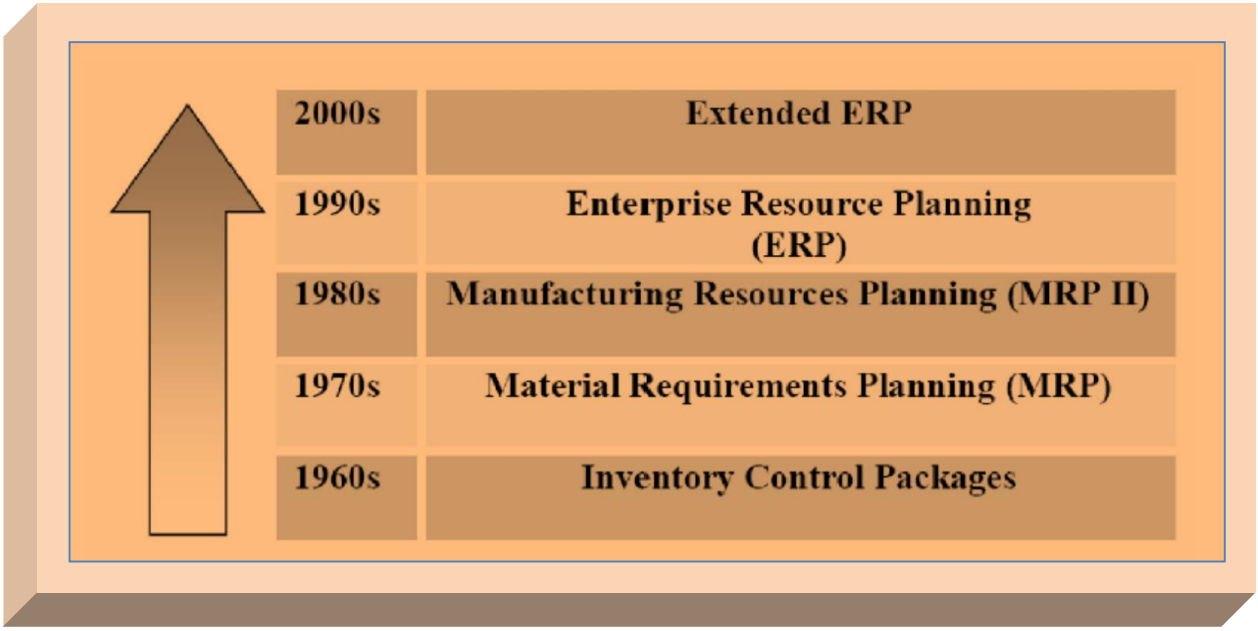

Figure 1. 1 Development of ERP systems [5]

To date, the possibilities associated with Information Systems (ISs) have been studied by several researchers focusing on robustness among IS and precise managerial dimensions. These research studies have not attempted to adapt the ERP system. ERP systems are then not be utilised competently and returns on investment can be negative as they are not tailored to companies. Thus, recently, advanced strand of ERP research have arisen that target the development and incorporation of ERP systems in companies [7, 8].

A new strand of ERP system research has been introduced in order to discover if ERP systems are operating effectively in each of a system's parts in a varying environment; in addition, if these systems are sustaining improvements to company performance and being used or otherwise. However, many current surveys are restricted as it is not possible to obtain complete data sets. Thus, there is a need for profound examination of ERPs in use in order to obtain acceptable results [9, 10].

Most United States companies are in the ERP systems designing stage [3]. Internationally, predictable ERP software proceeds were huge in 2006 [4]. The volume of practitioners, expenditure and academic researchers has broadly concentrated on the mechanism that is used to execute ERP systems productively [11, 12].

ERP system design worldwide now focuses on variations in sectors showing its importance in the managerial innovation field. Sectors such as shipbuilding, building, engineering and entertainment have several restrictions that are related to their main company processes. These sectors are distinctive and a lot of stakeholders are needed in order to facilitate them to work together to achieve their main aims and objectives. This thesis, therefore, seeks to collect data for a distinctive and complicated data set. The abundance and time sensitivity of collapsed data can impede control of the knowledge it contains; leading to the supposition that bequest systems should be substituted in order to deal with the extremely disjointed, low consistency and organically isolated company processes [13, 14].

In the midst of the rapid improvement in the data tools sector, ERP systems adoption has increased rapidly worldwide, especially in Chinese companies; however, in most of these companies ERP systems are still in the initial and planning stages. These companies' managers currently require training to acquire enough knowledge about what they will have to do after the ERP implementation and what will be happened [15].

When other countries have used ERP system, they have seen many benefits. Two industrial companies interviewed in the northeast of China discovered the common ERP process after implementing them in Chinese companies. Since, there are many differences in ERP performance, multiple stages were identified, such as the fervour of innovation stage; Shake downward stage, fast civilising stage and finally stability and continuously civilising stage [15].

A critical management point in ERP outsourcing management refers to which ERP outsourcing association the outsourcing agreement is attributable to, because when the contract for outsourcing was offensively or moderately written, an imperative unconstructive insinuation could affect it. Additionally, to clearly implement managers' interests, contracts have to encourage the performance of vendors while also discouraging underperformance. Although outsourcing has arisen in many research studies, the incentivising of ERP outsourcing contracts has not been adequately addressed, possibly because incentivising ERP outsourcing contracts issue is in its infancy [16].

ERP systems in the form of social embedding were examined in the presence of practices of the Knowledge Management (KM) to determine if using such systems hinders or facilitates Knowledge Management (KM) practices. Knowledge Management (KM) practices require the researcher to socially embed computer mediated communication. In general to facilitate the performance of organisations, enterprise-wide information is shared though the performance ERP systems. This also implies that the Communal Manufacture of Technology (CMOT) advance's applications which aim to solve possible ERP systems problems. It was found that CMOT advance depends on three main factors; these are interpretation flexibility, relevant social group, closure and stabilisation [17].

After explaining the Communal Manufacture of Technology (CMOT) advance and using [18] knowledge, diversity amid non mixed and mixed additions to ERP systems is

conceptualised and the need for social embedded ERP systems implementation to enhance the transfer of the knowledge addressed. Many questions should be answered, the most important questions are: (1) how can problems be solved by applying the Social Construction of Technology (SCOT) approach? (2) What is the role of the ERP?' what facilitates the transfer of the knowledge in organisations? (3) What are the associated problems that occur when ERP is implemented to support knowledge management processes in organisations [17, 18].

Software applications of ERP systems are known as track information; these are very important and frequently used by government agencies and businesses to track their financial information, items and employees, and to assist them in management in general. After financial information, items and employees are managed, an organisation can acquire a reliable overview of the whole "enterprise" through the application of integrated applications. Then from a wider perspective of ERP, ERP systems can be left in the midst of conventional deals that depend on accounting systems to sustain association level and provide personal capital management and fully integrated finances, which depend on manufacturing results and observations [19].

As a result of the difficulties of global competition, businesses are incorporating data systems to connect contractors, clients and additional partners in production, to whom they may interrelate. With the aim of improvement, more people are travelling for their work. This means that when they are away from home, they prefer familiar systems are accessible. Traditional ERP was a C/S scheme, which could not meet these requirements. Consequently, new needs provided the imperative to improve the ERP system [19].

1.2 MOTIVATION

This thesis introduces the development and analysis of an ERP system and validates its workflow at various companies. This system was developed with the intention of resolving the problem of using both rollback and check-pointing activities, due to the inability of some actions involved in LRTs being reversed. This problem has been solved based on developing and using CCLs to handle events that cannot be controlled using automatic techniques. These layers permit the serialisation of a well distributed workflow in order to generate a simple and easy equivalent quasi-sequential workflow. This facilitates the design of a fault tolerant workflow as layers and the use of several available backward and forward error recovery methods.

A proposed motivation is performed based on developing and applying an approach known as LRT in the ERP system. This class of transactions can affect a variety of business processes performed minimally by one Information Technology (IT) system. It proceeds based on the premise of managing each business process separately, thus, the completed transaction for a business process is not processed elsewhere. This is considered as the main spatial feature of LRTs.

1.3 PROBLEM STATEMENT

Atomic transactions are used in variety database systems to specify designed rational work units for activities that are short–lived, lasting less than a few seconds. Transactions are either completely performed or not at all. This means that when something goes wrong during the transaction process, a rollback or check-pointing activity is used by database programmers to allow the system to rollback to a predefined working state, known as a checkpoint, to reboot the system to its initial state. However, some actions that are required for long running transactions (LRTs) cannot be reversed.

Thus, both rollback and check-pointing becomes unusable. To solve this problem, CCLs are used in order to handle events that cannot be controlled by automatic techniques. Furthermore, CCLs can be used to deal with failures in Long Running Transaction (LRTs) within Enterprise Resource Planning (ERP) systems.

1.4 THESIS SCOPE AND RESEARCH QUESTION

The following two sub-sections explore the thesis' scope and suggested research questions

1.4.1 THESIS SCOPE

This thesis aims to develop an advanced approach to implement and analyse the workflow of an organisation in order to deal with failures in Long Running Transaction (LRTs) within Enterprise Resource Planning (ERP) system using Communication Closed Layers (CCLs). This involves exploring the problems with both rollback and check-pointing activities due to the inability of some actions involved in LRTs to be reversed, determining the main failures in ERP systems, understanding the role of workflow in an organisation, introducing a computational model for a specific organisational workflow system, exploring the main concept of CCLs, supporting these layers based on a given development process, analysing the use of CCL in a certain case study, realising system traps and exploring their effects on both information management and cost reduction.

1.4.2 RESEARCH QUESTIONS

Main research question: How can CCLs deal with failure in long running transactions within an ERP system?

To answer this overarching question the following sub-questions were developed as research questions:

RQ1: How can the problems associated with both rollback and check-pointing activities be solved?

RQ2: Does the existence of both failures and malfunctions in ERP systems have an effect on the system's security and effectiveness?

RQ3: How the organisation workflow system can be enhanced?

RQ4: What are the main reasons for using LRTs in analyzing the workflow of an ERP system?

RQ5: Why CCLs are applied for LRTs?

RQ6: What are the problem of the old transactions in distributed systems model and how it can be solved?

1.5 RESEARCH METHODOLOGY

This thesis introduces an advanced approach to design and analyse the workflow of anorganisation to deal with failures in LRTs within ERP systems using CCLs.

In this thesis, an advanced approach is developed to facilitate the analysis of ERP workflow systems based on applying Long-Running Transactional (LRT) modelling concepts for those systems. Initially, the main challenges of ERP systems that need to be solved are determined. After that, those challenges are addressed based on building a LRT model using Communication Closed Layer (CCL) approach. Next, this computational model is tested and evaluated to study its effectiveness to be applied for simulating real ERP workflow systems. This is performed based on applying the model on a case study using three scenarios though the use of the reconfigure-ability features

of the framework to simulate behaviors, such as failures and delays at various sub-business processes. The data from these simulations are then used for analysis.

The typical stages of the research methodology are illustrated below:

- Understand the main concepts of ERP systems and LRTs;
- Determine the main failures in existing ERP systems;
- Review several of the preceding tasks associated with ERP systems' structure in the literature;
- Understand the workflow process in an organisation;
- Explore the problem of using both rollback and check-pointing activities in LRTs;
- Introduce a computational model for a specific organisation workflow systems'
- Explore and understand the main concepts of CCLs;
- Analyse the use of CCLs in a certain case study;
- Realise the system traps and explore their effects on both information management and cost reduction; and
- Evaluate the ability of the approach designed to deal with failures in LRTs within ERP system using CCLs.
- Compare the current model with previous ones

1.6 MEASURE OF SUCCESS

The success of the research and the developed system will be measured against the following success criteria:

- To reduce the overall failure rate associated currently available systems using CCL.

- The developed system of this study should offer more enhanced results than previously available systems and resolve the associated problems of such systems.
- To answer and resolve the research questions through the research and development of the system.

1.7 THESIS STRUCTURE

This thesis is divided into several chapters as follows:

- **Chapter One: Introduction**

This chapter introduces the thesis and contains an overview of ERP systems, outlines the research motivation, scope and methodology, explores the suggested research questions, illustrates the main components of the measures of success and details the structure of the thesis.

- **Chapter Two: Preliminaries**

This chapter offers a brief background concerning the principles, concepts and definitions of Enterprise Resource Planning (ERP), long Running Transactions (LRT) and Communication Closed Layers (CCLs).

- **Chapter Three: Literature Review Concerning Enterprise Resource Planning (ERP)**

This chapter offers a review of some of the available works that related to Enterprise Resource Planning (ERP) systems with exploring their main advantages and limitations which need to be addressed.

- **Chapter Four: Literature Review Concerning Long Running Transactions (LRT) and Communication Closed Layers (CCLs)**

This chapter offers a review of some of the available works that related to both Long Running Transactions (LRTs) and Communication Closed Layers (CCLs). In addition, it explores the application of Layers in LRTs as well as the application of LRTs in ERP Systems.

- **Chapter Five: Modifications For Communication Closed Layers**

This chapter introduces modifications for CCLs and describes some of the processes used for choices and iterations in the local environment; analyses the main model of CCLs using the programming language Occam.

- **Chapter Six: Methodology and design approach**

This chapter explores the main two design challenges that are solved in this thesis and describes the conducted methodology to develop the communication closed layers in ERP context.

- **Chapter Seven: Implementation and case study**

This chapter describes computational model of an Enterprise Resource Planning (ERP) workflow, offers a case study and describes the core model implementation in details.

- **Chapter Eight: Evaluation and Results**

This chapter introduces the evaluation of the implemented computational model performance using three scenarios and compares the current model with two previous ones.

- **Chapter Nine: Conclusion and future works**

This chapter summarizes the presented research, evaluates the developed system, provides contributions to knowledge and illustrates some works that can be done in the future to improve the system.

1.8 SUMMARY

This chapter has introduced ERP, their background, uses and development history. As proposed, these systems can be compacted to obtain enhanced performance by normalising data, as well as incorporating several company functions such as marketing, Human Resource (HR), sales, manufacturing and economics. Thus, ERP systems have the most enveloping characteristics.

Atomic transactions are used in various database systems to specify a rationale for activities that are short–lived, regularly long-lasting and under few seconds. When something goes wrong during the transaction process, a rollback or check-pointing activity is used by database programmers to allow the system to rollback to a predefined working state that is known as a checkpoint and will reboot the system to its initial state. On the other hand, some actions in the LRTs cannot be reversed. Thus, both rollback and check-pointing may be unusable. To solve this problem, this research proposes the

development of an advanced approach to implement and analyse the workflow of an organisation in order to deal with failures in LRTs within an ERP system using CCLs.

CHAPTER 2

PRELIMINARIES

This chapter offers a background and brief description concerning the main principles, concepts and definitions of Enterprise Resource Planning (ERP), long Running Transactions (LRT) and Communication Closed Layers (CCLs).

2.1 INTRODUCTION

Generally, Enterprise Resource Planning (ERP) is a group of software packages that ties several functions into a consistent dataset. ERP systems replace many bequest applications with only one incorporate set, guarantee optimal business practices, offer enhanced access for company data and standardizes the technology infrastructure. Those systems can be supported using both long running transactions and communication closed layers. Mainly, transactions are used to offer all the required information to keep a certain business running efficiently and correctly. In addition, it can offer appropriate reports and documents, data for various systems and safeguard data. Transactions are one of the main business processes that must be computerized to carry out routine operations. On the other hand, communication closed layers are used to schedule actual tasks a limited environment to sustain an imprecise computation technique. This chapter introduces the main concepts and definitions of ERP systems, transactions, long running transactions and communication closed layers.

2.2 TRANSACTIONS OVERVIEW

Transactions are mainly used in database systems, to indicate a logical work unit that is implemented for activities that are short–lived, frequently lasting for less than a few seconds. The transactions are fully executed or not at all. In other words, when something goes wrong during the transaction execution, then a rollback activity is executed in which the system state is re-established precisely as it was at the transaction start. The execution of the rollback activity by the system depends on acquiring locks on the essential resources at the transaction start. These locks are released at its end only in one of two cases; full achievement and rollback. The utilisation of locks that prevent access to the resources is rationalised by the short duration of the transaction [20].

Transactions should satisfy the Atomicity, Consistency, Isolation and Durability (ACID) properties [20]. The current improvement in distributed systems has produced the need for a fresh transaction notion where distant entities that are mainly in different regions may interrelate by executing complicated activities that need human interaction which may take minutes, days or, in some cases, weeks. This enlarges the time period regarding the ACID transactions, prevents the use of locks on resources and makes rollback activities impossible. In these transactions, the alternative to rollback activities is the use of compensations that are activities explicitly programmed in order to eradicate the effect of the executed actions and may need some penalty payment. These transactions are frequently called LRTs and also sagas [21], extended transactions [22] and web transactions [23].

The main property of databases is atomicity which means that transactions happen entirely or are rolled back to their previous state. In other words, this property follows the all or nothing principle, so that the transaction prevents a partial database update. The atomicity property is considered a very useful principle in e-commerce [24]. Atomic transactions include many operations on the dataset which either must all happen in their entirety or none of them must happen at all. Since an atomic transaction prevents partial database updates, then many problems can arise. The origin of the word atomic is from the Greek Classical concept of a basic and inseparable component known as an atom.

One atomicity that is related to e-commerce is booking an airline voucher, which involves two main actions: a seat condition and payment. In this case, the traveller has to reserve a seat and pay for it but does not have to pay if a seat is not reserved. Thus, the airline booking system must not take payment from the traveller without reserving a seat for them and must not reserve a seat without payment.Another current example of atomicity is electronic payment, which is used widely, and is, along with the privacy of consumers, a very important issue. In electronic payment, digital cash is used, whereby buyers pay using coupons in order to purchase various goods. These coupons are

unidentified until they are used more than one time. When a buyer purchases goods and pays by using coupons but before recognition, then the network will fail. In this case, the buyer can send the used coupon again but this will result in making the buyer identity known if the seller has previously received this coupon, but if the buyer does not use this coupon and the seller did not receive the coupon, then this will result in locking the funds. [24]

One solution to the proposed problem is the utilisation of the ACID properties [25]. These four properties are dependent; consistency depends on both the durability and rollback, while isolation depends on the atomicity in order to roll back all the modifications when a failure occurs.

2.3 MAIN CONCEPTS OF LONG RUNNING TRANSACTIONS

Transactions represent the isolation of a certain action such that it emerges to process out of the transaction where an atomic action has taken a certain place. Mainly, the basic motivation for transactions is that maintaining the consistency rules is not possible on a per-operation foundation. Therefore, some operations must be gathered in one transaction and illustrated for other processes as one action that can succeed or fail. These concepts with the durability one represent the ACID principles; atomicity principle guarantees that the transaction completely succeed or does not leave an effect on the state of the system, consistency principle guarantees that the state of the system progresses from a correct state to another one, isolation principle guarantees that the transaction intermediate results cannot be demonstrated for other transactions giving the notion that transactions are isolated and the durability principle guarantees that the transaction result is persevered and cannot be undone [26].

For several years, the ACID principles have proven to be a main issue for handling various dataset operations where they depend on the resource-locking policy in which transactions work under complete isolation illusion from the world rest. The recovery from errors through a transaction needs simply a save point for the whole processes that

participate in that transaction before starting a new transaction. When a failure occurs, the involved processes are reverted for the save point. When the transaction commits, the save points are then discarded that is reversing the transaction impacts is not possible outside the commit operation where this is considered in line of the durability principle [26].

2.4 Main Concepts of Communication Closed Layers (CCLs)

A CCL is an advanced method for serialising a wide distributed workflow in order to offer an equivalent quasi sequential flow that can be easily understood and analysed. It allows the designer to arrange the workflow in layers that represent various atomic activities based on using several forward and backward error retrieval approaches. In any workflow, a layer is considered as a communicative one when it includes minimally one communication primitive. On the other hand, it is considered as communication closed when it begins and ends in one layer. A non communication layer is the one that does not include any communication primitives. Figure 2.1 demonstrates the difference among communication and non communication layers:

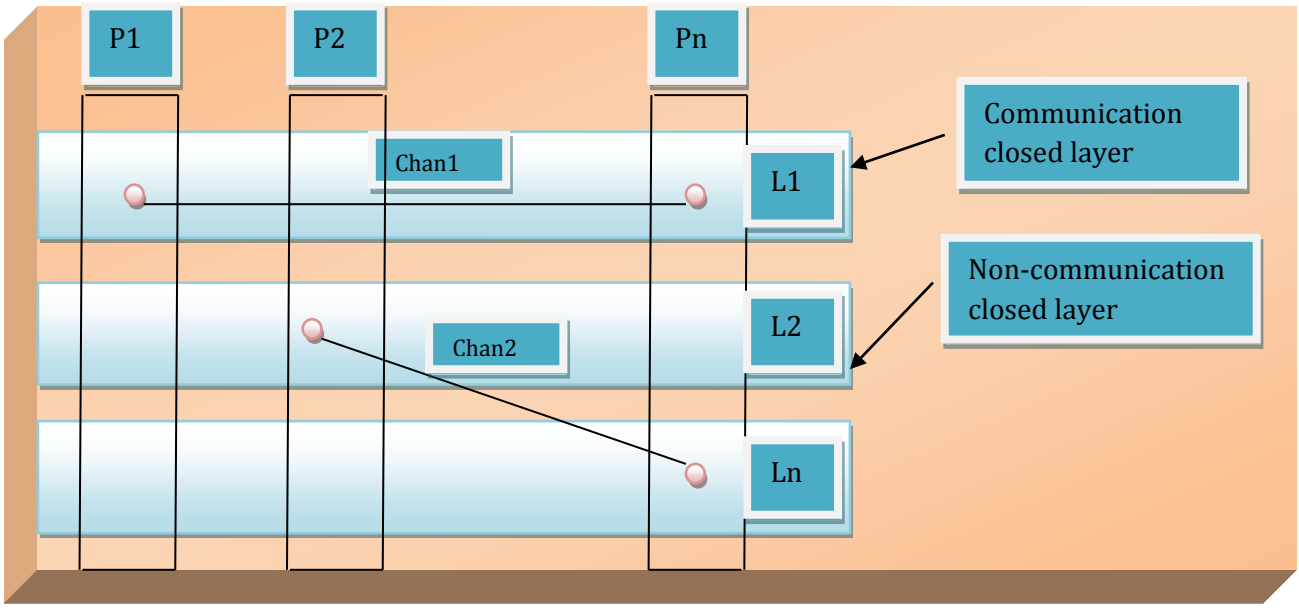

Figure 2. 1 Communication and non communication close layers

The first use of CCL was for analysing distributed systems through message passing. After this, they were used for scheduling actual tasks in restricted environments.

2.5 Main Concepts of Enterprise Resource Planning (ERP) Systems

ERP schemes are considered to be central to both a company's venture backbone and its successful data management. Businesses must predict, react and respond to the development demands of the marketplace. If not, these businesses will fail. Business strategy verifies achievement and ensures the continued success of a business. Recently, business strategy has focused on the aggressive and competent usage of IT. ERP systems are packaged software systems, which allow companies to control resources competently and utilise them efficiently by offering complete integrated solutions for the data processing requirements in companies. Examples of these resources are finances, materials and human resources [27].

ERP systems have the ability to sustain both the business process familiarised view and the business processes which are consistent in the venture. The main characteristics of ERP systems are: automating and integrating the business processes of a company, providing standard information and measures across the whole venture, and generating and accessing data in a concurrent environment [27]. ERP systems have the ability to develop processes as well as to reduce costs. In addition, the main two frontiers of ERP systems are "supply chain management and e-business". Connecting the applications of the supply chain management with several business systems facilitates both slashing cycle times and decreasing the numbers of records kept by the system users. Furthermore, users can connect with their customers, suppliers and distributors easily in order to involve them in the e-business. [28]

The support of top management is one the most significant factors that are required during ERP implementation: a new system should have the approval of the senior

managers and be aligned with the main strategic aims of the business. Project success can be ensured by linking it to managers' bonuses. The top management must broadly endorse the project and considering it a top priority [29, 30].

ERP teamwork and composition is another factor required during ERP implementation. The ERP system team must be composed of the best individuals available, including external advisors and internal staff, who have the ability to improve the skills required for technology implementation. Team members must be allocated enough time to implement the system and must be co-located in order to simplify their work. They must also be provided with reimbursement and inducements to implement the ERP system punctually and productively within the given budget. Finally, the team must be familiar with the various products and functions of the business in order to recognize the requirements of the system if it is to support the main business processes [31, 32].

Other critical success factors are an appropriate business plan and clear vision, which are required during implementation in order to keep the project on track. The business plan is significant in determining the anticipated, planned and actual advantages, costs, resources, timeline and risks. In addition, there must be an obvious model of the business that determines the way that the company must operate. The work of the project must be associated with the business requirements and must be clearly confirmed. Thus, the business plan facilitates the work and affects its likelihood of success [30, 31, 32].

2.6 SUMMARY

As a conclusion, the main concepts and definitions of Enterprise Resource Planning (ERP), long Running Transactions (LRT) and Communication Closed Layers (CCLs) are introduced in this chapter. Transactions are the processes of isolating certain actions which has taken certain places. The main properties of transactions are the Atomicity, Consistency, Isolation and Durability (ACID). The main type of transactions is the long running one which used to prevent locks on non-local resources with the use of compensation for handling failures.

Communication Closed Layers (CCLs) are used to allow designers arranging the workflow of their business in a form of layers that represent several atomic activities with the use of forward and backward error retrieval approaches. Furthermore, ERP is a system that is able to maintain both the business process familiarised view and the business processes. This system can automate and incorporate the business processes of a company, offer standard information and measures across the whole venture and access data in concurrent environment. Those are the main issues in this work, where an ERP system is developed depending on both LRT and CCL. The following two chapters review some of the related concerning those three concepts.

CHAPTER 3

LITERATURE REVIEW - ENTERPRISE RESOURCE PLANNING (ERP)

This chapter reviews some of the related works concerning ERP systems, particularly about the application of ERP systems in companies, the influence of man machine interaction factors on those systems, the ERP user acceptance in knowledge based industries and the ERP cloud computing.

3.1 INTRODUCTION

Enterprise Resource Planning (ERP) systems can be defined as software packages which try to incorporate each part of a company as well as business functions into a particular computer system that in turn presents the requirements of each part of the business. ERP systems have the ability to incorporate both company functions and processes. Over the last ten years, these systems have become the most accepted standard business software. These systems have gained importance since both process information development and precision have become significant issues. The essential aims of an ERP system are to support and integrate all aspects of the procedures conducted by a company and to generate a system able to provide timely and appropriate data to decision-makers, staff and business associates. However, achieving these aims can be very difficult and expensive. Thus, this chapter reviews some of the related works concerning the obtained achievements in that field.

3.2 ERP SYSTEMS IN COMPANIES

There is a strong danger of various kinds of systemic mismatch, i.e. gaps between the functions that the ERP system is capable of performing and those required by the adopting organisation, in areas such as information, procedure and output, which can greatly reduce the return on its investment [33]. Given their natural complexity, ERP systems can usefully be considered to consist of four main components—software, user attitudes, change management and operating procedures within the organisation—plus a fifth element underlying these four: the method of ERP implementation [34, 35, 36, 37, 38].

The deployment of ERP systems into companies composed of several divisions is considered to be a real challenge since companies do not construct their business in a standard way. Thus, the main intention of companies during the implementation of ERP systems is rarely known. Authors in [39] discussed that three quarters of ERP systems are not successful. According to [40], the most significant breakdown factors are: lack of Pinnacle management support, user training and system announcement. These factors lead to several problems such as business processes control hammering, consumption of time and money and insufficient user acceptance [39, 40].

According to [41], controlling variations in a business process, choosing a software system as well as a supportive end, and deploying the system and exploring the new system ability, all lead to a successful implementation of an ERP system. Authors in [42] discussed that the best ERP systems are implemented by a software business operating in association with a major manufacturing customer in order to improve the software to meet the most important business requirements [41, 42].

Authors in [43] demonstrated that the significant success factors of ERP systems must be understood in order to reduce the high ERP system implementation failure rate. According to [44], the ERP system lifecycle consists of four main phases: a chartering phase that defines decisions and recognises the constraints of the solution and business case, a project phase that develops the scheme and the end user operation, a shakedown phase that fixes and reduces the number of bugs and achieves normal operations, and an "onward and upward" stage which maintains the system, supports the system users, obtains results and improves the extensions of the ERP system. These phases are shown clearly in figure 3.1 [43, 44].

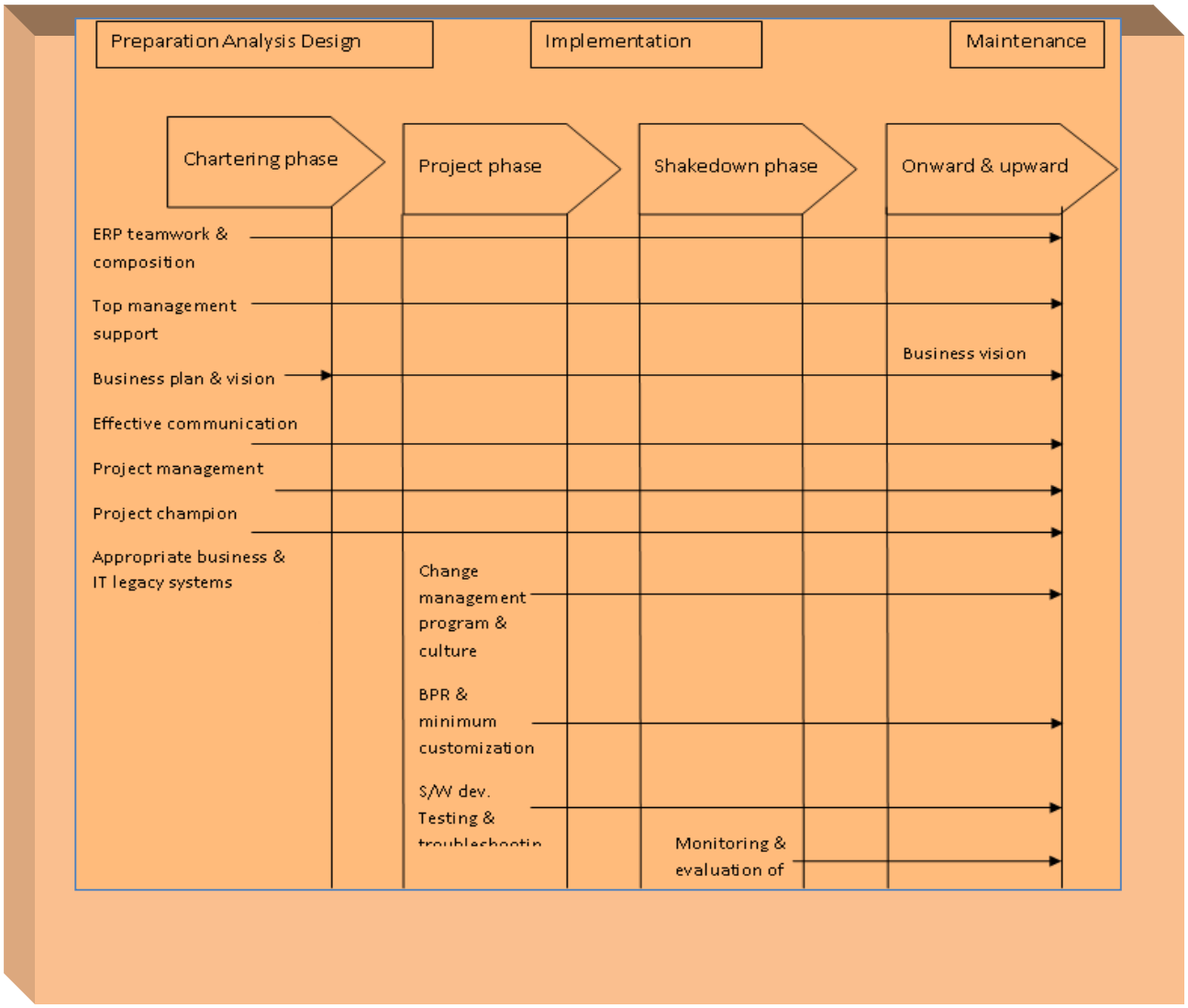

Figure 3. 1 ERP system existence series stages [44]

Authors in [45] explained the classification of independent agents and explained how they are dissimilar from a computer program. The term agent, in literature is also referred to as software agent, intelligent agent, business agent, and autonomous agent. Thus, an agent can be defined by the action of observing its surroundings environment with the help of sensors and then acting upon any observable changes using effectors. The word agent is employed to correspond to computer programs utilising software that

comprises two capabilities; the capability of independent execution and the capability of performing reasoning on the basis of domain. Generally, agent software is a goal-oriented, autonomous, software-based program that works with synchronisation, coordinating and communicating with other software agents as required.

Authors in [46] offered an improved implementation of an ERP system can be obtained by eliminating the incompatibility between various ERP system parties such as the company, its main processes, the ERP system and its end users. This system can be used by users in an effective and efficient way. This system achieves its goal by applying an efficient change management plan for all stages in which incompatibility is an issue. Authors discussed a new ERP system model that focuses on eliminating incompatibility between the system parties in order to obtain an improved system. Figure 3.2 below shows the most common ERP system actors.

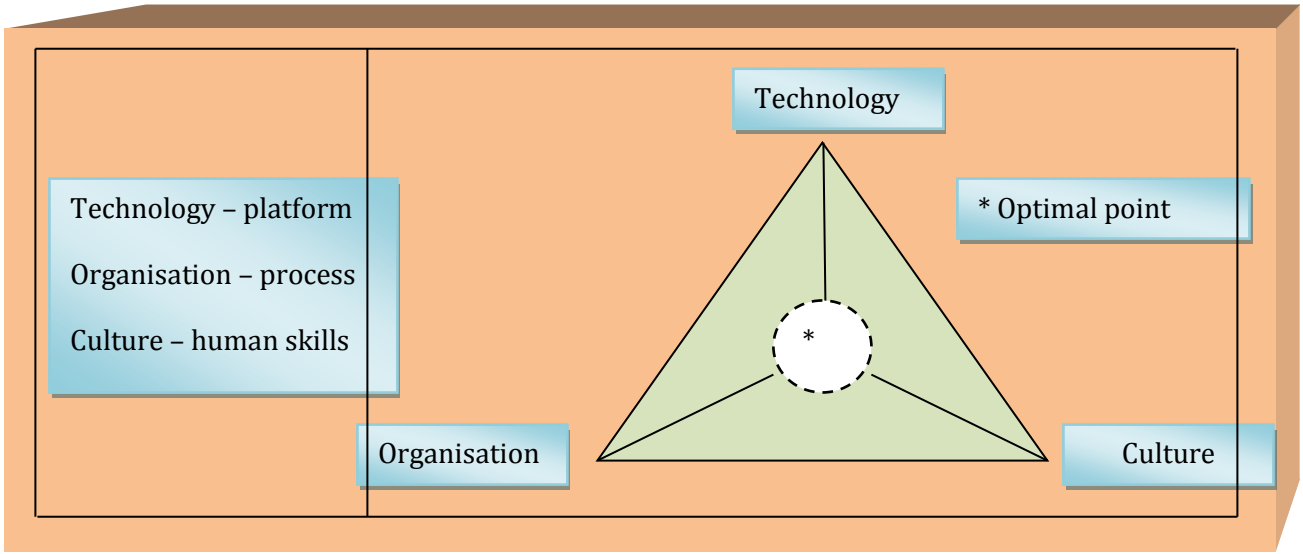

Figure 3. 2 The most ERP system common actors [46]

In this figure, the dashed circle represents the optimal point in which gaps are minimised. Attainment of this point maximises ERP system use. On the other hand, this attainment requires the effort of various players. According to [46], the three main hypotheses of their study are as follows: hypothesis one states that the company and

technology compatibility and the system usage achieved are directly related to each other, hypothesis two states that the human and technology compatibility and the system usage achieved are directly related to each other, and hypothesis three states that the human and company compatibility and the system usage achieved are directly related to each other.

According to [47], the main features of any ERP system are technology and functionality. The vendors of ERP systems are frequently proposing new software and new updated versions of their software packages. Thus, there are various options for users. Authors in [48] proposed that the most significant principle that is used in choosing any ERP system is the optimal fit with the current processes of a business. Figure 3.3 below shows the selection process for an ERP system in which both the IT management and top management have a significant role. The main purpose of both managements is to choose the optimal well-matched ERP system via a specified managerial procedure, since all the ERP systems are generic and to demand that their designs are changed does not seem sensible. As shown below, Product A is the most appropriate one for the company. Selection of Product B results in the company carrying out additional work in order to attain the optimal point, at which the best product (Product A) and the organisation interconnect [48].

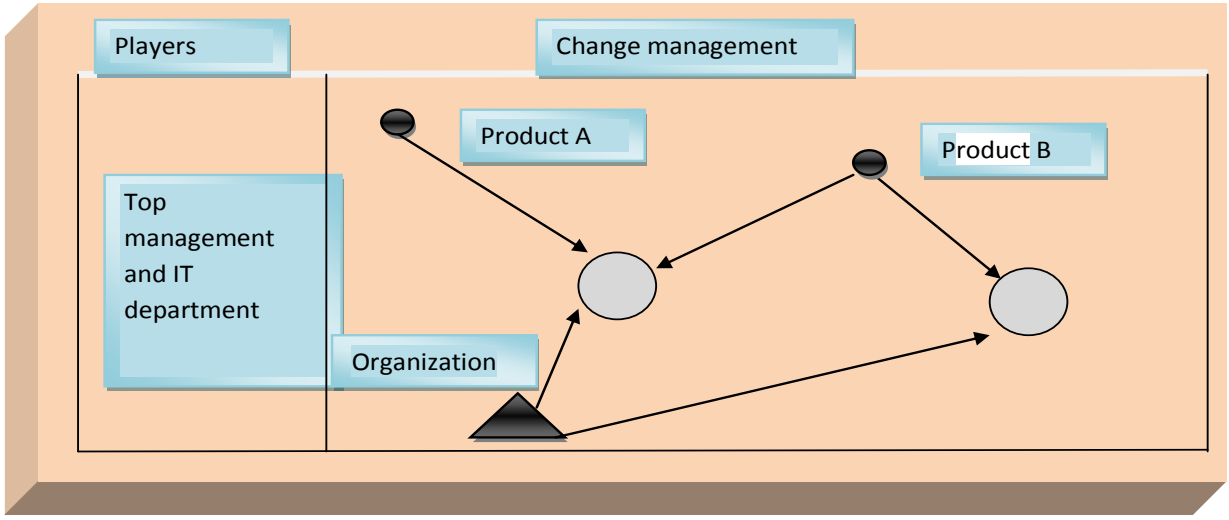

Figure 3. 3 The selection of an ERP system [46]

After selecting the ERP system, the IT management, top management and users have to adopt the selected system into their company. According to [49], both the ERP system adjustment and the managerial processes are iterative procedures that depend on both the company properties and the ERP system properties. As illustrated by [50], both obligatory Business Process Reengineering (BPR) and sufficient customisation, where the processes of the business are implemented in the required ERP system frame, must be carried out. Figure 3.4 below demonstrates the ERP system implementation. The optimal point in Figure 3.4 for the gap between the company and technology is point P_2 [50, 51].

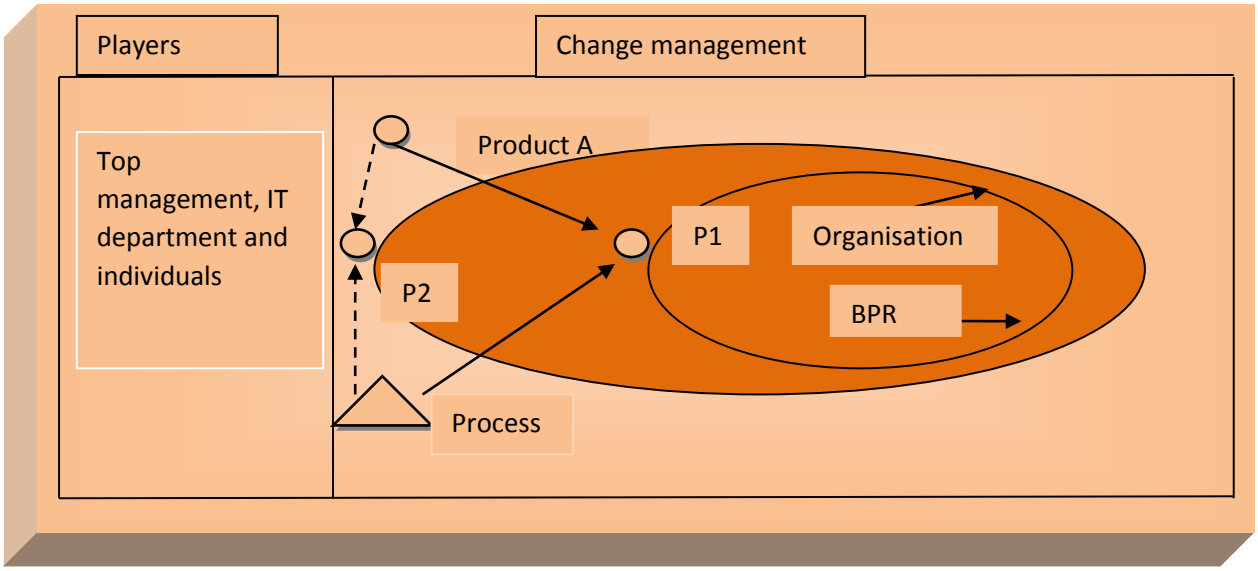

Figure 3. 4 The ERP system implementation [45]

According to [51], companies can achieve several benefits from ERP system customisation at a satisfactory level, such as decreased training requirements, fewer management changes and less user resistance. Conversely, the adoption of an ERP scheme may affect the company and cause fundamental managerial changes, which must be cautiously controlled. As discussed by [52], the implementation of any ERP system software package needs to include the acceptance of various managerial

processes and procedures in order to fit with the main practices of a business. These practices are entrenched in the implemented software package. On the other hand, fixing the ERP system by changing the code is not ideal. Usually, system improvements are created when there are managerial processes which are not reengineered [51, 52].

Individuals are important actors who must be considered in addition to both the company and technology. The new software package proposed for individuals will be inappropriate for them in some respects. According to [53], all teams using an ERP system expose a high level of managerial resistance because of the difficult changes involved.

Authors in [54] discussed how premature introduction to an ERP system forms the views of the ERP systems users which do not change when the system is completely ready. Thus, when the ERP system is first implemented, the company IT management team must provide training, discuss and promote the system with users. In addition, this team must hire and fire employees in order to prevent any user resistance. According to [55], users who are familiar with the processes of the business are chosen from several departments. These users have to improve the key system requirements, serve as trainers, assist consultants, trainers and office staff, and act as representatives for the system end users. Thus, these users have a significant role in the adoption of the ERP system. Figure 3.5 below shows the management of human resources.

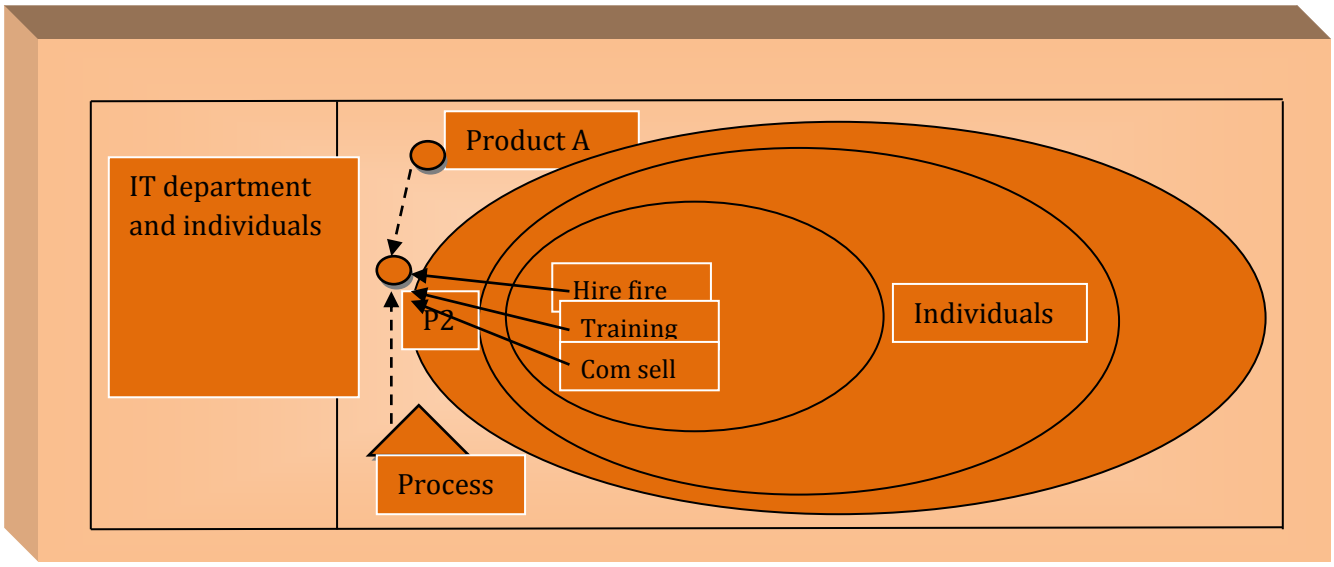

Figure 3. 5 The management of human resources [43]

After implementing the ERP system, the company must carry out the activities proposed above. Therefore, several issues will occur, such as a gap between the company and its employees and the way of making business changes. Authors in [56] proposed that perceived helpfulness and ease of use have an effect on the use of the ERP system. When the system users are inclined to learn the implemented system, they will have the ability to understand and accept the new management system processes. Thus, key users must be selected from various departments in order to put together training, and to communicate and sell the implemented system. The selected users have the ability to ease the understanding by individuals of the system processes.

3.3 Effect of Man-Machine Interaction Factors on ERP Systems

Enterprise Resource Planning systems combine functions and processes by using a single data depository across the company. ERP systems are now used as the central engine of a framework intended for many business functions and processes, and also these systems are obtained as simple and easy software packages. Although the software vendors must differentiate their products, the adopters of ERP systems have a huge

number of alternatives when they go through their selection process. To achieve competitive benefit in the market, many policies must be applied by software vendors. One of the most important applied policies is the area of man machine interface [57].

In [57], a scheme, entitled "Conceptual model of factors affecting end-user satisfaction with ERP systems" has been proposed. This model shows the effects of interface usability properties, perceived value and usefulness, and perceived ease of use, and the effect of these on end user satisfaction was explored. Figure 3.6 shows the theoretical schemes of elements affecting end user approval with ERP systems.

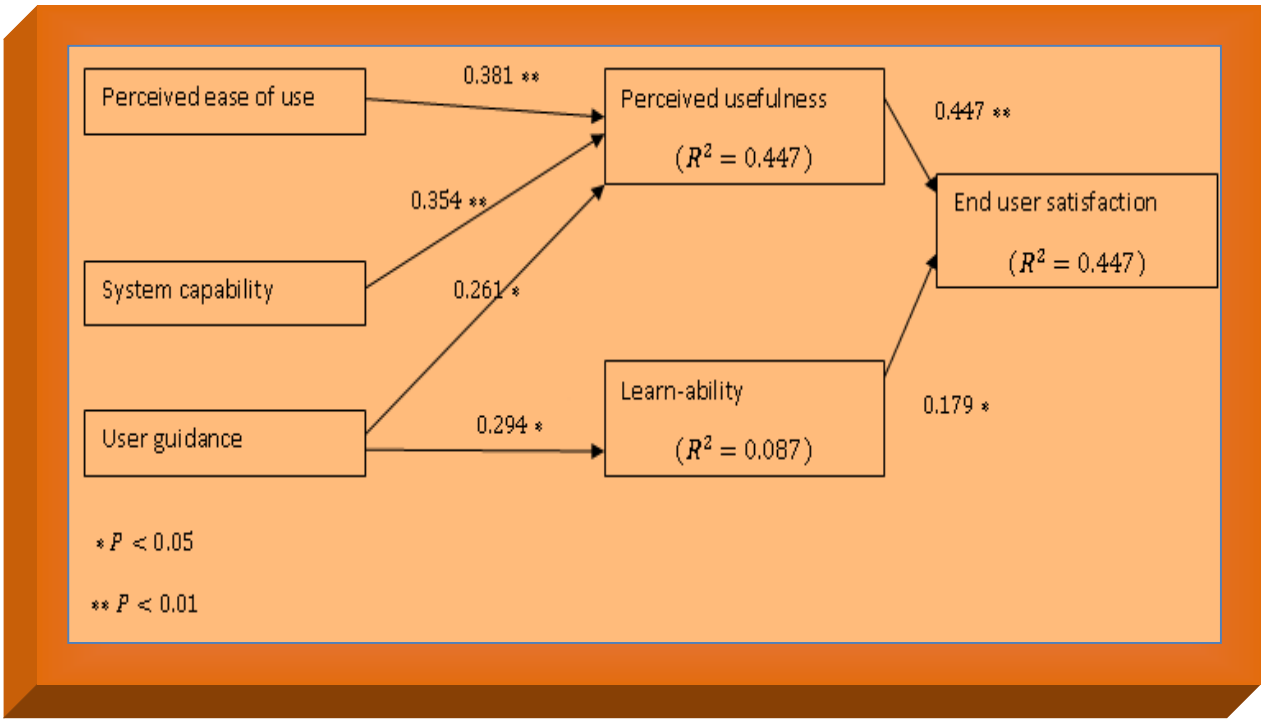

Figure 3. 6 A Theoretical schemes of elements affecting end user approval with ERP systems [57]

In [57], a theoretical scheme to predict end user satisfaction for an ERP system was investigated and improved in the study. Based on their research, the main distinctive features of their model are that learn-ability and perceived value are determinants of end user satisfaction with ERP systems. Of these features, perceived value has the strongest

effect on end user satisfaction. The two most important models of ERP systems are the Technology Acceptance Model (TAM) and the Task-Technology Fit (TTF) model. Davis planned the Technology Receipt Scheme (TRS) shown in figure 3.7. The technology receipt scheme concentrates on attitudes towards the use of an information systems product which the users held based on the perceived value and perceived ease of use of the product.

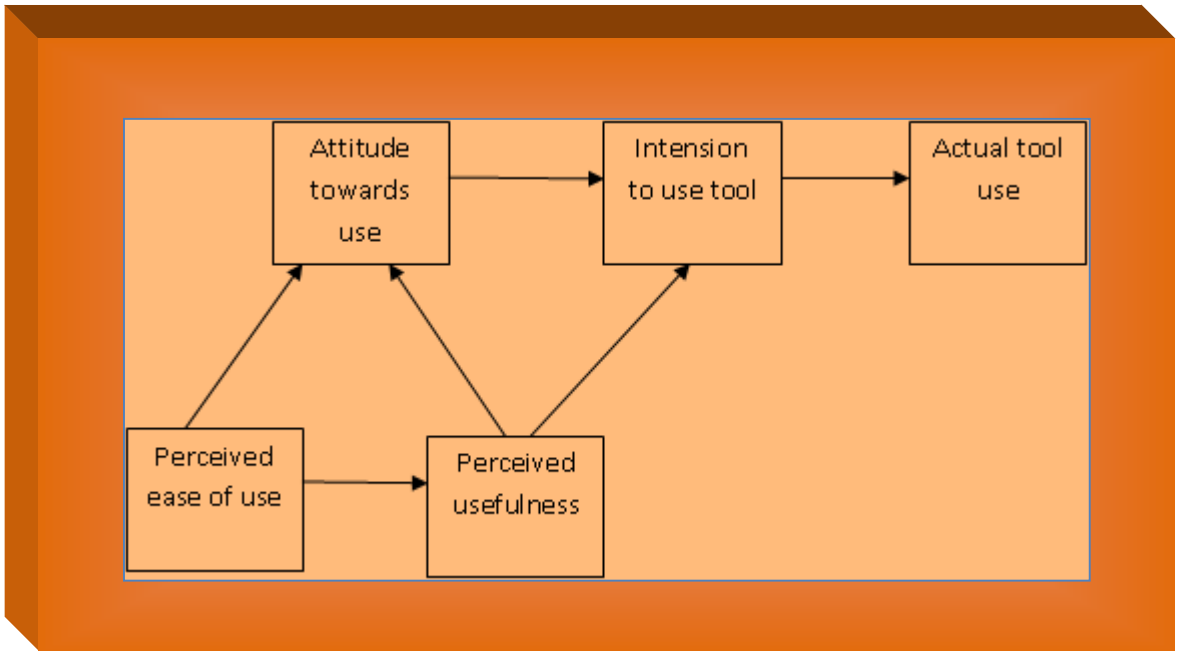

Figure 3. 7 Technology Receipt Scheme (TRS) [57]

As illustrated in [57], the technology receipt scheme (TRS) has a disadvantage, which is that the user task requirements are not taken into consideration, which means that the user task requirements are not considered important in the technology receipt scheme. This is where the task-technology fit (TTF) model comes in. At the heart of the task-technology fit model is the official construct identified as TTF that is shown in figure 3.8, which identifies the capabilities of the tool with the requirements of the task, which means the capability of IT to support a task.

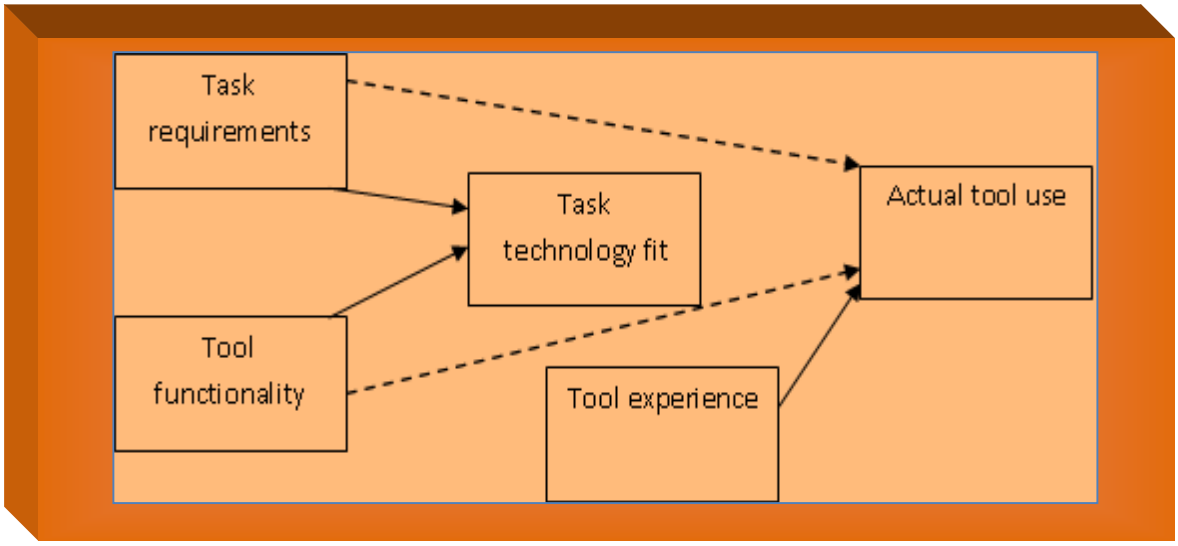

Figure 3. 8 Task-Technology Fit (TTF) Model [57]

According to [57], in order to make it easier to understand the user, the TRM scheme and task-technology fit model must be integrated. The relationship between task-technology fit, ease of use, flexibility, navigation, relieve of utilise, data quality, visual factors, minimal memory load, hypotheses, and satisfaction is built up. Figure 3.9 below illustrates the model of the factors affecting end user satisfaction, which was implemented based on the models discussed in the literature study.

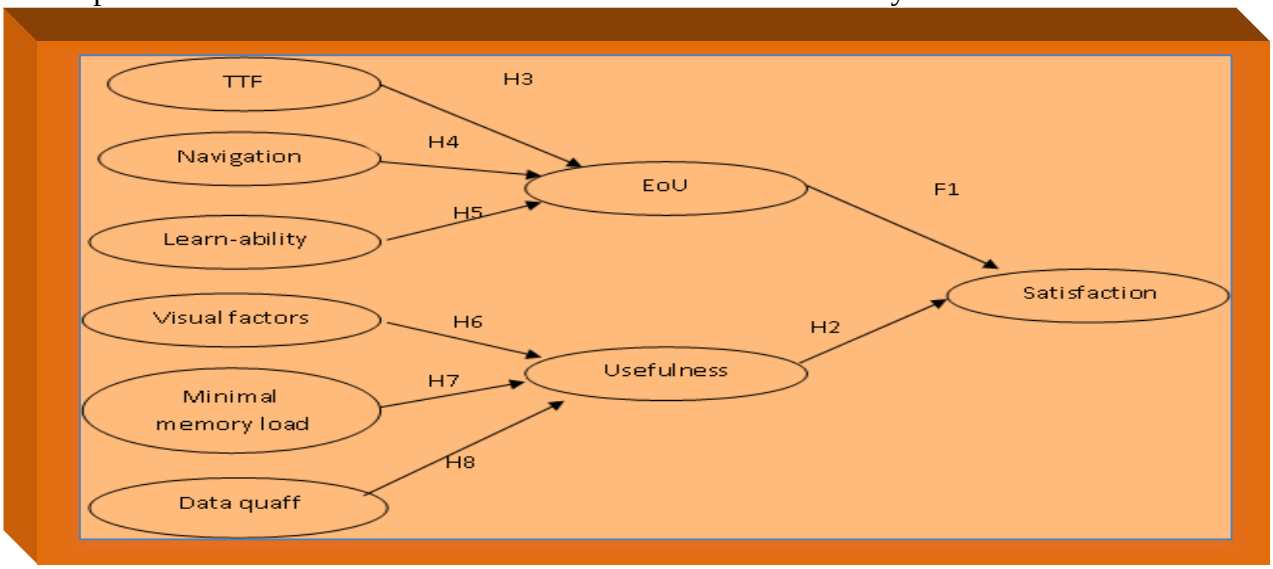

Figure 3. 9 Model Framework [55]

3.4 USER ACCEPTANCE OF ERP IN KNOWLEDGE-BASED INDUSTRIES

An international review of ERP system success should also consider changes in knowledge-based industries, which are significant in the domain of executive development. Unlike process-oriented industries, knowledge-based ones such as construction, engineering, entertainment, media and shipbuilding have dissimilar tasks connected by means of their business procedures. Each assignment that they undertake is unique and involves many stakeholders collaborating in order to achieve certain aims related to a one-off combination of facts and data. The quantity of soft data and its time-responsiveness tend to make these businesses hard to supervise, suggesting that the outdated legacy systems need to be changed to manage extremely disjointed, and non-uniform and organically isolated business procedures. Lately, some knowledge-based companies have begun to consider ERP adoption as a reliable way forward. A number have already completed implementation, while others are in the process or are considering an ERP system. Major ERP dealers like Oracle have begun to supply specific solutions for knowledge-based customers [12, 13, 27, 58].

Authors in [58, 59] proposed that it is significant to consider that choices related to ERP adoption are made by higher level directors not considering the larger picture and that most of the end-users are concerned only in the later stages of the plan, such as training. This leads us to a wider question: do end-users have the same understanding as top managers of ERP system success? There is evidence that this is not always the case; indeed, the two parties often have dissimilar viewpoints concerning the needs of the system.

Furthermore, it is reported that end-users are usually less persuaded of the value of any recently-implemented complex system, resulting in rejections or underuse. Investigators have identified such user-related issues as presenting a serious threat to ERP

implementation plans, notwithstanding their complex character. That is to say, even if a system is adopted on time and within budget, it will be rejected if users consider it worthless in terms of their work procedures or if it takes them a long time to work out how to use it properly [59, 60, 61, 62, 63].

The idea of developing user acceptance seems to be more difficult. One reason for this can be explained by the conventional environment, irrespective of the business, meaning that outdated legacy systems are not simply abandoned if clients are very sure of the need for a new one. Furthermore, the fact that many clients are not IT specialists might keep them at the extremely basic functional stage of work procedures. There is a need to consider their technical ability and little contact with the best practices offered by ERP systems, which makes it hard for them to contribute to such systems. In extremely disorganized knowledge-based firms, many stakeholders of various kinds may come and go, while the knowledge continues to be held collectively. It is highly likely that inter-departmental relationships are necessary to support the flow of data and contact. Therefore, there must be a dissimilar example that includes input from end-users who value their acceptance, to communicate the varied levels of user acceptance, particularly in knowledge-based industries [64, 65, 66, 67].

Until now, very few empirical investigations have been done to examine the acceptance of technology from a user viewpoint, let alone that of ERP systems in knowledge-based organisations. Ignoring users' opinions may lead to an unthinking use of the system, as a result of which the organisation may fail to achieve its expected returns from its costly investment. An exclusive focus on process-oriented enterprises may also make it difficult to provide a body of empirical evidence across a selection of organisational situations. Given the huge body of information on ERP systems, it can be said that there is an important gap in the present literature regarding the knowledge-based sector, since our area of concern is to identify the ways in which knowledge organisations perform, consistent with end-user sensitivity to knowledge and ultimately with their performance

in using the ERP system. Simultaneously, account should be taken of how environmental and social factors influence the actuality, especially wherever many stakeholders come together to use organisation-wide structures [68, 69].

As a result, the Technology Acceptance Model (TAM) proposed by Davis is planned to be extensively used in order to search for interactions since it is the model most related to this investigation in addition to being the most dependable since it is a theoretical model. TAM looks at the perceived value and perceived Ease Of Use (EOU) as the main limitation of IS usage. Depending on TAM, many researchers have tried to recognise outside issues that affect internal TAM variants. Nearly all of the studies, however, have been mostly paying attention to comparatively simple systems like office automation or email. Recent studies of ERP structure have paid attention to internal management tasks such as training, or have included knowledge and educational parts in TAM. There has been an effort towards connecting product value and management style to ERP users' perception feedback too. However, not many have studied different managerial backgrounds or measured the effect of social and environmental issues and outside influences (such as consultants) on the decision by users to accept the ERP structure. With the increasing problems of IS linked with cross-functional business procedures and the many types of user involved, it is expected that study on achieving successful IS deployment would benefit from expanding TAM to a complex business function like ERP structure [70, 71, 72, 73, 74, 75, 76].

3.5 ERP CLOUD

Cloud computing is currently considered to be a hot topic in modern business. More or less every IT corporation presents a number of types of cloud manufactured goods or services and almost every IT specialist uses a different meaning of the expression "cloud computing". For a traditional individual customer, cloud computing means applying a

Web-based service, for example online services for data storage, word-processing, e-mails, worksheets, teamwork, file sharing, and social media [77].

The universal concept of cloud computing is connected to an on-demand service model whereby a number of diverse resources are connected on an "on the fly" basis. The service(s) are transported over the network, which could be the intranet of a corporation or the Internet when the service is provided by an outside supplier.

The service feature of the cloud contains three different parts (applications, systems software and hardware) which can be united to construct a cloud-specific service package or offering. Depending on how a cloud supplier unites these parts inside a cloud offering, there is a number of different cloud service levels.

At present, there are four possible cloud service levels that can be combined to construct a complete back-to-back cloud offer:

- Software as a service — offering an application such as ERP on demand over a network or the Internet.
- Platform as a service — offering a complete development platform containing the essential integral services, like MySQL databsase, GlassFish application server, LDAP, NetBeans software and Oracle Solaris Studio, on demand over the network.
- Infrastructure as a service — supplying hardware and software infrastructure parts, like calculation, storage space, systems, Oracle Enterprise Manager Ops Center, Sun Management Center and Sun Identity Manager from Oracle.
- Desktop as a service — moving the desktop environment of a cloud customer into the cloud and offering secure remote access to server-based applications. It helps to reduce administration costs and establishes superior security, as IT employees can run applications from a central console to consumers to whom appropriate access is granted according to individual or collective criteria.

3.6 SUMMARY

ERP systems composed of five main components in companies; software, user attitudes, changes management, operating procedures and the method of ERP implementation. Companies can achieve various benefits from applying those systems, such as decreased training requirements, fewer management changes and less user resistance. However, the application of those systems in companies is a real challenge because companies do not construct their business in a standard way. Furthermore, those systems may influence on companies and result in fundamental managerial changes, which must be cautiously controlled. Thus, various researches have been proposed and conducted to study the benefits of applying ERP systems in companies and find solution for those problems.

Development and acceptance of any ERP system by users is considered to be a real problem. It requires the development of the ERP system by changing the interface and developing both processes and individuals by recognising their important roles in reliable and improved managerial processes. The main factor that has an important role in ERP system usage is the compatibility of three main factors: technology-company, technology-human and human-company.

Cloud computing is one of the main topics in modern businesses, where it represents the application of a web-based service, such as online services for data storage, word-processing, e-mails, worksheets, teamwork, file sharing, and social media. It can be constructed based using four cloud service levels; Software as a service, Platform as a service, Infrastructure as a service and Desktop as a service

Some of the major advantages of ERP systems are reduced vendor dependence and cost savings. These and other advantages are widely recognized but so far they have often been offset by certain shortcomings. There are critical issues within ERP system such as the closed information typical in proprietary software, the need for specific functionality for the cities' demands and the need of ERP proficiency.

ERP system systems represent a critical choice in proprietary administration software, so the system is needed to be enhanced within ERP schemes. The lack of functionality for pubic strategy of existing ERP systems is a serious limitation for large cities and even for smaller towns in various places. Local authorities, with their recurring cutbacks, restricted resources and reliance on public funding, are possibly not the most adventurous adopters that will illustrate the method for ERP systems.

Particularly, the main problems of the available ERP systems are the complexity, limited customization, the need for high costs, difficulty in managing the whole business processes in those systems and those systems are very difficult to get used to certain workflow and business process of various companies, where this causes failures in those systems. Therefore, a long running transactions (LRTs) model is implemented in this work to facilitate analyzing the workflow of an ERP system to control all its processes and guarantee that completed transaction in a business process is not processed in any other process.

The LRTs have two main problems which resulted from the inability of some actions in the long running transactions to be reversed; rollback and check-pointing activities. Thus, to enhance this work and solve those problems, Communication Closed Layers (CCLs) are used to decompose processes into layered forms that can be analysed using simple techniques for sequential programs.

CHAPTER 4

LONG RUNNING TRANSACTIONS (LRT) AND COMMUNICATION CLOSED LAYERS (CCLS) - STATE OF THE ART

This chapter reviews some of the related works about Long Running Transactions (LRTs) concerning the atomicity and recovery in Long Running Transactions (LRTs) and some of the recent published works about them. On the other hand, this chapter offers a review concerning the recent published works about communication Closed Layers (CCLs) and their general methodology of work. Finally, it explores the application of Layers in LRTs as well as the application of LRTs in ERP Systems.

4.1 INTRODUCTION

Recently, various researches have been proposed concerning the Long-running transactions and their importance in preventing locks on sources and rollback problems and handling failures that result from small ACID transactions using compensation. Another technique that has been extensively studied in the recent decades is the Communication Closed Layers (CCLs) due to its importance in scheduling real time tasks in restricted environments. Due to the importance of those two techniques, this chapter reviews some of the related works concerning them and their applications, especially in ERP systems.

4.2 RELATED WORKS CONCERNING LONG RUNNING TRANSACTIONS

4.2.1 ATOMICITY IN LONG RUNNING TRANSACTION

Long Running Transactions (LRTs) include tasks that can be executed in sequence and in parallel. They may include sub–tasks and also may be obliged to complete before a specific deadline. The LRTs are not atomic and a compensation mechanism has to be applied when an execution is not completed. [20]

Transactions guarantee a realistic mechanism for management, which must ease both the coding and design of various concurrent systems. In particular, in shared memory concurrency, the systems that depend on the transactions can develop a fine grained locking capability, while decreasing the chances of deadlocks and race conditions. Although they have many recognisable advantages, progress in low level transaction implementations is not sufficient and needs related programming techniques and languages. On the other hand, although long-running transactions can lead to extreme

differences and the use of obscure hardware that is based on different implementation strategies, these transactions sustain a conventional programming style where transactions in combination with their guarantees are the default [78, 79].

In [78], authors introduced the Automatic Mutual Exclusion (AME) programming model which is similar to the supportive multithreading in the planned implementations. Nevertheless, the transactional memory of the software supports the implementation of synchronised atomic fragments. In addition, authors studied both the simple static and dynamic atomicity check mechanisms in the AME programming model.

In [80], in both service oriented architectures and web services, the orchestration and flow composition languages are utilised in order to merge different services into integrated applications. In [81], this differs from the colossal applications that are built on a single dataset, like the composite applications which have difficulty in obtaining the benefit of atomic transactions to guarantee that modifications are either completed or lost completely. In general, these applications improve the use of the compensation actions which are applied in order to undo the previous dedicated actions. The flow compensation languages have the ability to offer several compensation operators that relate compensation actions in combination with fulfilled actions and then run them in a failure event.

The published works about the utilisation of flow composition languages and the development of transactions are concerned with the formalisation of the compensation semantics [82, 83, 84]. Conversely, there are few attempts to define a rightness notion that is suitable for long-running transactions. In [85], there is a good definition and application of an independent principle of rightness for atomic transactions, which is "the Atomicity, Consistency, Isolation, and Durability (ACID)". However, these ACID transactions are not appropriate for flow work.

In [86], the author defined the proposed four ACID properties of transactions. Atomicity means that transactions perform totally or not at all; this is called the all or nothing principle. Consistency means that transactions conserve the interior database reliability. Isolation means that the transactions perform as if they were running alone without any other transaction. Durability means that the transaction results will not be missing in a failure.

In [81], atomicity is one of the four ACID properties of the datasets. It means that transactions either happen completely or revert back to the previous state. This means that the atomicity follows the all or nothing principle which means that the transactions prevent a partial database update. Atomicity is used widely in the design of various systems with the use of certain mechanisms.

In [81], authors explored the fact that the transactions that create several web services are frequently LRTs, which means that these transactions get a considerable time period over which to complete due to the asynchronous interactions across several systems which may need human interaction. Therefore, to assist both the participating systems stretching coordination and locking is not possible, the web services sometimes expose several dispersed transaction interfaces because of the application restrictions, protocol restrictions and the organisational boundaries. Therefore, looser notions of rightness must be taken into account.

In [87], authors explored the idea that a LRT must be completed effectively or all its changes must be lost. The process of losing these changes is a compensation action that is specified by the programmer. In [87], authors explored the example of sagas. Sagas are transactions that consist of two stages with compensation actions for each one of the forward actions. They attain atomicity by resolving the other system needs.

In [81], it was found that flow composition languages allow the building of several LRTs by using various independent, atomic services. However, these transactions cannot be made to match the typical ACID semantics due to environmental restrictions. Thus, authors in [81] proposed set reliability, an influential, and up until now instinctive, reliability notion for LRTs. The set reliability properties are considered as predicates over the process atomic actions and they require the execution of these actions along many system traces to satisfy the set reliability requirements. In addition, set reliability simplifies the termination semantics, the typical reliability requirement of LRTs in which the unsuccessful processes are responsible for undoing actions that are partially completed. Furthermore, it can articulate much stronger requirements, such as dependency or exclusion. In [81], authors believed that these predicates are helpful for the representation of the constraints of the real world and are instinctive for developers.

In [81], authors formalised a centre calculus for the representation of LRTs, offering both chronological and equivalent composition and also compensation actions and exemption handling in the language. They illustrated the corroboration problem of set reliability for many processes in the centre language versus the reliability requirements that are offered as Boolean predicates achieves, and present an algorithm for confirming set reliability via decrease to the related satisfaction. Authors designed this algorithm and then illustrated the value of their approach on three different cases where in each one of the cases, the needs of the reliability could be confirmed.

4.2.2 RECENT PUBLISHED WORKS ABOUT LRT

A lot of research has been conducted on transactions of long duration. Among the most influential models of long duration transactions, saga, which was suggested by Garcia-Molina, is considered to be the most influential. This model is based on the idea of decomposing every transaction of long term duration into a multiple short duration transactions, which can be performed independently. Every short duration transaction is

a sub-transaction and is considered to be a traditional transaction. When the transactions with long duration need to be rolled back, the compensating transactions are used. In Garcia-Molina's model, when a transaction of long duration must be rolled back, not just the compensation of executed sub-transactions is required, but every other transaction that either indirectly or directly utilised the intermediary outcomes of the transaction with long duration must be compensated as well. As those transactions which utilise the intermediary outcomes of a transaction with long duration cannot be forecasted in advance, the compensation transactions cannot usually be developed in advance, and the process of compensation has to be dependent on human interaction in the majority of circumstances.

For improvement in isolation and atomicity for transactions of long duration, several other models of transactions of long duration employ a check-in/check-out mechanism of concurrency control. These types of models are mostly used in the areas of Computer-Aided Software Engineering (CASE), Strategic Information Systems (SIS), and Computer-Aided Design (CAD). The fundamental idea for the mechanism of check-in/check-out is as follows: whenever a transaction of long duration wants to operate on some particular data in a system's database, it firstly checks out the database's data into its own space of local data, and then it executes the process on the data in the local data space. After the whole transaction of long duration is finished, the model checks the complete data from space of local data back into the system's database. Throughout the period of execution of the transactions of long duration, other transactions might have modified the data in the system database, in order to integrate the database data and the data relating to the local space into an incorporated version of data after the check-in of the data has been done. The major disadvantage of this method is that there is a lack of formal definitions of the data, and therefore the criteria for correctness cannot be mathematically characterised. The users are required to correct errors and solve conflicts that might actually be very complicated for them. As a result, this method is not appropriate for various enterprise applications [86].

The model combines the semantic knowledge of transactions with a technique of nested transactions and multiple versions for supporting transactions of long duration. This model offers strong capability to communicate multifaceted interactions, and suggests multiple criteria for correctness of scheduling concurrency. However, the model employs compensating transactions for rolling back transactions of long duration that require every sub-transaction to have a subsequent transaction for compensating, which is not practical for a lot of applications in the real world.

The model supports two kinds of mechanism for concurrency control, the most important being the mechanism of transform/predicate. The key idea behind the mechanism of transform/predicate is to suspend the execution of actions in a transaction of long duration until the committing time of the transaction, and after that carry out these actions as a batch job. A transaction with long duration can thus be transformed into a short traditional transaction and therefore can have similar durability, isolation, consistency, and atomicity semantics to conventional transactions. For instance, in a typical application for Internet shopping, subsequent to the decision of a customer to purchase a product and entering the purchase amount, the web application just keeps the order information in the customer purchase session, rather than immediately updating the database. Just after the consumer has decided on the purchase of all the items they need and has finished the process of payment, the web application will update the database. The process of shopping by the customer can be considered to be transaction of long duration, in which the mechanism of concurrency control is transform/predicate.

The major drawback of the transform/predicate mechanism is that it cannot guarantee the priority of a transaction of long duration when it clashes with short traditional transactions, although the failure cost of a transaction with long duration is generally very high as compared to the traditional transactions with short duration. Furthermore, it cannot inform the user of the failure of a transaction of long duration in an appropriate way. For instance, in the earlier example of an Internet shopping application, subsequent

to the end of the customer ordering process for a service or product but previous to the completion of the transaction, another transaction might modify the product stock to a quantity which is lower than the customer has requested. This modification in stock is not acknowledged to the customer, and they might carry on ordering other services or products without being aware of the shopping session's failure until they attempt to complete the process [82, 83, 84, 85].

4.2.3 RECOVERY IN LONG RUNNING TRANSACTIONS

The continuous growth of computer systems in the recent years is resulting in increasing the failure possibilities and the expectations on their main reliability. This issue increases the need for finding efficient error recovery methods for solving failures. Generally transactions allow both handling the failure propagation in various concurrent systems because of the dependencies and returning the system for the original point before the occurrence of failure. Practically, in many settings, mainly during the interaction with real world, the reversal process cannot be performed. The compensations notion has been used to address this issue through specifying activities that can be operated for undoing partial transactions. But, there is not any accepted theory where the previous works provides an excess for distinct approaches and formalisms [26].

4.2.3.1 Error recovery techniques in Long Running Transactions

The main used method for fault tolerance is the error recovery method which aims to fix the state of the system upon the detection of errors to allow normal execution proceeding unhindered. The main error recovery methods are the forward and backward methods. The forward methods aim to correct errors before proceeding, while the backward ones aim to correct the system state before proceeding. The main difference

among the forward and backward methods is that the backward method utilises the execution context in fixing the error where this is not required in the forward method. On the other hand, the backward method can be seen as a particular form of the forward method where further data structures are utilised in storing the execution history. Conversely, the forward method can be seen as an optimization of the backward method where the recovery path is assumed with no need for keeping a log about the previous actions [26].

The forward recovery method is mainly helpful when the failure is essential for determining the best solution to be applied. One of the simple methods that can be used for encoding the forward recovery in various recent programming languages is the utilisation of exception and error event trainers. The used code in the recovery acts as a compensation intended for fixing the occurred problem. The exception trainers are aware of failures that occur in a certain code block only. Performing a backward recovery depends on the continuous tracking of the old transitions or states of the system which include recording past actions or data that have been performed [26].

4.2.3.2 Compensations in Long Running Transactions

When the expected reliability of a certain system increases, the number of precautions that must be taken into account increases too, but, it perhaps introducing further complexity and unreliability sources. The situation can be aggravated via various faults sources in computer systems, such as operating system, hardware and computer system faults. Compensations have been presented since more than forty years [88] as a forward recovery form [89] that aims to correct the system state before proceeding. Authors in [90] proposed the compensation of transactions as a further layer on the top of ACID transactions, while authors in [91] proposed them as sagas. Mainly, its argument is two folds; long running transactions that turn into ACID transactions as resources for long time periods are infeasible in the highly concurrent systems and the ACID transactions

cannot sustain the transactions nesting, thus, compensations are used as counter-transactions where transactions can be nested and turned into sagas.

4.3 Related Works concerning communication Closed Layers (CCLs)

4.3.1 Recent published Works about Communication Closed Layers

Communication Closed Layers (CCLs) are used in many distributed systems to decompose its distributed development processes into layered forms which can be then easily analysed with the use of techniques for sequential programs. Mainly, the use of communication closed layers requires defining the analysis and design methodology of layers. Analysis methodology includes converting a system workflow design into comparable communication closed layer design and generating a layer design that is known as quasi-sequential, while the design methodology includes the decomposition of requirements, design of layers and integration.

Communication Closed Layers (CCLs) have been studied and evaluated by various researchers. Authors in [92, 93, 94] studied the CCLs notion and embedded it in a refinement setting. Authors defined the CCLs notion among actions in models of sharing variables with taking into account difference programs composition where actions which do not diverge can be concurrent and actions that can diverge must be operated in sequences. With the use of this CCLs notion, authors proposed a refinement frame for improving the distributed and concurrent applications from qualifications to designs. Both the applicability and power of this frame were illustrated in various researchers as case studies, such as the work proposed in [95, 96] which focused on extending this work based on considering the epistemic logics for defining CCLs. Authors in [97], proposed that the CCL definition is generated in terms of the whole program as context.

Authors in [98] used the CCLs concept for distributed systems to schedule actual tasks in tightly clear concurrent programs. The proposed scheduling approach sustains an inaccurate calculation method; therefore, it can be utilised in actual systems in which deadlines are achieved with appropriate accuracy. The CCLs based scheduling approach is mainly flexible since the given priorities for tasks in a certain layer can be managed using various factors. The scheduling logical units are defined in the implementation stage with the use of layering; therefore, programmers have more control about the program scheduling than writing programs which outfit a certain scheduler.

4.3.2 METHODOLOGY OF COMMUNICATION CLOSED LAYERS

The use of CCLs requires the analysis and design methodology of layers to be defined. As to analysis, a system workflow is assumed to be analysed by involving the spectrum of both validation and verification that ranges from simulation validation to the checking of various known properties, such as security. The main concept of the analysis dimension is the conversion of a system workflow design into a comparable CCL design to facilitate the analysis process. The resultant layer design, in which all the available formalisms of the sequential systems are used, is known as quasi-sequential.

In the design dimension, a comprehensive statement of the system requirements is assumed to be decomposed into various layer requirements and carried on in a conventional way. It depends on applying three main stages: decomposition of requirements, layer design and integration. In the requirements decomposition stage, the requirements of a specific system workflow are decomposed into various sub-requirements. This stage is considered the most difficult in the whole layer design process, since the effective decomposition of the workflow requirements depends on both the experience and definition of all of the non-functional system workflow requirements. Thus, this stage must be naturally iterative. Experimentally, it was found

that the use of actors can assist in determining layer interfaces. On the other hand, this stage also includes the recognition of all of the channels that are required for the communication and the variables of interest that are required to study the performance.

At the layer design stage, layers that satisfy the defined system workflow requirements are designed using well established techniques. Basically, the layers must be CCLs. At the integration stage, all the designed layers are combined to form a complete system workflow. The whole CCL design methodology is summarized in the figure 4.1:

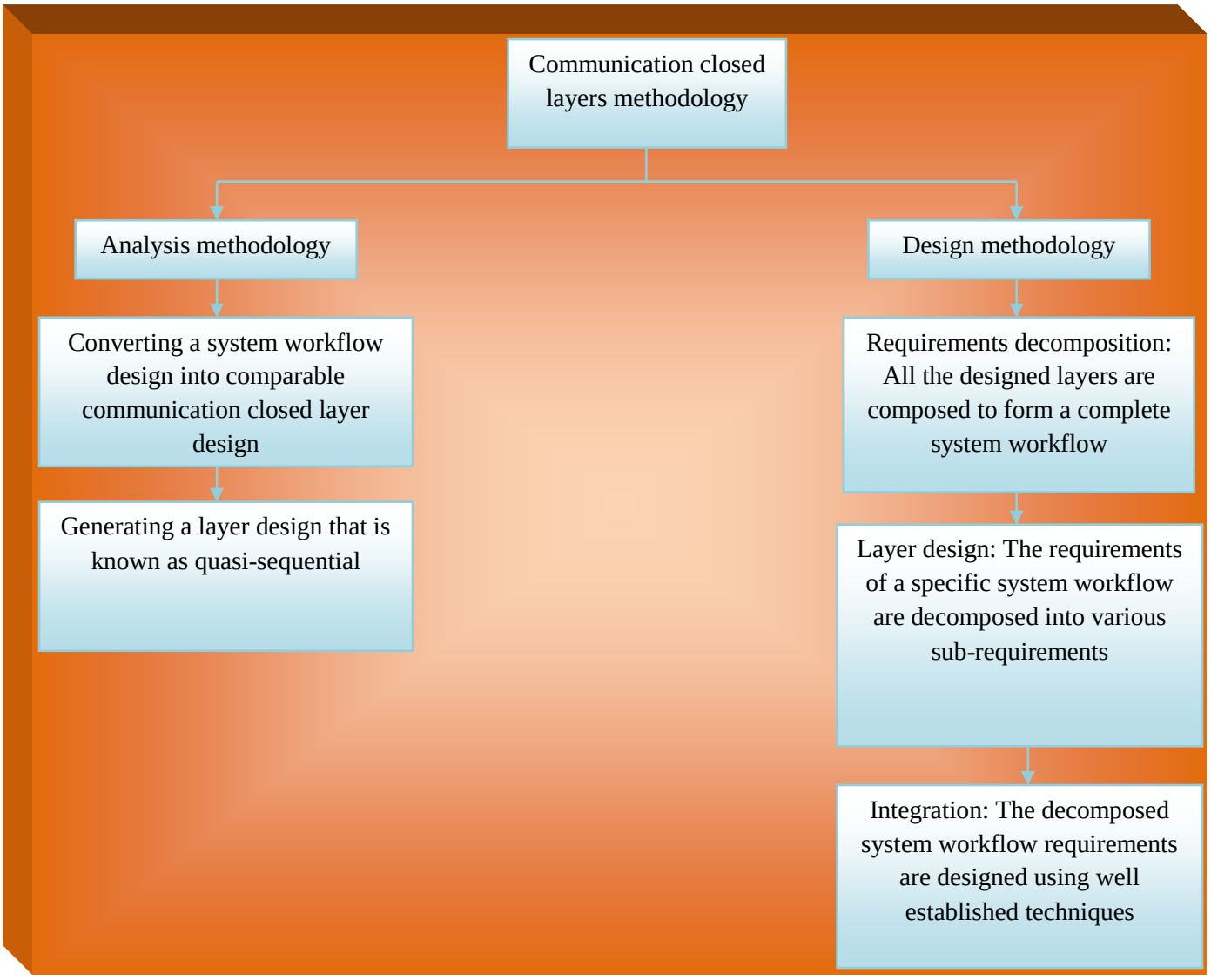

Figure 4. 1 Communication closed layers design methodology

4.4 APPLICATIONS

4.4.1 APPLICATION OF LAYERS IN LRTS

Generally, the applications of workflow management require enhanced transactional control that could not be provided by traditional database systems. To this end, several models have been developed; however, none of these fulfils all the necessary requirements, since workflow management demands various transactional semantics at several process levels. This section introduces a two layer transactional model that can be used to resolve these problems.

4.4.1.1 Model of business workflow management

The model used for business workflow process management uses a multi-level process that permits the decomposition of hierarchical workflow processes to express transactional semantics. A full workflow process is used to form the process hierarchy upper level, while the lower level contains various individual tasks that cannot be decomposed. In contrast, the upper levels apply long running processes that require transactional semantics, due to their cooperative properties; while the lower levels are short running processes that use limited transactional semantics. The business transaction concept in a workflow application generates the division between these two types of levels. This model uses a rollback technique to undo a specific workflow to a specific point where there are errors. This technique must deliver application oriented semantics and so must return the workflow to a certain state that is typical of the previous one.

Generally, the process levels that relate to business transactions are symbolised using atomic tasks, which must be run with limited atomicity and isolation. In the proposed model, all the non-critical tasks must be supported, since they permit the recognition of

processes that do not cause errors or require rollback functionality. This model is decomposed into two layers, where changes in one layer do not affect the other. An example of the proposed model is shown in figure 4.2, wherein the boxes represent tasks and the dotted boxes represents tasks that are below or above the business transaction level.

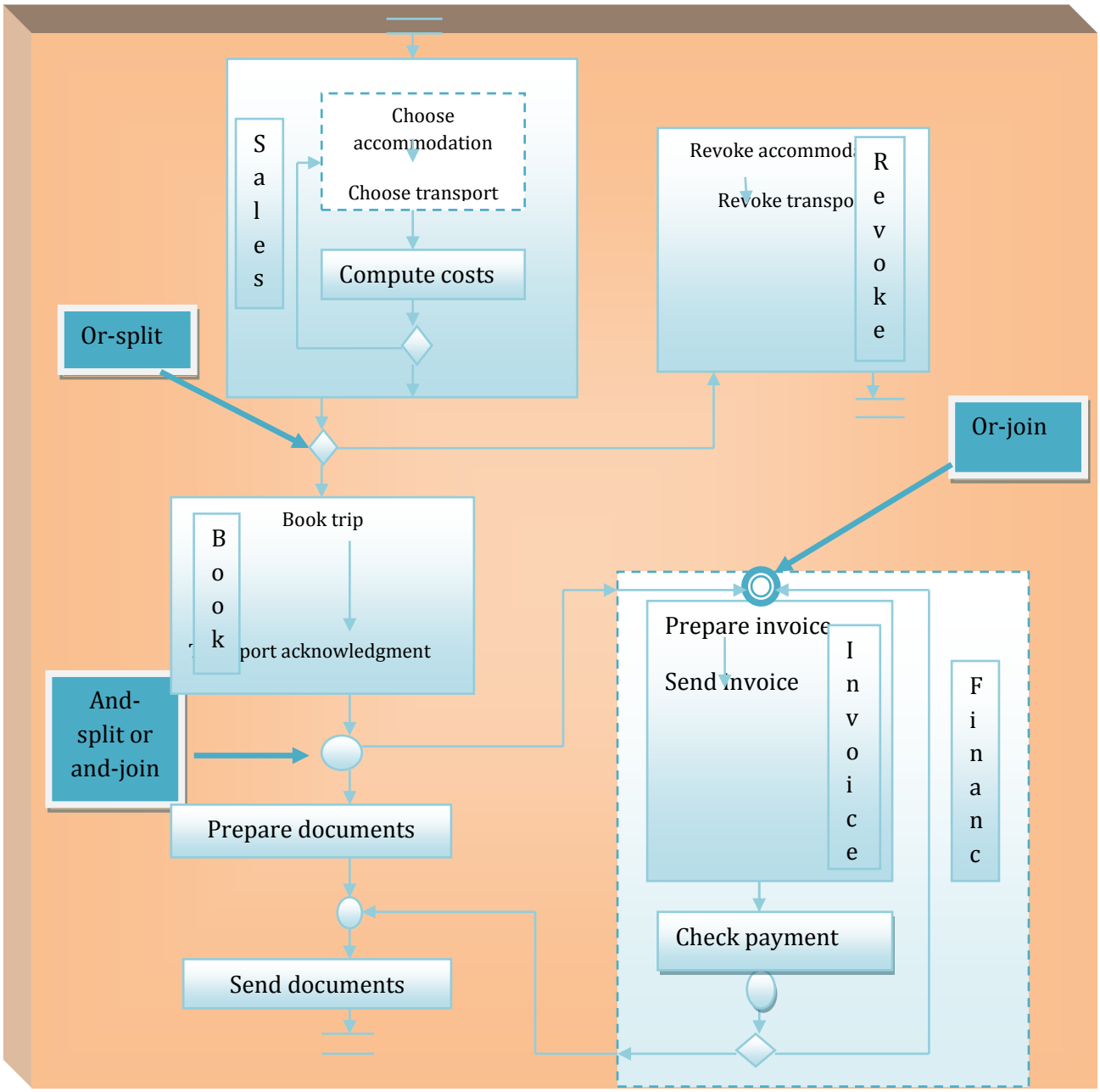

Figure 4.2 Business process example [101]

4.4.1.2 Two Layers Transaction Model

The proposed model in the previous section is decomposed as a two layer transaction model. In this model, the upper layer is generated using global transactions, whereas the lower layer uses local ones.

Local Transactions

This type of transaction is used to support transaction semantics in the workflow processes. These business transactions require limited ACID transactional characteristics. These are used here to generate a lower layer, using a nested transactional model with critical sub-transactions that signify the success of parent transactions and non-critical sub-transactions that do not influence the success of parent transactions. Meanwhile, local transactions offer an intra-transaction concurrency control that can be used to regulate data access among the concurrent and sub-transactions of the local transactions.

The following figure 4.3 shows an example of local transactions. In this figure, the sub-process sales is considered as the local transaction of the workflow where this local transaction sells a trip based on choosing the details of both the transport and accommodation for a certain customer and offers price for the choice. It includes two sub-transactions; choose trip' and calculate costs'. The first sub-transaction includes two main tasks, while the second sub-transaction is a main task by itself. One can have critical and noncritical sub-transactions in the local transaction; the critical sub-transaction decides the achievement of its main parent transaction, while the success of the non-critical sub-transaction has no effect on the success of its parent. When the local transaction is the book, the book trip sub-transaction is critical, while the 'send acknowledgment one is noncritical. Thus, the failure in transporting a booking acknowledgment has no effect on the whole booking transaction, while the failure in the booking itself does.

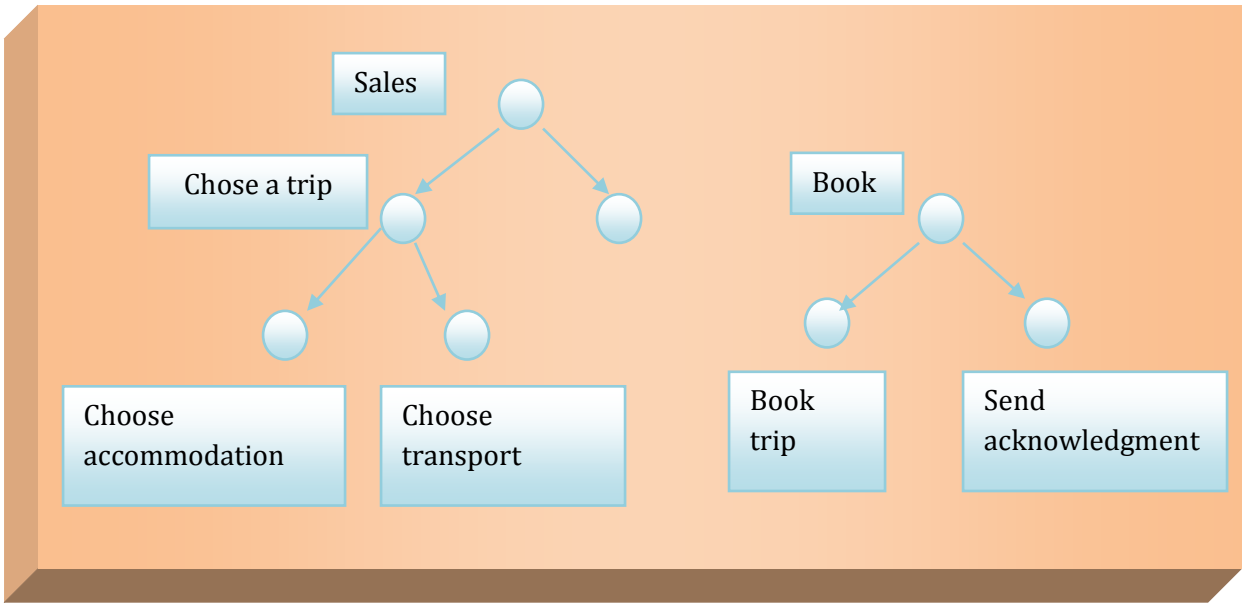

Figure 4.3 Local transactions example [101]

Global Transactions

The upper layer is generated using global transactions and requires relaxed concepts of both atomicity and isolation in order to satisfy workflow process requirements above the transactional level. In the case of global transactions, the rollback technique must apply specified semantics. Generally, local transactions should apply black box stages regarding transactional semantics. Global transactions contain ingrained directed graphs showing local transactions. This graph has a single starting point and symbolises the probable run orders of the workflow process stages. Figure 4.4 shows as example of global transactions. This figure illustrates the execution graph of a certain specification graph where the cancel is considered as the local transaction which has not been executed. On the other hand, the invoice payment iteration has been executed for two times.

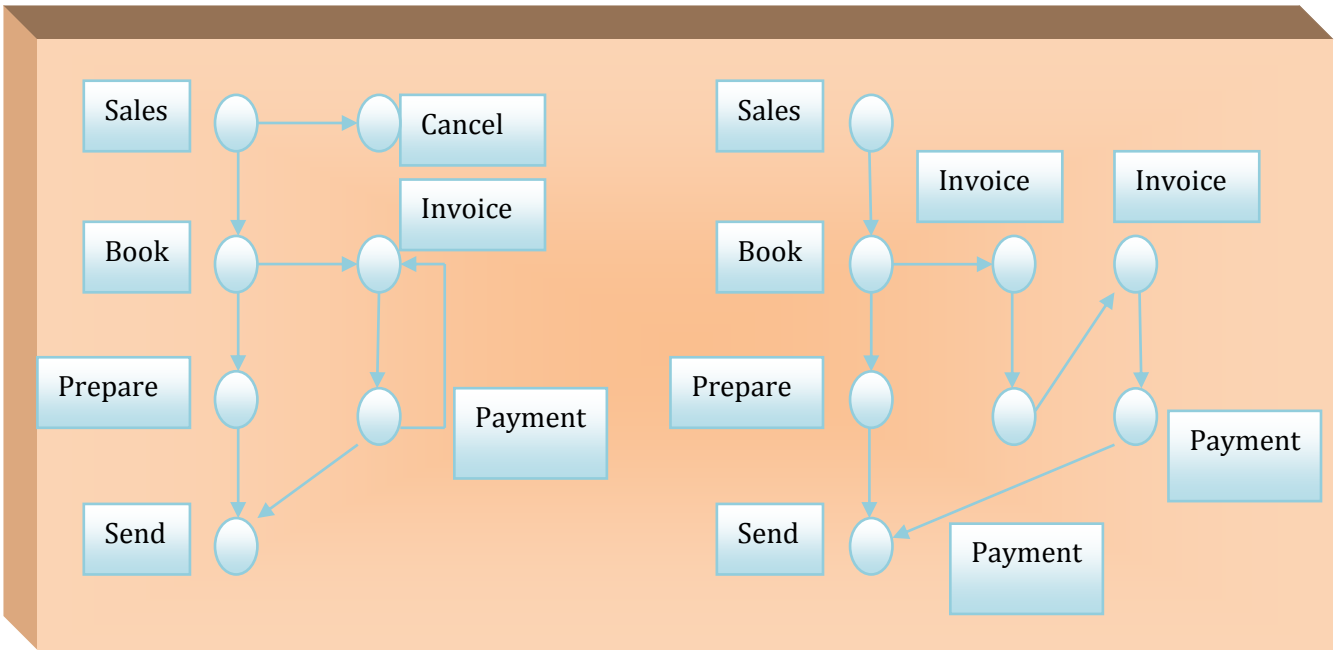

Figure 4.4 Global specification and run graphs [101]

4.4.2 Application of Long Running Transactions in ERP Systems

4.4.2.1 Database State's Multiple Versions

Multiple versions techniques have been used in this paper to assure isolation of different transactions of a long duration when executed at the same time. All transactions of a long duration have their own local space for data, in which the inside data is not visible to other transactions. In the rehearsal phase of a transaction of long duration, step *ti* would like to change the database's data; thus, it duplicates the data to its own space for local data from the database and modifies the data within the space for local data, whereas the actual database data is left unchanged. The execution of operations does not in fact take place until the performance phase for the database.

The actual database's state is a "global database state". In the rehearsal phase, every transaction of a long duration can only access its own database state's version, not the

state of the global database. The database's state version is seen in transaction with a long duration, and is combined version with the state of the global database and the local data space's state of transaction [102, 103].

Function LDi(l)is used to signify the local data space's state at the start of step ti in a transaction with a long duration; l. LDi(l) consists of the data states which have been modified from t1 to ti − 1. LD1(l) = ∅. Function Si(l) is used to refer to the state of the global database at the start ofstep ti, and the Vi(l) function indicates the database state version seen by l, at the start ofstep ti . The Vi(l) Function is created as follows:

$$Vi(l) = iLD\ (l)\ over - ride\ Si(l)$$

The override of the operator indicates that if in the initial operand, an object of the data exists, the retrieval of its value would take place from the initial operand; otherwise, the retrieval would take place from 2nd operand.

Subsequent to the ti function execution, the local data space's state transforms to LDi + 1(l) from LDi(l), however the state of global database will not change:

$$Fi\ (Vi\ (l)) = LDi + 1\ (l)\ over - ride\ Si\ (l)$$

In between the period of the start of ti+1 and finish of ti, the state of global database may change to Si + 1(l) from Si(l)due to the other transactions. Consequently, at the start of ti+1, the state of database which is visible to l (i.e.Vi + 1(l)), is synthesised once again from Si+1(l) and LDi + 1(l).

The current view is that four significant properties can be exhibited by an agent:

- Autonomy, which permits software agents to function exclusive of direct human intervention; this means they have a level of control and power over their internal state and actions.

- Social ability, which permits software agents to communicate and interact with other software agents (and with humans possibly) by means of different language of agent communication.
- Reactivity, which permits software agents to recognise their surroundings (e.g. the user, physical world through a GUI, another agents' collection, or the Internet environment) and respond in a timely manner to modifications that have taken place
- Pro-activeness, which permits software agents to demonstrate behaviour directed towards goal by different initiatives, not only by responding to their environment.

4.4.2.2 Control Mechanism of PP/T Concurrency

For the ti step, in a transaction of long duration, to be executed successfully on state S of the database, pi must be correct. As a result, for a transaction of long duration to be successfully placed, the state of the database in the performance stage should assure every step's predicate. However, the mechanism to transform/predicate is not certain.

To cover the drawback of the mechanism of transform/predicate, it is useful to examine the business processes involved in the banking sector. Suppose a customer requests the bank purchases a bond worth of $1000 for him. The request will be forwarded by the bank to a stockbroker, however the customer will not be informed of the result of the deal until the following day due to the bank's fear that the customer may withdraw the money from her/his account during the request period. Thus, the bank will freeze the customer's account to the requisite level to cover the bond; thus, although, the account balance is X, the balance that can be drawn by the customer is X − 1000 (where the value of X is greater than 1000). This type of check on the object of the data is known as a "security lock" at the semantic level. It does not forbid other transactions' access to the object of the data, but it demands that if another transaction alters the data object's

value, the modified value has to meet the established constraints. Therefore, it is certain that the LRBP would be completed successfully. Our control mechanism of PP/T concurrency is supported by this practice of system data [102].

In the mechanism of PP/T, a table of constraint is developed for a database that consists of all the controls that should be satisfied by the consistent state of a database. These controls are similar to "semantic locks" on the objects of the data. As a transaction of long duration ends each step in the rehearsal phase, it places the pre-condition of the stage in the table of constraint. Then, to prepare a transaction to be committed, the system first verifies whether the state of new database assures every constraint in the table of constraint. If the constraints are not satisfied, the transaction will not continue.

As a result, in the phase of the performance of a transaction of long duration, the pre-conditions for all the steps are met with satisfaction, thus transaction execution with long duration can be completed successfully. We consider the original mechanism of transform/predicate to be an optimistic mechanism of the transform/predicate (as it does not apply any restraints to the state of database), and considers the new mechanism of the transform/predicate which is based on the constraint of a pessimistic mechanism of transform/predicate.

In the rehearsal phase for a transaction with long duration, for the ti step to be executed successfully, the pi phase has to be fulfilled on the stat eof database of the own version of the transaction; that is,pi (Vi (l)) $=$ true. Because Vi (l) is created from Si(l) and LDi(l), we can split pi in 2 components: in one component, the data in LDi(l) is involved, which is symbolised as PLDi; the in the other component the data is not included in $LDi(l)$, and is indicated as psi. The following conclusion can be reached:

$$Pi\ (Vi\ (l))\ \Rightarrow\ psi\ (Si\ (l)).$$

Thus, psi can be added into the table of database constraints, in order that the following transactions will not violate the constraints. However, we cannot state that

piLD (Si(l)) = true frompiLD (Vi(l)) = true, therefore we have to consider$pLDi$, in particular. In [101], the researchers suggested a scheme model for a weak state function of wpwhich is useful in solving this problem. The wp function can be defined as following:

The $wp\ (t,p)$function is the weak state that must be held in the database prior to the implementation of stage t, in order that p predicate is correct, subsequent to stage t execution, that is:

$$wp\ (t,p)\ (S)\ \Rightarrow\ p\ (t\ (S))$$

As per the 3rd definition, the conclusion below can be reached:

$$wp\ (t1 \circ \ldots \circ ti-1, piLD)\ (S1\ (l))\ \Rightarrow\ piLD\ (Vi(l)).$$

The transform/predicate $wp(t1 \circ \ldots \circ ti-1,\ piLD)$is a very weak state that must be maintained as $S1(l)$to ensure $piLD\ (Vi(l))\ =\ true$. Thus, we can initially test whether the transform/predicate $wp(t1 \circ \ldots \circ ti-1,\ piLD)$ is true based on the present state of the global database $Si(l)$. If it is not right, the ti step cannot be executed successfully, and the system will return an error message. If the transform/predicate is true, then the database system would place the transform/predicate into the table of constraints, in order that the constraints will not be violated in later transactions. Unfortunately, the computing process $wp\ (t1 \circ \ldots \circ ti-1,\ piLD)$ from $piLD$ can be extremely complex (often impossible). As a result, we can alter the mechanism of PP/T so it is weak, to record only psi in the table of constraint, and to not deal withpiLD. In this type of case, the mechanism of PP/T still represents progress on the original mechanism of the transform/predicate, although it cannot guarantee the winning execution of all transactions with a long duration in the final stage. figure 4.16 illustrates the relations among the involved parts in the model [104].

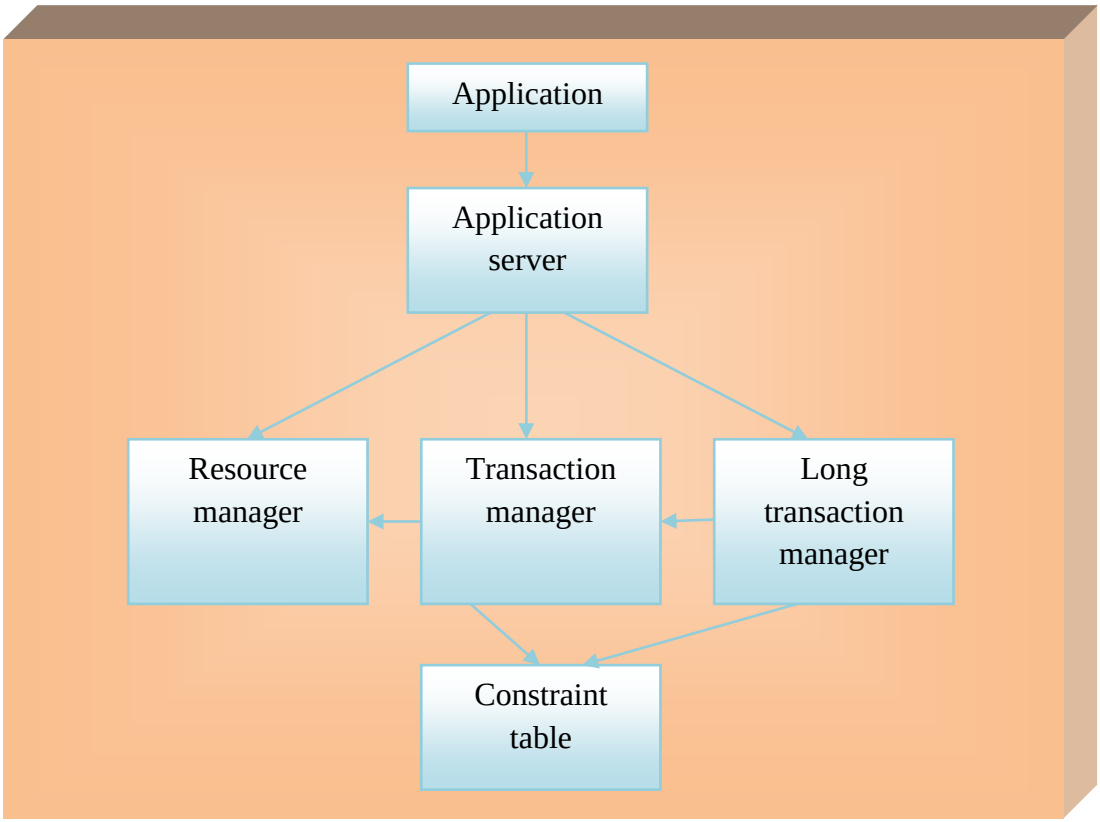

Figure 4.5 Relations among the involved parts in the model

4.4.2.3 Composition of Long Running Transactions

One of the important decisions is determining if it is allowed for transactions to be constructed from other long running transactions or not. As an example, the payment action can be considered as a single transaction with a related compensation and which is invoked when a failure happens and the payment that must be revoked. But, this type of transactions may engage many account transfers that occur behind the scenes where each one may happen with its own compensation when a failure happens before terminating the payment transaction. It was noted that there are not adequate formalisms that permits the ACID transactions to be part of the long running ones. On the other hand, various flow composition languages, as the ones proposed in [49; 105; 106] supposed that the atomic actions are the building blocks for long running transactions

where they can be used in representing ACID transactions. Furthermore, there are many published works that support the nested long running transactions and some other works as the one proposed in [107] does not support the nested long running transactions where it considers them as parallel transactions. Many design decisions focused in the relation among a certain transaction and its parent concerning the access of sub-transaction accumulated compensation to its parent after completing the sub-transaction and state of parent transaction when one of its children fails.

4.5 SUMMARY

Long Running Transactions (LRTs) composed of certain tasks which can be in sequence and in parallel. Those transactions are not atomic, where they depend on applying a compensation mechanism when a certain execution is not finished. Previous researches demonstrated that the most influential model of long duration transactions is the saga which depends on the idea of decomposing each transaction into various short duration transactions. The main benefit of using those transactions is the recovery after failures. Those transactions handle the failure propagation is systems and return them for the original point before the occurrence of failure. In fact, the main decision of those transactions is determining if they can be constructed from other long running transactions or not

The application of workflow management depends on using improved transactional control which can be offered by traditional database systems. Although several methods have been developed, no one of them fulfils all the necessary requirements because the workflow management demands various transactional semantics at several process levels. The general used model for business workflow process management depends on using multi-level process, where it can be mainly decomposed into two layer transaction models; local and global transactions. The main tasks that must be performed by a software agent are: the use of major amounts of domain knowledge, error tolerance,

unexpected or/and incorrect input, the use of both abstractions and symbols, exploring goal oriented performance, learning capability from the environment/surroundings, actual operations and communication through natural language.

Communication Closed Layers (CCLs) are mainly used to decompose the distributed development processes of certain distributed systems into various layers to facilitate the analysis of them using sequential programs techniques. Those layers require defining their analysis and design methodology. A system workflow can be analysed based on involving the spectrum of both validation and verification that ranges from simulation validation to the checking of various known properties, such as security

CHAPTER 5

MODIFICATIONS FOR COMMUNICATION CLOSED LAYERS

This chapter introduces some possible modifications to CCLs. It describes some of the processes used for choices and iterations in the local environment; analyses the main model of CCLs using the programming language Occam, and explains the error recovery technique in these models.

5.1 INTRODUCTION

Mainly, the efficient use of distributed systems depends on the concurrency of tasks which excludes the sequential operation of these tasks. In other words, tasks cannot operate at a node before completing all the previous ones through the system. Thus, there is a need for ensuring the noninterference between the system concurrent tasks. This can be achieved using the layering technique that offers an abstraction of advanced communication. The most used layering technique is the decomposition of a system into Communication Closed Layers (CCLs) which are considered as improved techniques that can be mainly used in the serialization of various systems and programs.

Mainly, the decomposition of a specific distributed system or program workflow into communication closed layers is considered a superstructure of its main decomposition into various communicating stages. The main benefit of using the communication closed layers is simplifying the analysis of various distributed systems and programs to implement several program layers that do not interfere with each other's. This chapter introduces some modifications for the presented Communication Close Layers (CCLs).

5.2 WORKFLOW DECOMPOSITION TECHNIQUES

Various steps are required for the decomposition of a certain system workflow to cope with several structures, such as iterations and choices.

Based on using a general choice form, and a non-deterministic case as a prioritised choice, and excluding any declaration, a process that solves choices can be initiated. On the other hand, it is assumed that the guards used are non communicated Boolean variables based on considering the following P as a guarded choice workflow, where its sub-activities include communication activities and workflows:

$$Choices: S$$

$$G_1 \rightarrow R$$
$$\square$$
$$G_2 \rightarrow Q$$

Where: R and Q are two activities and G_i represents choices.

The proposed work (P) can be converted into an equivalent workflow that contains a CCL construction as follows, where Ṕ represents a safe decomposition:

$$\acute{P} \triangleq \left\{ \begin{array}{ll} \left[\begin{array}{l} G_1 \rightarrow R; \quad GFlag := false \\ \square \\ G_2 \rightarrow \quad GFlag := true \end{array} \right] ; & (P_{11}) \\ \left[\begin{array}{l} GFlag \rightarrow Q \\ \square \\ \text{not } GFlag \rightarrow skip \end{array} \right] & (P_{12}) \end{array} \right\}$$

The presented workflow includes for cases:

- First case: When both G_1 and G_2 are false, the first part P_1 will end as detailed by the semantics;
- Second case: When G_1 is true, while G_2 is false, R will be operated and $GFlag$ will be set as false. In addition, the second part P_2 will operate to skip;
- Third case: When G_1 is false, while G_2 is true, R will not operate and $GFlag$ will be set as true. In addition, the second part P_2 will operate Q as involved; and
- Fourth case: When both G_1 and G_2 are true, the first part P_1 will determine if R will operate or not and $GFlag$ will be set to true or false. In addition, the second part P_2 will only operate Q when $GFlag$ is true.

If there is a local environment workflow (E), then $P \parallel E$ can be decomposed as follows:

$$((P_1 \parallel E_1) \parallel (P_2 \parallel E_2) \parallel (P_3 \parallel E_3))$$

Each P_{ij} can be a communication or non communicated closed layer

If there is another local environment workflow (E) with the same structure as P, where both of them have three parts, then $P \parallel E$ can be decomposed as follows:

$$((P_{11} \parallel E_{11}); (P_{12} \parallel E_{12}) \square (P_{21} \parallel E_{21}); (P_{22} \parallel E_{22}) \square (P_{31} \parallel E_{31}); (P_{33} \parallel E_{33}))$$

The proposed equation can be rewritten as the structures shown in the following which show the possible combinations of the above equation. Here these structures show that the layers can be composed as alternatives or sequentially. Additionally, iteration, interrupt and conditional structures can also be employed.

$$(P_{11} \parallel E_{11}) \square (P_{21} \parallel E_{21})$$

$$(P_{11} \parallel E_{11}) \square (P_{22} \parallel E_{22})$$

$$(P_{11} \parallel E_{11}) \square (P_{31} \parallel E_{31})$$

$$(P_{11} \parallel E_{11}) \square (P_{33} \parallel E_{33})$$

$$(P_{12} \parallel E_{12}) \square (P_{21} \parallel E_{21})$$

$$(P_{12} \parallel E_{12}) \square (P_{22} \parallel E_{22})$$

$$(P_{12} \parallel E_{12}) \square (P_{31} \parallel E_{31})$$

$$(P_{12} \parallel E_{12}) \square (P_{33} \parallel E_{33})$$

$$(P_{21} \parallel E_{21}) \square (P_{31} \parallel E_{31})$$

$$(P_{21} \parallel E_{21}) \square (P_{33} \parallel E_{33})$$

$$(P_{22} \parallel E_{22}) \ \square \ (P_{31} \parallel E_{31})$$

$$(P_{22} \parallel E_{22}) \ \square \ (P_{33} \parallel E_{33})$$

The presented expressions explore the layers that can be decomposed.

The processes for iterations depend on assuming that loops are finite and the communication symmetry is guaranteed. The use of finite loops allows a replacement as a group of composed statements, while ensuring communication symmetry means that each one of the layers is a CCL. Thus:

$$While\ G\ do\ \{R\} \equiv G\ \rightarrow R;$$

Then, with:

$$while\ G\ do\{R\} \parallel Q$$

It can be converted to an equivalent workflow system as follows:

$$\big((G\ \rightarrow R \parallel Q); (while\ G\ do\{R\})\big)$$

Thus, $(G \rightarrow R \parallel Q)$ can be layered into a CCL based on the structure of both R and Q.This process must be repeated with several iterations such as while G do{R}.

The theory includes that: For any workflow system (P), there is an equivalent quasi sequential system. This theory can be proofed based on following three main stages:

$$P \parallel E \mathrel{\widehat{=}} \big((P_1 \parallel E_1) \parallel (P_2 \parallel E_2) \parallel (P_3 \parallel E_3)\big)$$

- First stage: Making sure that all the $P_i E_i$ are of the same length.
- Second stage: Locating the related communication activities and then constructing a communication layer. Other activities can be used to construct non communicated layers.
- Third stage: Resolving the communication.

5.3 CCLs Model

This section proposes a modified model for CCLs that can be used in the detection, identification and recovery of errors. In addition, the evaluation of damage in the system resulting from faults introduced by the delay in faulty system performance and error detection causes the spread of damage throughout the system. The proposed model depends on the Occam programming language that has an equivalent quasi sequential approach. This is performed based on constructing a model with a set of atomic action CCLs that work in a sequential way. All the processes in the constructed model using Occam occur concurrently, which means that the whole process ends when the interior processes end [99,100].

5.3.1 Processes of Occam Programming Language

The Occam is a concurrent programming language that depends on two main concepts; processes (PROC) and channels (CHAN). Processes are considered to be the main block in any Occam structure, affecting where they begin, operate and then terminate. These processes communicate with each other by channels as shown in figure 5.1:

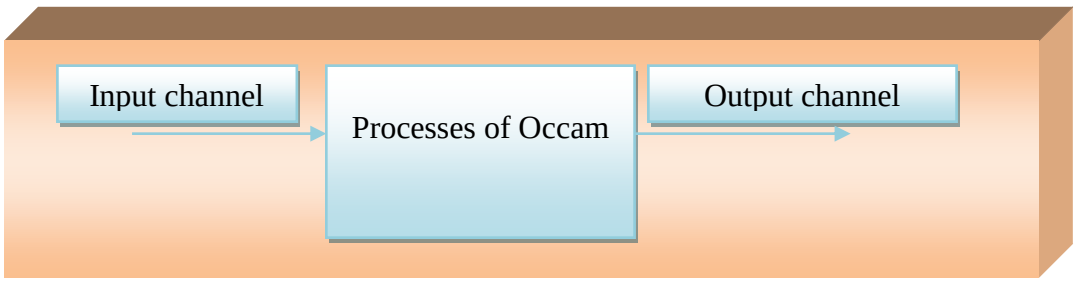

Figure 5.1 Occam structure model [99]

The Occam includes four types of processes; assignment (:=), output (!), input (?) and null (SKIP) processes. There are three other processes can be combined with these four types; sequential, concurrent and choices processes as follows:

Suppose there is a set of workflow choices processes, with conjugated environmental processes $(P \parallel E)$; the sequential process is:

$$SEQ$$
$$P_1 \parallel E_1$$
$$P_2 \parallel E_2$$

In the sequential process, the operation of the $P_1 \parallel E_1$ is followed by the operation of the process $P_2 \parallel E_2$. Thus, the whole process ends when the second process ends.

For the concurrent process:

$$PAR$$
$$P_1 \parallel E_1$$
$$P_2 \parallel E_2$$

In this process, both $P_1 \parallel E_1$ and $P_2 \parallel E_2$ operate concurrently. Thus, the whole process ends when both of them end.

For the choices process, there are two types; deterministic and non-deterministic as follows:

$$Deterministic:$$
$$If$$
$$G_1$$
$$P_1 \parallel E_1$$
$$G_2$$
$$P_2 \parallel E_2$$
$$\dots.$$
$$G_n$$
$$P_n \parallel E_n$$

For the deterministic type, all the presented processes $P_i \parallel E_i$ execute when their related guard process, which is a Boolean expression is acceptable. After this, the process ends. More clearly, when all guard processes are acceptable, the whole process ends.

$$Non-Deterministic:$$
$$ALT$$
$$G_1$$
$$P_1 \parallel E_1$$
$$G_2$$
$$P_2 \parallel E_2$$
$$\ldots.$$
$$G_n$$
$$P_n \parallel E_n$$

In the non-deterministic type, the input process can be a guard process, with Boolean expression only or a Boolean expression with a null process. In this type, when there is more than one acceptable guard process, related processes cannot be defined.

5.3.2 Processes of CCLs Model

In this work, the CCL model constructed consists of sequential processes $P_1 \parallel E_1$, $P_2 \parallel E_2$,……, $P_N \parallel E_N$. These processes communicate with each other to offer various services. Each one of these processes in turn consists of a set of segments as follows:

$$For\ P_i \parallel E_i:$$
$$S_1^i$$
$$S_2^i$$
$$\ldots\ldots$$
$$S_k^i$$

Since the used Occam program is a concurrent one, it takes the following form:

P ::

PAR

$P_1 \parallel E_1$

$P_2 \parallel E_2$

… ….

$P_N \parallel E_N$

For the following parallel programs:

PAR

$For\ P_i \parallel E_i$:

$S_j^{,1}$

$S_j^{,2}$

… ….

$S_j^{,N}$

Where: each $S_j^{,i}$ is a program segment (S_j^i) or a null process (SKIP)

Generally, a Communication Closed Layer (CCL) can be defined as a layer that excludes an input/output command, as its related input/output command is in another layer. This layer can be symbolised by the application of a time diagram, as a cut that divides the time diagram into two parts, where each part is called a layer. Figure 5.2 shows an application with three processes, where the horizontal rows represent time, while the vertical ones represent the communication between the sender process and a receiver process [99, 100].

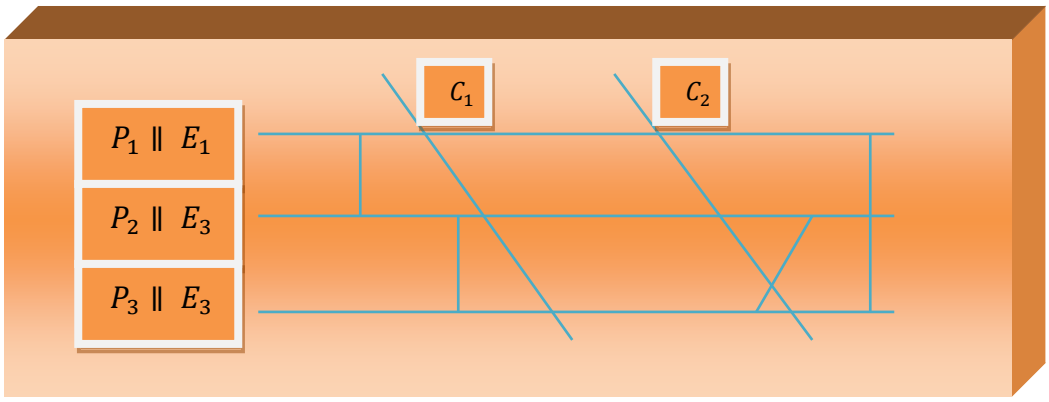

Figure 5.2 An application with three processes

As proposed, the communication layer has no arrows; beginning in one part and ending in a second one. Thus, C_1 is considered a communication closed layer, while C_2 is not.

When a new layered process $(\acute{P} \parallel \acute{E})$ is added for a proposed distributed application, the proposed program will involve a sequential decomposition, which is considered safe when its layers are communication closed ones:

$$\acute{P} \parallel \acute{E} ::$$

$$SEQ\ i = [1\ FOR\ k]$$

$$L_i$$

The main point here is proving that both the proposed distributed decomposition applications and the sequential decomposition programs are semantically equivalent. These two programs are semantically equivalent only when the sequential decomposition is safe. In reality, this is a challenge.

A safe decomposition can easily be constructed using the SKIP process, as in the following example:

For a distributed program (D) with three processes; $D_1 \parallel E_1$, $D_2 \parallel E_2$ and $D_3 \parallel E_3$ and three Occam channels; C_1, C_2 and C_3

D ::

PAR
$D_1 \parallel E_1$
$D_2 \parallel E_2$
$D_3 \parallel E_3$

Each one of these processes is sequential, thus:

$D_1 \parallel E_1$ :::
SEQ
$\{ D_1 \parallel E_1 : non\ input/output\ segment \}$
.
.
$C_1 ! O_1$ ← Output of channel 1
.
.
$\{ D_1 \parallel E_1 : non\ input/output\ segment \}$
.
.
$C_3 ? I$ ← Input of channel 3

$D_2 \parallel E_2$ :::
SEQ
$C_1 ? X$ ← Input of channel 1
.
.
$\{ D_2 \parallel E_2 : non\ input/output\ segment \}$
.
.
$C_2 ! O_2$ ← Output of channel 2

$D_3 \parallel E_3$ :::
SEQ
.
.
$\{D_3 \parallel E_3 : non\ input/output\ segment\}$
.
.
$C_2?Z$ ← Input of channel 2
.
.
.
$\{D_3 \parallel E_3 : non\ input/output\ segment\}$
$C_3!O_3$ ← Output of channel 3

Based on the distributed program proposed with its processes, six CCLs can be then easily built:

L_1 ::
PAR
$\{D_1 \parallel E_1 : non\ input/output\ segment\}$
$SKIP$
$\{D_3 \parallel E_3 : non\ input/output\ segment\}$

L_2 ::
PAR
$C_1!O_1$
$C_1?X$
$SKIP$

L_3 ::
PAR
$\{D_1 \parallel E_1 : non\ input/output\ segment\}$
$\{D_2 \parallel E_2 : non\ input/output\ segment\}$
$SKIP$

L_4 ::

PAR
$SKIP$
$C_2!\, O_2$
$C_2?\, Z$

L_5 ::

PAR
$SKIP$
$SKIP$
$\{\, D_3 \parallel E_3 : non\ input/output\ segment\}$

L_6 ::

PAR
$C_3?\, I$
$SKIP$
$C_3!\, O_3$

Since all the proposed layers are CCLs, the proposed distributed program (D) can be decomposed as follows:

D ::
L_1
L_2
L_3
L_4
L_5
L_6

As shown, it is a safe sequential decomposition.

5.3.3 Nested Communication Closed Layers

Atomic actions are essential programming concepts since they can be used in the generalisation of the error recovery concept in the Occam concurrent programming language. An atomic action can be defined as an instantaneous and indivisible program segment that has a complex interior processing structure. These atomic actions are mainly called transactions in database systems. The proposed model for CCLs in the previous section offers a frame for constructing, implementing and designing a fault tolerant system. As demonstrated previously, the structure of CCLs can generate a sequential decomposition whereby they can decompose any known distributed system into various atomic actions. In other words, a resultant decomposed sequential system is known as an atomic system.

Generally, a layer that has communication closeness features can be used to support the error confinement idea, since it can restrict the propagation of errors resulting from inter-process communication. Layers in a sequential (atomic) system can be nested to any degree to offer communication closeness features, to guarantee the safety of the decomposition process. The following example explores this nesting process.

For the following layer with three sequential processes; $P_1 \parallel E_1$, $P_2 \parallel E_2$ and $P_3 \parallel E_3$ and their matching processes; $\overline{P_1 \parallel E_1}$, $\overline{P_2 \parallel E_2}$ and $\overline{P_3 \parallel E_3}$

$$
\begin{aligned}
&L :: \\
&PAR \\
&SEQ \\
&P_1 \parallel E_1 \\
&\{P_1 \parallel E_1 : input/output\ process\} \\
&P_2 \parallel E_2 \\
&\{P_2 \parallel E_2 : input/output\ process\} \\
&P_3 \parallel E_3 \\
&\{P_3 \parallel E_3 : input/output\ process\}
\end{aligned}
$$

$$SEQ$$
$$\overline{P_1 \parallel E_1}$$
$$\{\overline{P_1 \parallel E_1} : input/output\ process\}$$
$$\overline{P_2 \parallel E_2}$$
$$\{\overline{P_2 \parallel E_2} : input/output\ process\}$$
$$\overline{P_3 \parallel E_3}$$
$$\{\overline{P_3 \parallel E_3} : input/output\ process\}$$

This layer is then reconstructed as three nested layers L_1, L_2 and L_3 as shown in figure 5.3

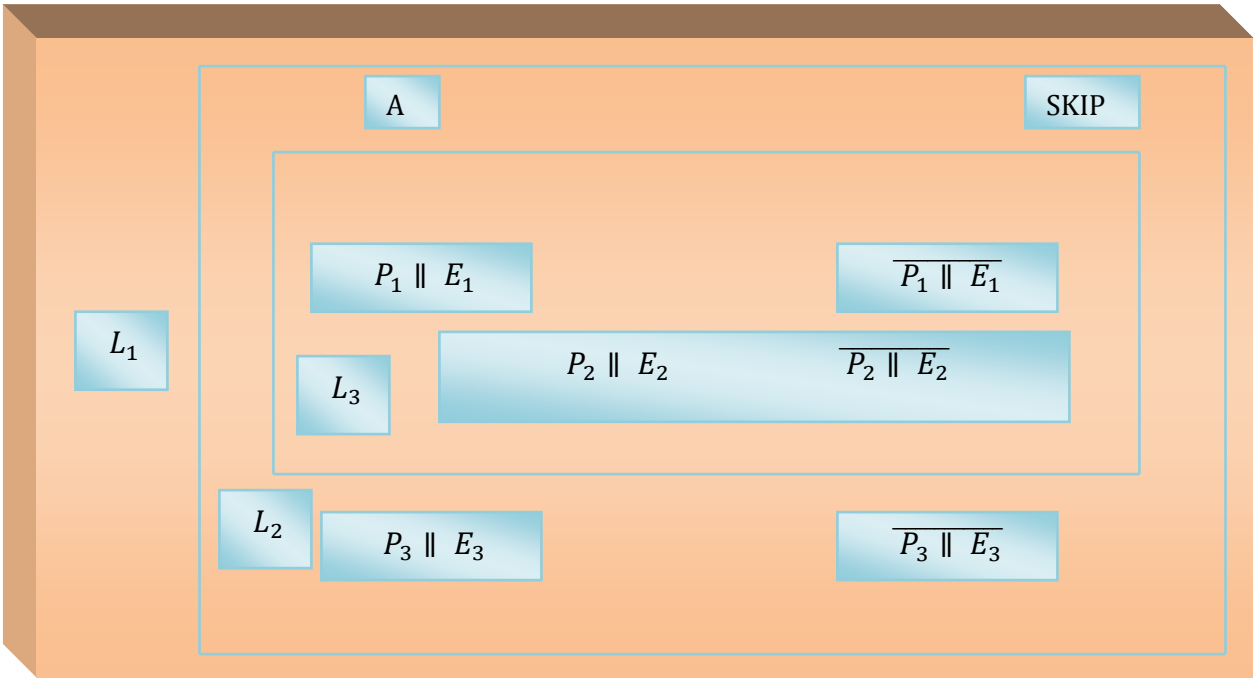

Figure 5.3 Nested layers

Where:

L_1 ::

PAR
SEQ
A
L_2^1
$P_3 \parallel E_3$
$SKIP$
L_2^2
$P_3 \parallel E_3$

L_2 ::

PAR
SEQ
$P_1 \parallel E_1$
L_3^1
SEQ
$P_1 \parallel E_1$
L_3^2

L_3 ::

PAR
$P_2 \parallel E_2$
$P_2 \parallel E_2$

Where L_j^i represents the i^{th} thread of the j^{th} layer.

5.4 Process of Error Recovery in CCLs

The previous section explores the method for structuring a designed application program using Occam into various atomic actions known as transactions. In addition, it shows how these transactions can be nested. In this section, the error recovery process in the CCL structure is illustrated.

Generally, faults result from the incompatible performance of certain processes. Thus, systems must offer and support mechanisms that can convert faults into incompatible performance. This resultant incompatible performance can trigger various actions, such as terminating the related process. This section introduces the addition of an error model to a system model, designed using the Occam; detailing its error recovery process. In the system designed, the proposed activity of the system performance, termination conditions of processes can be classified as follows, based on two approaches; communication and concurrency:

- Communication: Termination conditions of a specific input/output process that is conjugated with an environmental process $P \parallel E$ can be classified as shown in figure 5.4:

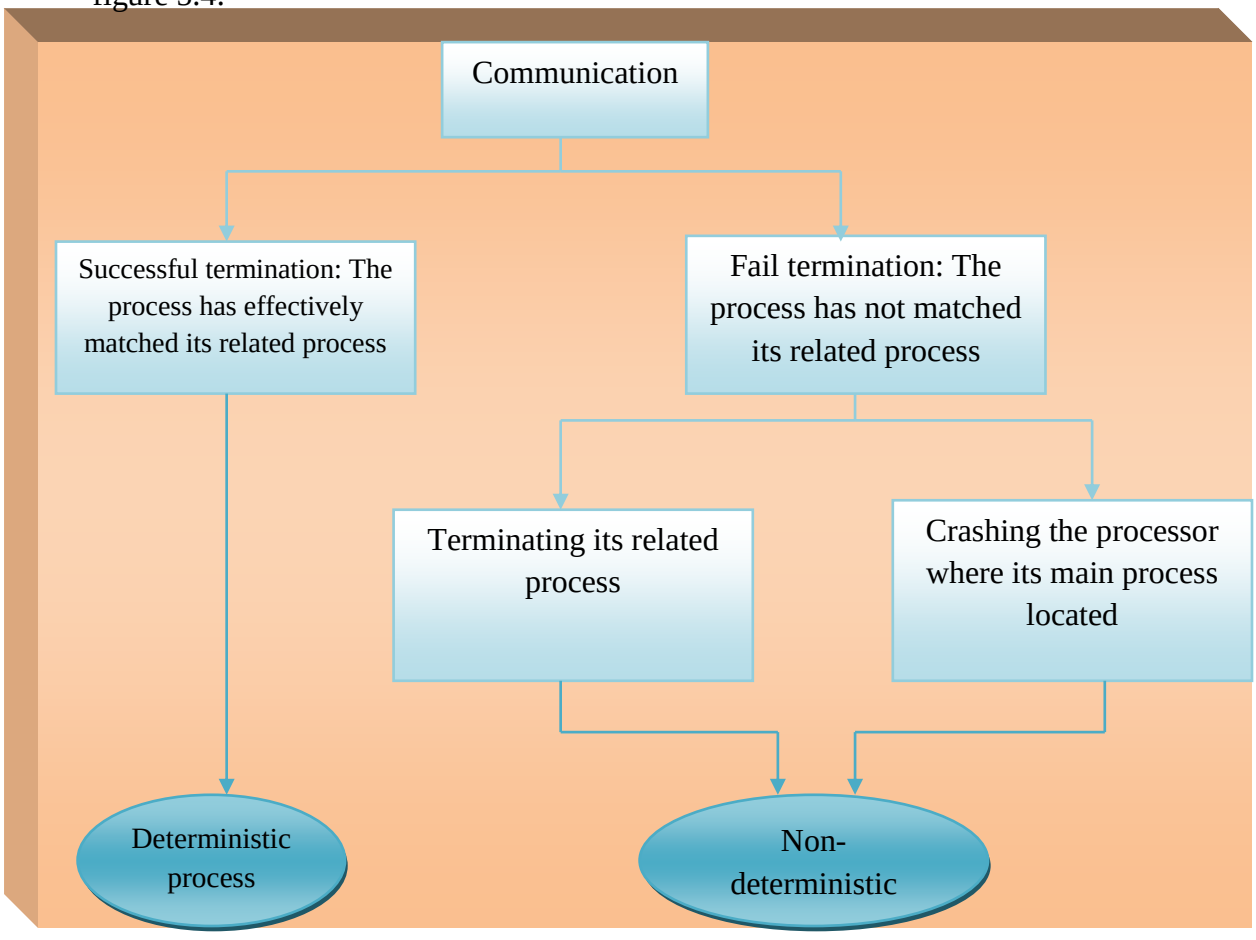

Figure 5.4 Communication process

These two conditions of termination; successful and fail depend on the way that a specific process is chosen for operation. These two conditions are called: deterministic and non-deterministic, respectively.

- Concurrency: Based on the proposed distributed termination principle, CCLs terminate only when their related threads terminate. For non-communicated closed layers, one of the following conditions may happen as shown in figure 5.5

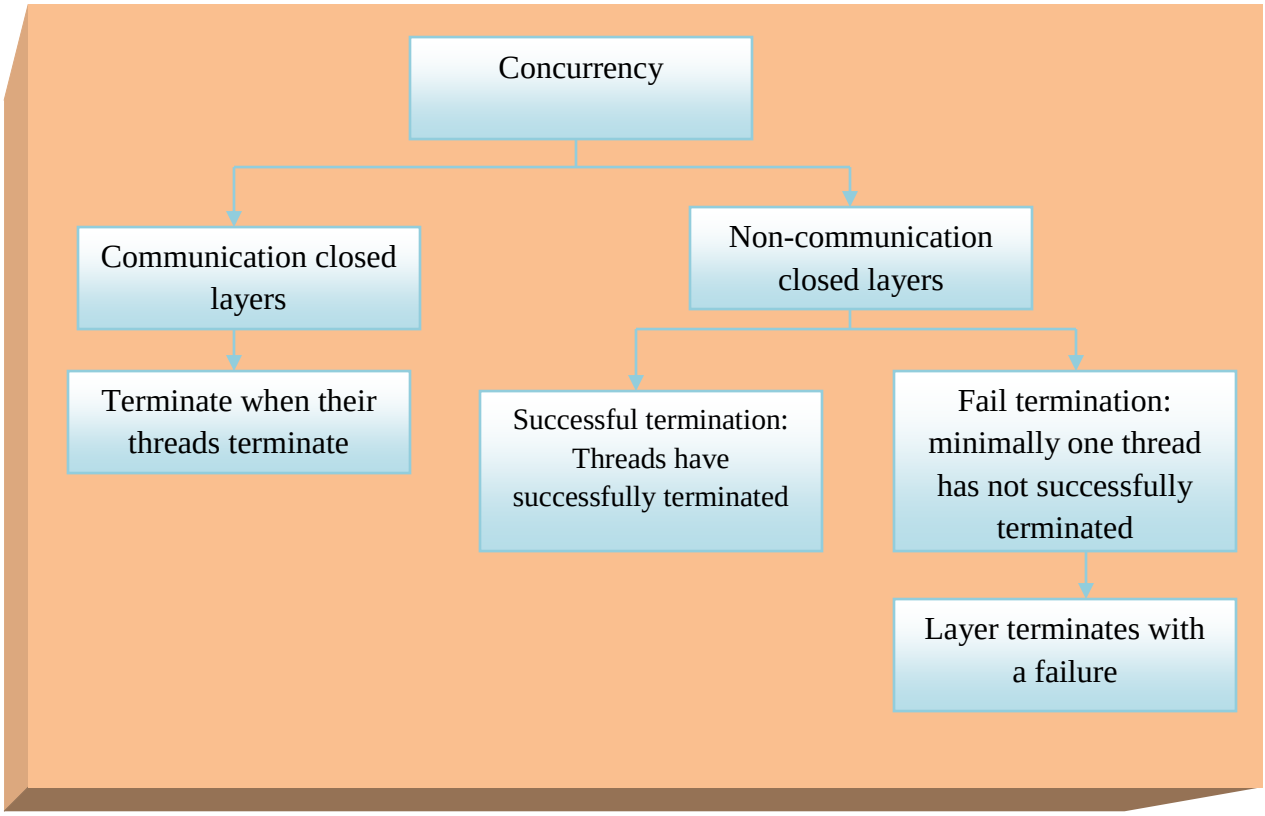

Figure 5.5 Concurrency approach

5.4.1 Rules of Error Recovery Process in Communication Closed Layers

In a successful termination, fault tolerance measures in one CCL cannot be seen by the other layers. Thus, these measures are encapsulated in the sub-layers. On the other hand, communication closed layer specifications depend on the relationship between both the conditions at the beginning and the layer termination. Based on the assumption that a

combined fault with an environmental error condition F||EF has occurred in the CCLL_i, error recovery process is then performed. Typically, each layer has a local handler (H). Thus, an effort can be made to recover the error F||EF using a specific software program that is obtainable for the local handler of the layerL_i; H_i. This process then has two main conditions as shown in figure 5.6

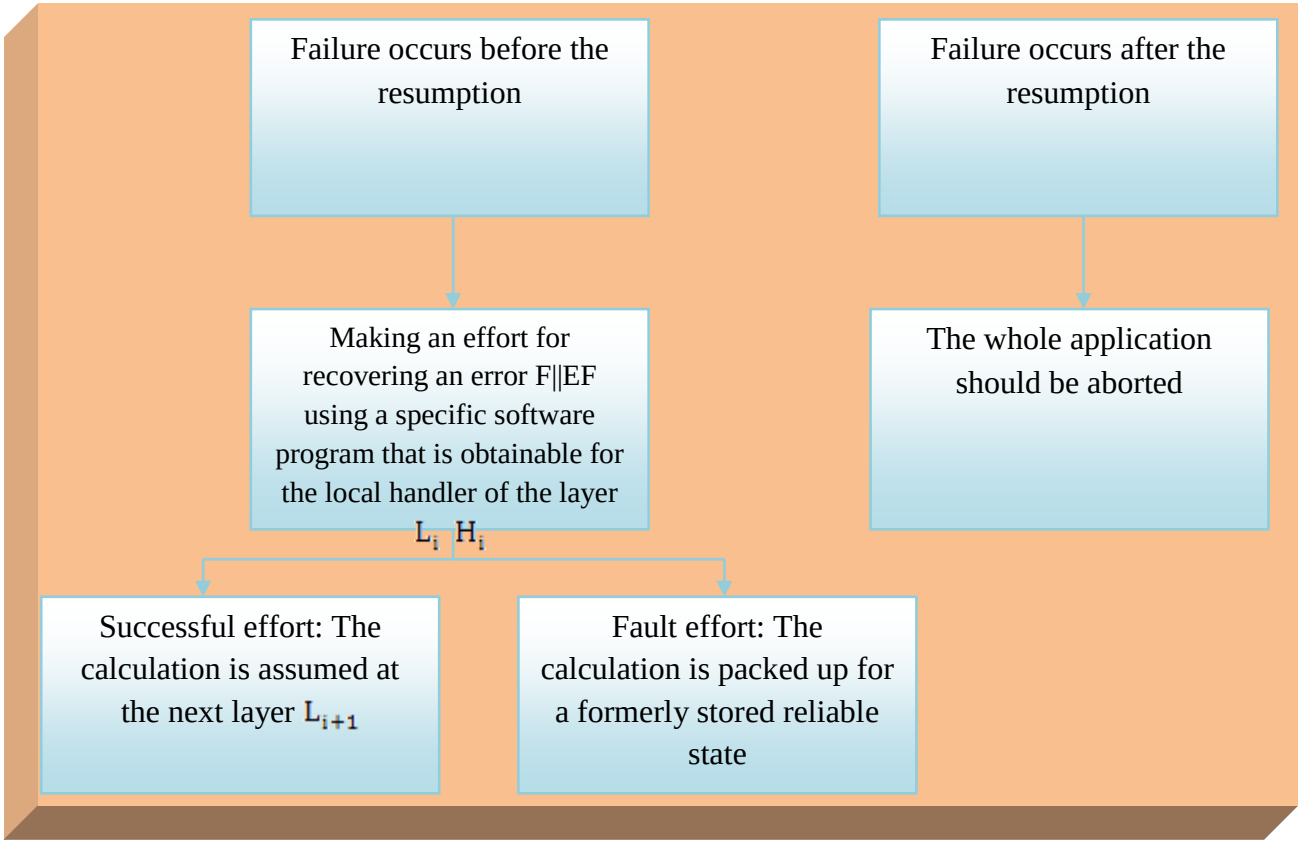

Figure 5.6 Error recovery

Figure 5.7 shows activity without errors. It includes three CCLs where each layer consists of three sub-layers:

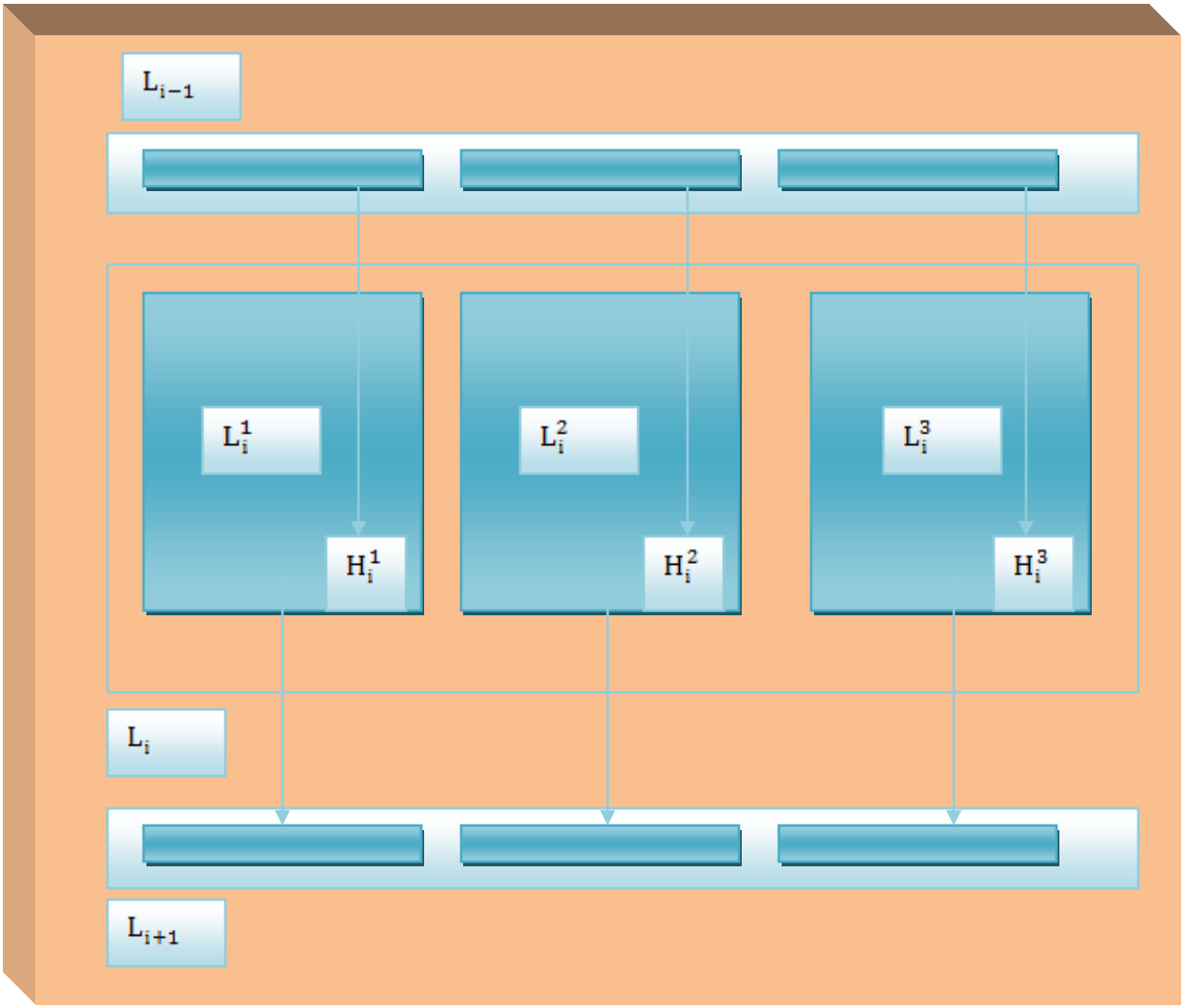

Figure 5.7 Activity without errors

Figure 5.8 shows activity with errors. It includes three CCLs where each layer consists of three sub-layers:

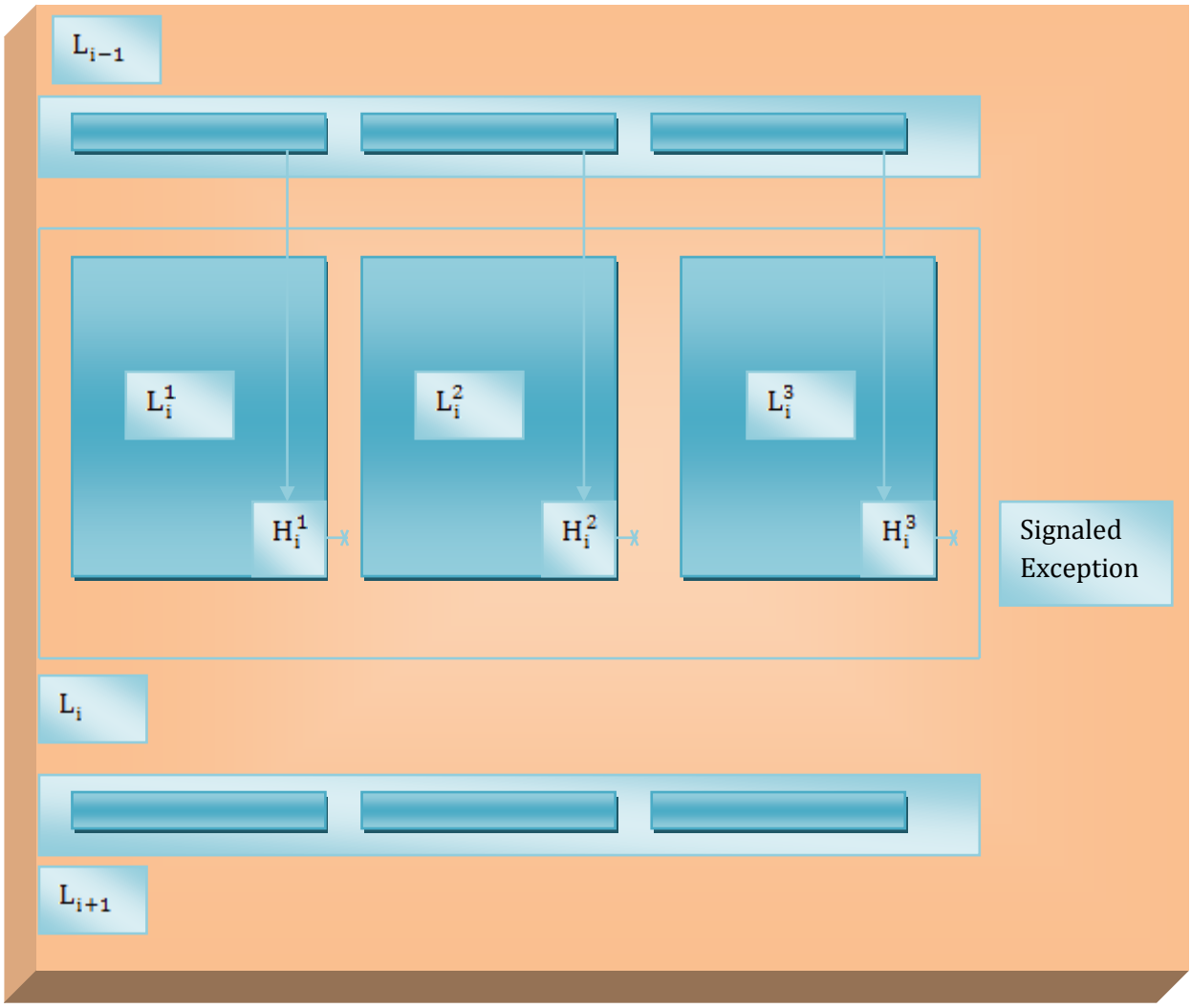

Figure 5.8 Activity with an error

5.4.2 Types of Error Recovery

The design and implementation of fault tolerant systems depend on the recoverability of CCLs. Thus, two types of error recovery models are proposed; forward and backward error recovery models.

First Type: Forward Error Recovery Model

The main goal of this model is the removal or isolation of faults. This depends on the correct identification of faults and the correction of mistaken system states prior to performing more processes. This model uses accurate and full knowledge of the mistaken system state. To manage these errors, various exception handling techniques must be used.

For the proposed layer $L_i$in the previous section that has a handler$H_i$as a sub-layer and some exceptions that cause errors combined with environmental error conditions F||EF, the forward recovery model can be demonstrated as follows: Each one of the resultant errors obtains an Exceptional Value (EV) that succeeds the error F||EF. All the resultant EV values are then declared as exceptions. These exceptions work are shown in figure 5.9

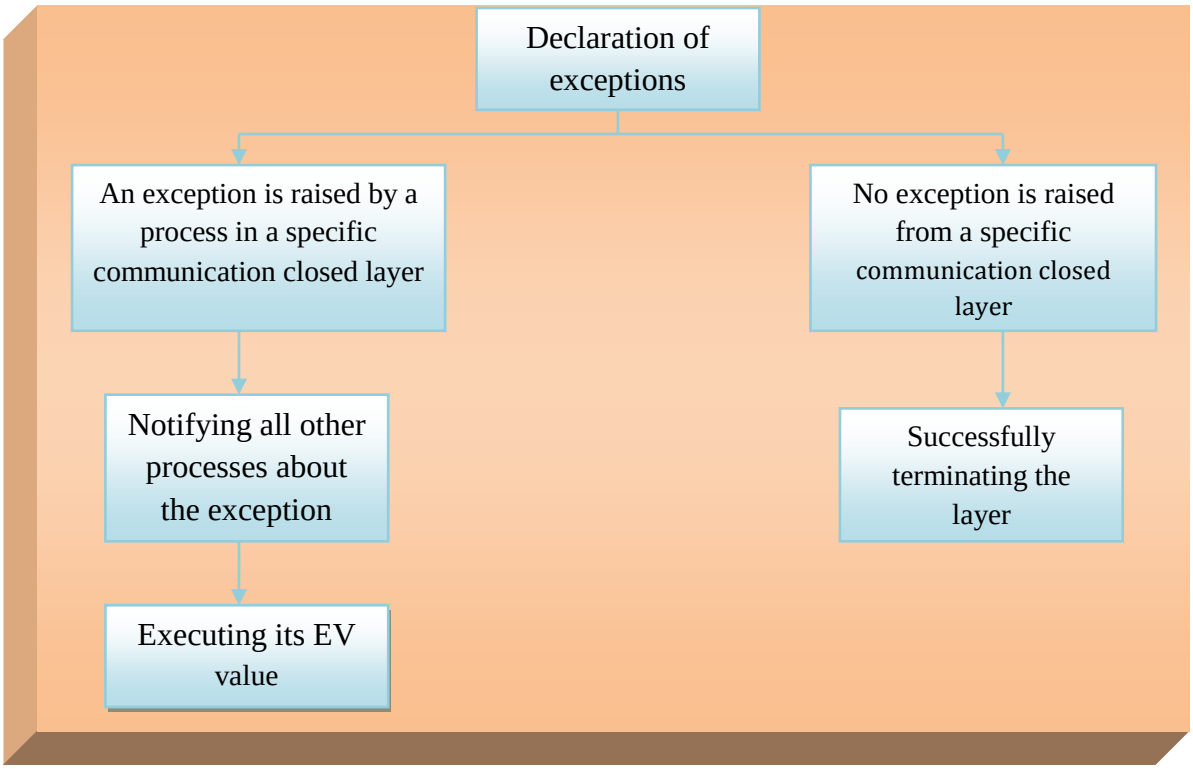

Figure 5.9 Declaration of exceptions in the forward error recovery model

The handler of the communication layer is similar to that for the previously proposed ALT process; but, it is deterministic. Meanwhile, any abnormal layer handler termination can be indicated by a process called signal. This process is included in an Occam process that is called an action process. This structure is explored in the example below, in which figure 5.10 illustrates the proposed structure. It is an algorithm for finding the root of a number with three main CCLs; an input layer where numbers are inserted, a root layer where the root value is found for the input numbersm, and an output layer where the resultant value is demonstrated.

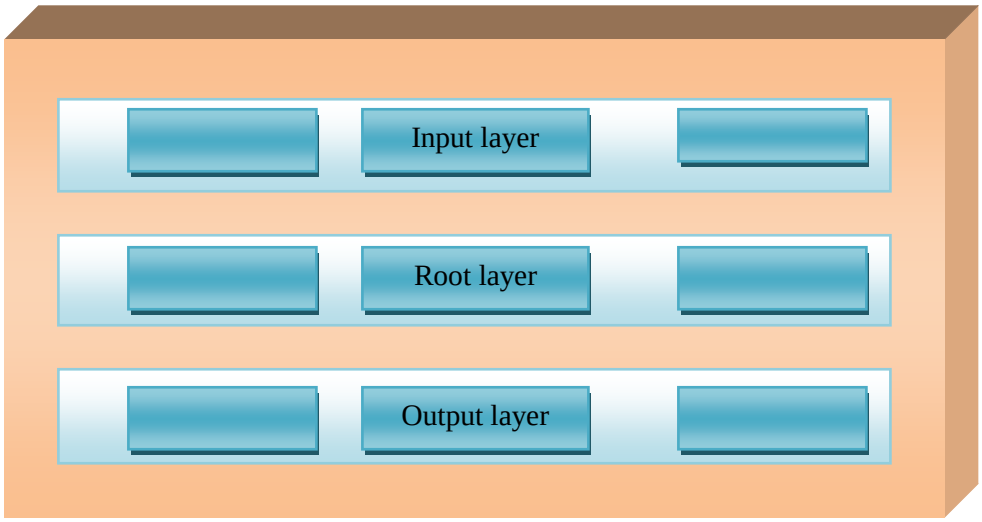

Figure 5.10 Subtraction algorithm

In this example, the first layer is the only one considered, assuming the following:

- Threads are connected by an input channel
- The main error that can happen is the (Negative) input number, where the root cannot be found for negative numbers (when X<0).
- Thus:

```
LAYER Input (Parameter list):
--
-- Declarations
--
EXCEPTIONS Negative:
```

```
−−
−− Other declarations
−−
PAR
SEQ
Input ! X
IF
X < 0
Negative
ALT*
Negative
SIGNAL
```

Where ALT* is the layer handler

As proposed, when an exception is raised, the control goes with the related EV to the ALT* and then operates a suitable action process. After this, the X value is retransmitted for the exception negative. The forward error recovery process ends after the action process is completed. When the number inserted is positive, the layer will successfully terminate and then goes to the second layer; the root layer.

Second Type: Backward Error Recovery Model

This model backs up a process related to a previous state that is expected to generate a fault and then completes the other process. The proposed communication layer structure here occupies a layer static upper boundary, which acts as a natural recovery line for the model. Thus, when a fault happens, a specific mechanism should be constructed to undo the layer and then activate a specific alternative layer. This depends on saving the layer state after commiting it.

Based on considering layer 4 in the first proposed example in this section: [99]

Based on considering layer 4 in the first proposed example in this section: [118]

L_4 ::
PAR
$SKIP$
$C_2?Z$
$C_2!O_2$

The backward error recovery model is:

$LAYER\ FOUR\ (Z, O_2)$
$--$
$--\ Declarations$
$--$
PAR
$SKIP — thread\ 1$
$C_2?Z — thread\ 2$
$C_2!O_2\ \ --thread\ 3$

The first thread is a skip process; the input value is assigned to the variable in the second thread and the output value is assigned to the last thread. This process is carried out only when the PAR structure is successfully terminated. Elsewhere, it is carried out from an incomplete state. Thus, it guarantees that a layer state update is successfully completed when the state transition is incomplete. If it completes, the state will not be influenced. On the other hand, if the PAR fails, as a minimum another thread will also fail.

5.5 SUMMARY

This chapter introduces some modifications for the presented Communication Close Layers (CCLs) based on describing some used processes of choices and iterations in a local environment, analyzing the main model of communication closed layers and explaining the main used error recovery technique. The presented modified model for communication closed layers can be used in the detection, identification and recovery of

errors. Furthermore, the evaluation of damages in the model which resulted from faults is explored. The presented model is constructed using a set of atomic action communication closed layers that work in a sequential way. Mainly, all the processes in the constructed model mean that the whole process ends when the entire interior processes end.

The presented model provides a frame for constructing, implementing and designing a fault tolerant system. Atomic action communication closed layers are also presented where they can be used in the generalization of the error recovery concept. The structure of communication closed layers can produce a sequential decomposition where they can decompose any known distributed system into various atomic actions. In other words, the resultant decomposed sequential system is known as an atomic system.

The main two types of error recovery models that presented in this work are the forward and backward error recovery models. On the other hand, this chapter offers a solution for the main problem that faces the use of communication closed layers in long running transactions which is the complexity of implementing transactions in a graphical notation depending on introducing an advanced graphical notation of certain business processes. The presented solution provides an easy formalism that can be used to describe the transactional characteristics of business processes.

CHAPTER 6

METHODOLOGY AND DESIGN APPROACH

This chapter introduces the main two design challenges in detail and then describes the design of the presented solution and motivation behind the solution based on exploring the communication closed layers in an ERP context. The main activities, processes, schedule and even the generator of the model are explored.

6.1 Introduction

Recently, corporations are structured in extremely complex ways to deal with an increasingly competitive marketplace, with offices, workforce and manufacturing plants spread over multiple countries and even continents. Managing such complex organisations is not an easy task. One of the tools that are considered essential in helping managers and executives in this task is the integrated system which often termed the Enterprise Resource Planning (ERP) system. This system aims to provide an integrated view of all the business processes within an organisation in real-time.

The goal is to examine the possible improvements for the currently applied methodology in the field of ERP systems by investigating the suitability and applicability of using Long Running Transaction (LRT) modelling approach to model the ERP workflows.

Two main challenges are identified in applying long-running transaction model to the ERP context. The first challenge is that some of the concepts used within traditional Online Transaction Processing (OTP) systems, such as unrolling and check-pointing cannot be readily applied to ERP contexts, unlike in OTP systems, what is represented in ERP workflows are real-life processes involving people. In this chapter, those challenges are examined and various concepts and mechanisms are proposed in a new model that can be used to represent ERP workflows as long-running transactional models.

The second challenge is with regards to the inherent complexity of the ERP models, as the sub-business-processes represented within the ERP grows, the number of possible interactions within the model grows exponentially. These cause significant challenges to system verification and implementation of such systems. In this section, a solution to

this ERP complexity issue is demonstrated by exploring a technique –called Communication Closed Layers (CCLs) –used in Concurrent and Distributed System (CDS) to manage complexity through compartmentalisation of the system to limit the possible number of interactions within the sub-systems. The rest of this chapter describes the challenges and the design of the proposed solution and motivation behind the solution.

6.2 DESIGN CHALLENGES

In many contexts where online transaction processing techniques are used, the activity often represented is virtual (e.g. fund transfer from a bank account) which makes undoing and abandoning the transaction relatively trivial. However, in the context of ERP transactions represent the model of a real-world activity and processes involving employees, machinery and materials where some actions cannot be physically undone. Furthermore, unlike transitional transactions, these workflow transactions could potentially run for many hours or even days. What makes these scenarios more interesting is that in many situations undoing and/or abandoning a transaction halfway through in the face of failure is much more complicated given that they represent real-world activity that cannot be physically/practically undone.

For example, a long running ERP transaction that represents a manufacturing process cannot be simply abandoned in the face of simple failure in one stage of the manufacturing process, as one would do in traditional transaction processing systems. This is because within an industrial setting, such failures would be dealt with differently as it is impractical to simply undo work or cancel the entire transaction. Therefore, it should be clear if there is a need to use the transaction processing techniques to model long-running ERP workflow transactions, additional modelling concepts and techniques are required.

Take a simple example of an individual who would like to attend a football match in a major football tournament in a foreign country. Intuitively, such an individual has three requirements; plane tickets, tickets to the game and accommodation during the stay. However, failing to secure single one of those three requirements would make the entire endeavor useless. Therefore, all three requirements need to be secured.

However, each of the three bookings may be handled within different systems that are managed by different organisations without any prior coordination. Thus, the football spectator has no other option but to (as depicted in the following diagram) book each of the entities individually ideally in the ascending order of difficulty of cancellation. If the user is unable to ensure confirmed booking at the second or third stages then, he/she would have to cancel the previous bookings he/she made in (first and/or second stages). In this example, cancellation of plane ticket is assumed to be relatively easy (done well in advance) than cancellation of hotel reservation and the cancellation of hotel reservation is easier than canceling tickets to the football game. Figure 6.1 below illustrates this transaction example.

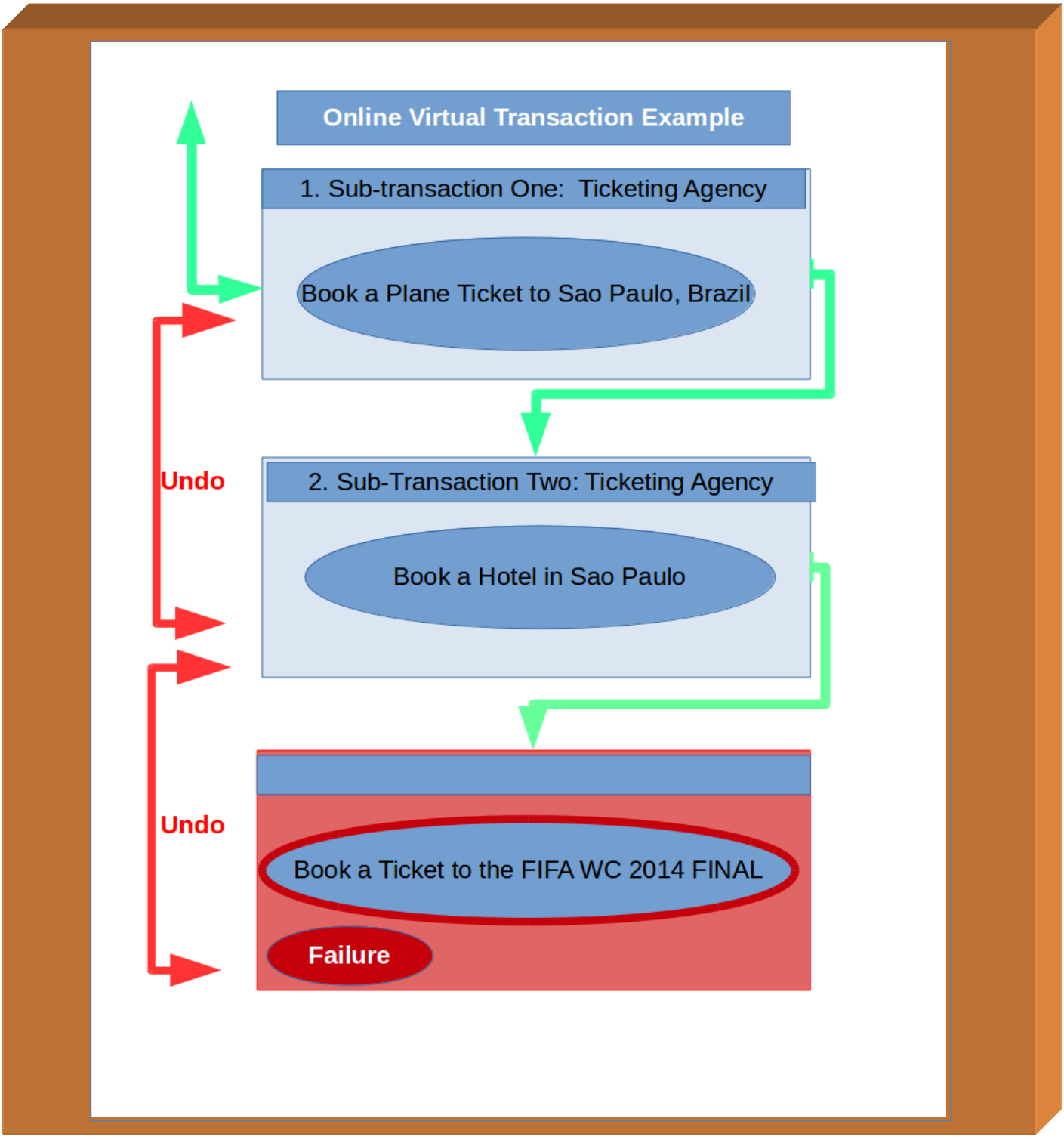

Figure 6.1 Transaction Example

In this particular instance, the cancellation concerns prior "bookings" which are essentially virtual entities. Thus, they can be reverted without incurring financial losses to either party. However, if one takes a ERP workflow of a manufacturing plant where raw-materials are put through multiple production processes, it would not be possible to cancel or abandon an order once the production process began, as it represents a physical process that cannot be reversed (they will also have direct financial implications).

The second challenge is with regard to the inherent complexity that exists within ERP systems. Since there is an attempt to model extremely complicated real-world processes, complexity is one of the key aspects that needs to be considered in any solution. After-all, ERP systems are some of the most complicated and expensive software suites on the market – often costing millions of dollars.

The aim of many modern day ERP system is to provide a real-time view of the entire business operations in an enterprise comprising of many interacting sub-processes (e.g. orders, manufacturing, raw-material management, production capacity, purchase orders, payroll, delivery system, etc). Therefore, in order to provide an accurate picture of the entire business process, not only do all these sub-processes needs to be modelled, but also all the interactions between these sub-processes need to be represented within the ERP model. The unstructured enterprise workflow diagram is shown in figure 6.2 below.

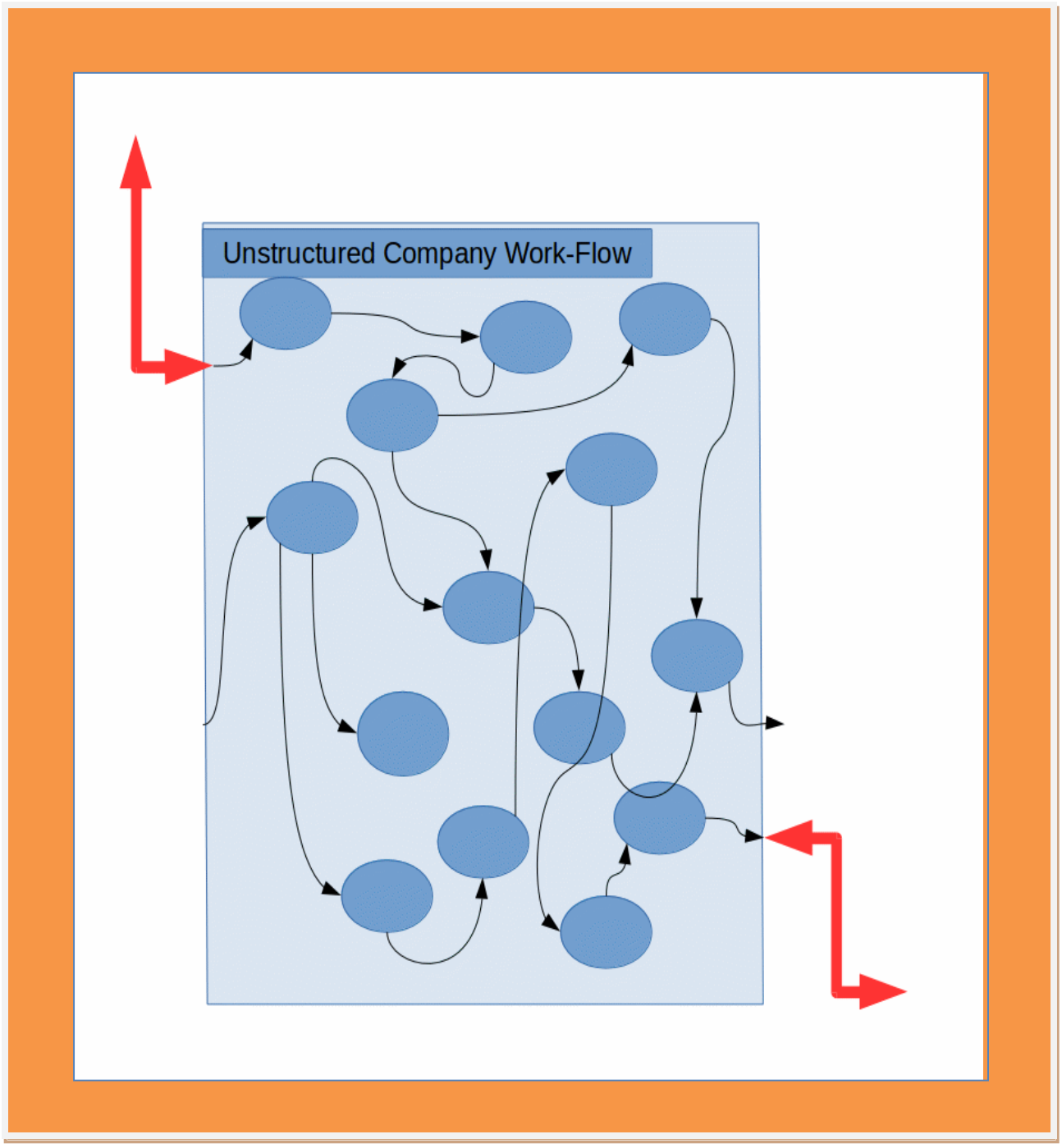

Figure 6.2 Unstructured Enterprise Workflow Diagram

It is clear even for an enterprise of very moderate size, as the number of sub-processes grows the complexity of the model will grow exponentially. To illustrate this in concrete terms, take an example of company that requires N sub-processes to represent its operations. The number of possible interactions is given by$N/2 * (N - 1)$. Assuming the all sub-processes form a fully connected graph, the possible number of paths between any two nodes is extremely large, given by the equation presented below. For a company with 50 sub-processes the number of possible different paths between the start and end process (meaning possible transaction paths) would be: 4133703843963862928624829050465088665149224322466937815041264922 5!

Number of paths in a fully connected graph can be calculated using the following equation:

$$\sum_{k=1}^{k=50} (50!/2(50-k)!) \qquad (5.1)$$

Such complexity poses problems in terms of increased development and verification costs, additionally, testing such systems is problematic as exhaustive testing and verification efforts are physically impractical. Therefore, it should be clear in order to provide a realistic solution, mechanisms must be used within the solution that would enable reducing the complexity of the resulting ERP models, because as can be seen from the previous example with only 50 sub-processes, many realistic real-world ERP models contain thousands of sub-processes.

The goal in this thesis is to introduce a new modelling framework for modelling a modern Enterprise Resource Planning System (ERP) using the Communication Closed Layers (CCLs). The transactions with the ERP are realised as Long Running Transactions (LRT), where the long-running transaction is sub-divided into smaller sub-transactions. Then, the possible use of rollback and check-pointing activities are explored as modelling techniques for some of the key characteristics of any ERP

workflow. Further, there is a need to implement a specific computational model of this ERP model to gain insights into the benefits of using the CCL approach for modelling and any performance bottlenecks or additional improvements an organisation would gain using this novel approach.

6.3 COMMUNICATION CLOSED LAYERS (CCLS) IN ERP CONTEXT

CCLs are proposed as methods of decomposing distributed and complex concurrent systems into groups (layers) that interact only within the group. This provides a means of tackling complexity that is inherently present in systems which naturally manifest as a large number of independent concurrent processes. This is the case in ERP systems as well as many distributed and complex concurrent systems. In fact, ERP can naturally be expressed as number of independent and interacting processes, as that would be the physical realisation of the organisationbeing modelled, as often companies are structured hierarchically into independent business processes.

As demonstrated in the previous calculation with only 50 sub-processes, the number needs to be managed and contained. Otherwise, the task of system development and verification becomes an extremely error-prone and costly endeavor. This is because not all paths through the system can be contemplated let alone put through verification. Figure 6.3 shows the compartmentalised ERP workflow model.

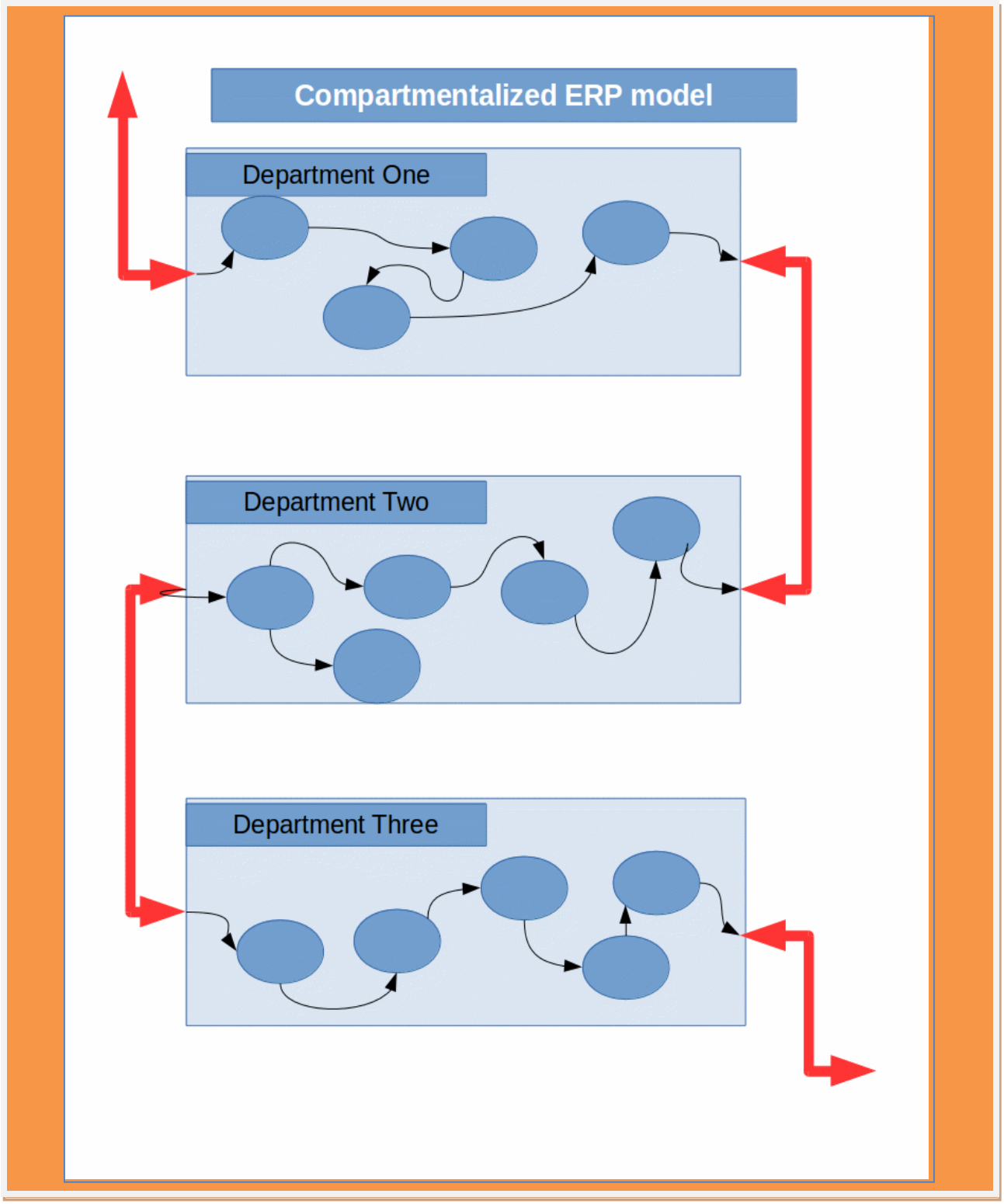

Figure 6.3 Compartmentalized ERP Workflow Model

By grouping the logically related processes into one group, which only interact with the processes within that logical group, the complexity of the entire system can be vastly reduced. To illustrate this point, take the previous example into consideration where 50 independent processes are taken (which is assumed as a completely connected graph in order to allow making some crude comparisons). If those 50 independent processes are grouped into 5 groups (or layers) with 10 in each of the grouping, then by applying the previous formula to calculate the possible transactional paths within those 5 layers, it is realised there would be only 160 paths instead of 4133703843963862928624829050465088665149224322466937815041264922 5. This is obviously a much more manageable number for the system architects to deal with.. On the other hand, assuming a fully connected graph might be a slight exaggeration of the reality, but it would be fairly obvious that the number of possible paths through an ERP system, even of moderate size, would be absolutely enormous. Especially, when one considers that the realistic ERP model of an organisation would contain not 50 sub-processes but many hundreds or even thousands.

6.4 THE GENERIC MODEL

In order to achieve the required goals, the two identified challenges in the previous section are setout and addressed, where a model is developed as well as a framework that could be used to model and implement ERP workflows as long running transactions is introduced. Within the presented solution, techniques and mechanisms are developed to deal with failures within the long-running transaction which are otherwise difficult to model within the current long-running transaction models, which make them unsuitable for modelling ERP systems.

The following section introduces all the concepts of the new model and the framework for modelling ERP workflow into long-running transactions. Also, it explores how the concepts of the Communication Closed Layers (CCLs) approach are used to minimise

the complexity of the resulting system, enhance the system verification process and to improve the easiness of the system development process of such ERP models.

As with any long-running transactional models, the model enables the long-running transaction to be split into a number of smaller sub-transactions. These are essentially the interactions between sub-business processes within the ERP model. These sub-business processes are represented by an entity called an activity. These activities have types based on physical characteristics of their physical nature, for example, the difficultly of being undone or reversed. Groups of related activities are grouped into a process which acts as a container for a group of logically related activities. The activities do not physically interact with one another, even within the processes. This is a conscious design choice to minimize complexity that could result as the number of activities increase. The interactions, or how number of activities (sub-business processes) connected together to form a long-running transaction, is done through a concept called schedule where the system designers are explicitly required to construct a schedule through which a transaction could proceed through a group of activities. Those concepts are described in the following sections.

6.4.1 ACTIVITY

Each activity represents a sub-business process, within each of these, there are two main functions for concerning how the work related to that activity is carried out, i.e. meaning the steps/stages of work done within that part of a the sub-transaction. This function is called the Do_Work. Within the Do_work function, all the steps need to be taken for successful completion of the sub-transaction that is specified. Activities have a generic concept of capacity, for example, in a manufacturing process this may mean the machinery available for the manufacturing process, or in a people-oriented process this may mean the number of people that are available. The Do_work function will allocate capacity before it starts processing the request and releases the capacity as appropriate at

the end of the sub-transaction. Within each Activity, the Undo_Work function represents the aspect of dealing with failures within the chain of sub-transactions. If subsequent sub-transactions fail the sub-business process, this activity is represented within the Undo_Work function. The layered ERP workflow model is shown below in figure 6.4.

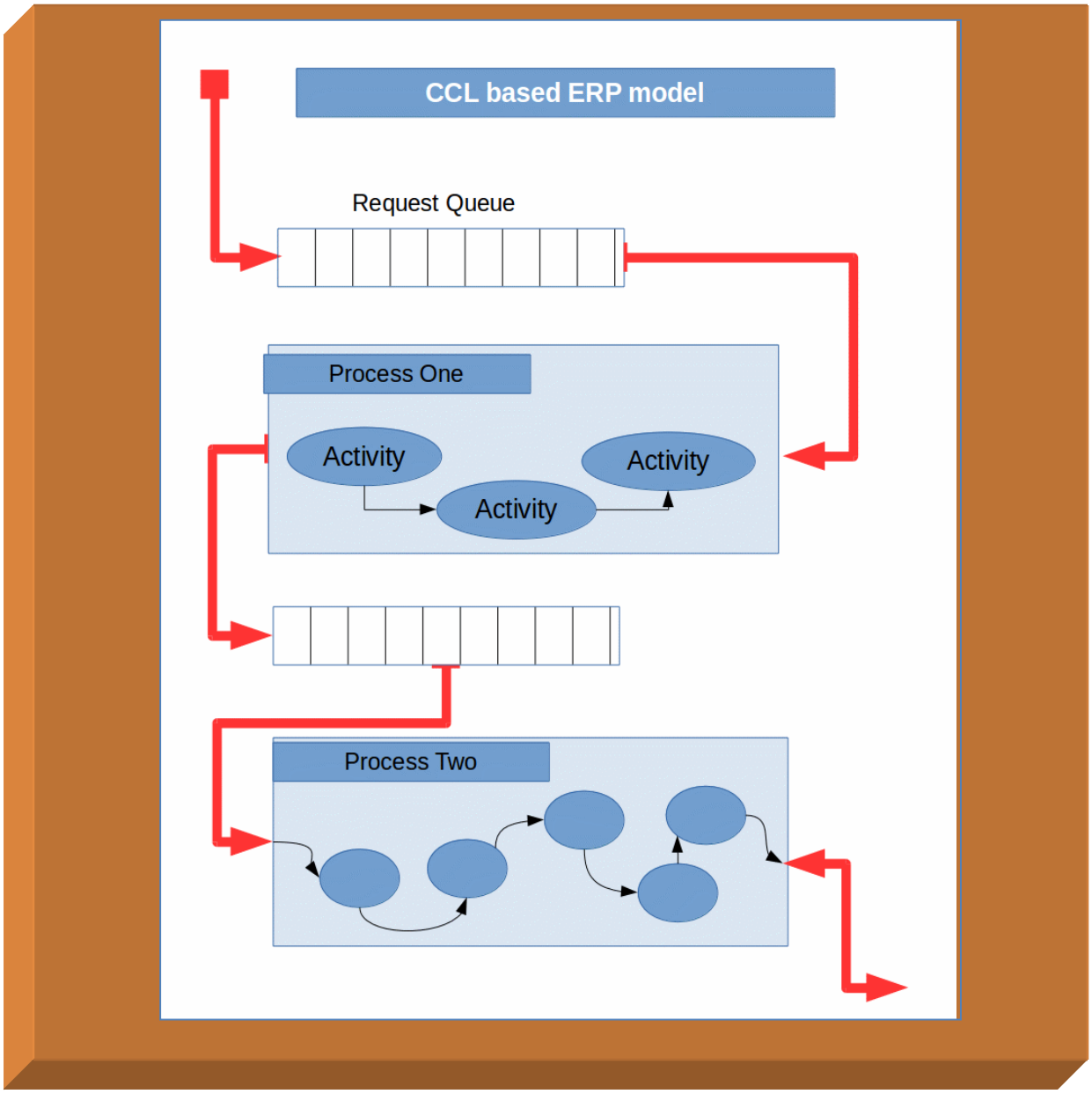

Figure 6.4 Layered ERP Workflow Model

The appropriate approach to deal with failure actually falls under the number of different categories. Therefore, the activities are divided into three different types as follows:

1. **Normal Activities**: - These activities can be undone in the face of subsequent sub-transaction failure.

2. **Check-Point Activities**: - Although these activities can be undone, they form the point in the sub-transaction chain where re-tries are attempted. This means if there is a failure, the Do_Work function within an activity down the chain of activities, then, all the activities (sub-transactions) preceding the failure can be undone (by using the Undo_Work function) until there is check-point activity, at which point the reversal can be terminated and the transaction re-tried/re-executed (i.e. by re-invoking the Do_work function of all the subsequent activities). Unlike critical activities, these represent processes that can be undone if faced with complete failure (i.e. if subsequent re-tries fail).

3. **Critical Activities**: - These form sub-business transactions that cannot be undone once completed. Therefore, if there is a failure in subsequent activity in a chain of sub-transactions, then the rollback of activities cannot proceed beyond the nearest Critical Activity.

Additionally, since the aim is to build a computational model of an ERP workflow, there is a need to simulate the workflow system and long-running transactions without having a real ERP system. Therefore, various parameters are introduced to the presented model, where they are required to simulate a computational model of a real ERP workflow system in operation. Those parameters are as follows:

1. **Do_Work Delay**: - It is the time required to complete the Do_work function, which represents a set of steps that are required to be carried out for successful completion of that activity (i.e. the sub-transaction).

2. **Undo_Work Delay**: - This is the time required to deal with a subsequent rollback of the activity. This is only applicable to check-point and normal activities as critical activities cannot be undone.

3. **Failure Probability**: - Since there is a need to simulate failures as real-life systems which would have to deal with failures, various mechanisms are used to inject failures into the system. Although introducing random failures is an option, in real systems every sub-business process is associated with different failure probabilities and failure-rates. For example, the delivery process would have a higher probability of failure or delay during severe winter months than in the summer, or an error prone manufacturing process might have higher failure probability than a straight-forward/simpler manufacturing process. Therefore, a failure is probably associated with each of the activities which can be adjusted as and when required to form a more accurate computational model of the ERP workflow in use.

6.4.2 SCHEDULE

The schedule represents the mechanism for forming Long Running Transactions (LRTs) from series of individual and independent activities. This is done by explicitly specifying for each valid transaction the order of series of activities that must traverse. Therefore, for each process in the ERP workflow, the model contains a number of predefined associated schedules. .

By requiring explicit declaration of these schedules, prior to the execution of the model eliminates another area where system complexity could become an issue. As previously stated, by grouping related activities into a process, the addressed issue of the

unmanageable number possible transactional paths through the system is eliminated. However, there is still a potential danger within each process, especially with processes that represent extremely complex enterprise tasks which contain a large number of individual activities within them; there is still the danger of facing the same issue of the number of possible transactional paths becoming unmanageable.

6.4.3 PROCESS

Processes form the container for the related activities. In the ERP context, the process may be the representation of the separate department or an independent unit within the enterprise. Processes only interact with the external world only through the associated queues, where process has an associated process request queue from which it takes a request and executes the series of sub-transaction defined with the schedule associated with that requested transaction.

Process periodically lines up a process request from the request queue and executes the appropriate transaction associated with that request. Each process request contains the transaction type. The process contains a map between the Transaction Types and the schedules. When a new request arrives, the process finds from the mapping the appropriate schedule for the requested transaction and executes it.

In some scenarios, it may be necessary to represent multiple processes of the same type. For example, a manufacturing company with multiple independent production lines would require the representation of these multiple processes (of the same type) within the ERP model. This is achieved in the ERP model by associating multiple processes with the same request queue, where multiple independent processes line up from the request queue and proceed to process them entirely and independently as would happen for physical processes that need to be modelled.

If the model contains a series of processes, when the request reaches the final activity of the process, it places it on the request queue of the next process that is required to be executed, and the same pattern continues. When the request reaches the final activity of the final process of the ERP model that represents the enterprise, the transaction is completed.

6.4.4 *PROCESS REQUESTS*

Process requests contain the transaction type and all the information required by that specific transaction type. It contains, similar to a schedule, the series of independent processes that the transaction is required to traverse.

6.4.5 *EVENT GENERATOR*

The aim of this work is to realise an ERP workflow model as a Long Running Transaction (LRT) for evaluation purposes that require simulating a computational model of the ERP workflow system which mimics a real ERP system. Within such a simulation, various events that initiate actions with the ERP system are required. This is done through the use of an event generator which can be configured with various mathematical parameters to control the frequency and mean time between the events. These events initiate various ERP workflow paths (i.e. transactions). For example, in a manufacturing oriented enterprise, this may be done through injecting/placing customer orders into the simulation.

Rather than placing random requests into ERP simulation, a configurable event generator is devised into the design as a first class entity. The event generator generates events within an exponential distribution where the timing can be adjusted as required.

As an example, for a company that sells computer equipment, the mean order arrival-interval may be a couple of hours or even minutes (depending on the company size). On the other hand, for a company that sells luxury boats, the mean order arrival-interval may be defined in terms of months.

6.5 SUMMARY

In this section, the importance of ERP workflow systems was reviewed for modern enterprises. After that, the possibility of applying novel concepts of using long-running transactional models is examined to represent an ERP workflow.

Two key challenges of formulating an ERP workflow system as a long-running transactional model were identified. First, there are number of incompatibilities when applying concepts of transitional transaction processing to ERP context as the modelled entities represent real-life processes within an enterprise, unlike virtual transactions where on-line transaction processing techniques are often widely employed. Secondly, the inherent complexity of ERP workflow was addressed for it to be represented in a transactional model where there would be system development and verification would become exceedingly challenging as the number of sub-processes increases.

Then, it is proceeded to formulate a long-running transactional model that can be used as a framework for modelling specifically ERP workflow systems. The complexity that arises from an unmanageable number of transactional paths is addressed through the system through the use of Communication Closed Layer (CCL) approach by grouping related sub-business activities into independent CCL layers.

The incompatibilities of the traditional transaction model are addressed through the introduction of additional semantics to enable the business processes to be modelled in a

manner that would represent the real-life execution of those transactions. In the following chapter, the implementation of a case study using this framework is described.

CHAPTER 7

IMPLEMENTATION AND CASE STUDY

This chapter introduces the computational model of an Enterprise Resource Planning (ERP) workflow, defines the practical users of ERP computational models, offers a case study and describes the core model implementation in detail.

7.1 INTRODUCTION

In the previous chapter, various areas are identified where improvements could be made to the implementation methodology of ERP workflow systems by applying a novel approach. This is mainly performed through the application of a long-running transactional model paradigm to the ERP context. In doing so, a number of challenges are identified. These challenges are addressed through the formulation of a new long running transactional model that is designed to be specifically applied to ERP workflow systems.

In this chapter, the implementation details of the theoretical model that was proposed in the previous chapter are discussed. Instead of dummy implementation of the transactional model, which would not have offered much more insights into the use or practical benefits of ERP systems in the industrial setting, the implementation is geared towards a computational model of a ERP workflow system. By configuring the model appropriately, one could simulate real-world enterprise scenarios. Therefore, the implementation described here can be run as simulation of a real ERP workflow system which can enable evaluating the long-running transactional model that is being formulated in this work.

Additionally, computational models of ERP workflow systems are useful tools in their own right. Such models are used in industry for simulation of future scenarios and to analyse adverse conditions. Therefore, the implementation presented here is also designed in a way that the list of parameters for each sub-process within the business can be configured (e.g. failure probabilities, delays, etc).

In the following section, a motivating example is provided to showcase the benefits of the computational model in a work-place setting. Thereafter, a case study is presented which is used as the basis of evaluating the system (which is presented in the following chapter). The case study is based around a real manufacturing plant and its workflows of how an order placed by a customer proceeds through the company's various sub-business processes through to completion.

The next section of this chapter discusses the internals of the software implementation of the system which is achieved using the Java programming language. Initially, the implementation of the core long-running transaction model framework is discussed. Thereafter, the implementation of the case study using the core framework is discussed.

7.2 Computational Model of an ERP Workflow

In order to demonstrate the feasibility of implementation of the proposed long-running transactional model and to provide a convincing demonstration of it correctness, some appropriate transactions need to be implemented using the core framework. This is done through an implementation of a computational model of an ERP workflow using the proposed long running transactional model. This section provides a motivation for the use of computational models in the ERP context and describes implementation in detail.

Modern enterprises are ever increasing in complexity due to various market demands. Therefore, it becomes increasingly important for managers and executives to be able to observe an overview of the entire operation of the enterprise, often in real-time. This is what most modern ERP workflow systems –costing millions of dollars—aims to provide.

However, sometimes, having only a real-time view of the business operations is not adequate to make important future decisions. Although an ERP system could show the operational view based on the current events, it cannot thesis into the future to help managers to make informed decisions. For that, computational simulation of the current business processes (using the past and predicted data) is required. For example, imagine a scenario where a manufacturer is faced with a scenario where he/she must choose from two production plants with two different up-time guarantees. But, this is at vastly different price-points, say one has guaranteed up-time of around 90%; while the second choice has only 70% guaranteed up-time (meaning only 70% of the operational duration can be used for production). Additionally, the first choice also offers a 20% faster manufacturing process. But the second choice, with all its disadvantages, is 3 times cheaper. Such business decisions are not straightforward to make, in this particular case the choice that is most cost effective is far from clear. It depends on the demand pattern of the manufacturer (arrival of orders for manufacturing).

In such scenarios, through the use of computational models, both scenarios can be simulated, by modelling the current company set-up and plugging in the data for two different manufacturing plants to see how they perform in various past conditions (using past and predicted order arrival data) within the current company set-up. Managers could analyse and compare the effective delivery times using both plants and make various other important comparisons. It may be the case, given the specific manufacturing order arrival patterns of the company are much cheaper, where less reliable manufacturing plants may not increase the delivery time-frame significantly to adversely affect customer satisfaction. Thus, enabling the company to make extremely intelligent choices would make a large cost saving. In large organisations, unlike the simple example described here, there are hundreds of parameters that must be taken into account. Such decisions cannot be made without a computational simulation of the organisational workflow model.

6.3 CASE STUDY

As described earlier, a realistic use-case needs to be implemented for the evaluation of the proposed long-running transactional model. For this purpose, a production process of a manufacturing plant is chosen, where all the sub-business processes are modelled from client order arrival through to the final delivery of the goods. The following section describes the case study which forms the basis for the computational model and the evaluation of both the core transactional model and the usefulness and practicality of the computational model workflow.

The case study is based on the workflow of a manufacturing process from a real-life manufacturing plant that is simplified to enable concentrating on the key aspects of the process. Thus, the minute details and various complications are abstracted away. The manufacturing process can be sub-divided into five broad areas that largely consistent with the company's departmental boundaries. These are as follows:

1. **External Client**s –Represents the interactions with individuals who place orders with the organisation.

2. **Finance Department** – Represents all the interactions with clients regarding financial matters such as payments and negotiates prices for goods with the clients. Once the orders are confirmed by clients through payments, it initiates the manufacturing process.

3. **Manufacturing Plant** –This deals exclusively with the process of manufacturing, once the orders are confirmed, the manufacturing process initiates the interactions with the layer below; the stores, to obtain required raw-materials. All the sub-manufacturing processes and allocating manufacturing resources are represented within this section.

4. **Stores Management** –This deals with required "bill of material" for each order of the interactions with the external raw material providers.

5. **Supply Chain –**This represents the external providers of raw-material required for the manufacturing process.

Each of these sub-sections is divided into number of sub-business processes each of which have been represented as an activity in this implementation. All of these activities, as described in the previous chapter, are given an activity-type. Therefore, each of these activities belongs to one of three types defined in the model: critical, check-point or normal.

For example, the manufacturing process contains four separate activities. Initially, there is a resource allocation phase where manufacturing resources are allocated for newly initiated customer order. Thereafter, the processes of manufacturing are sub-divided into two phases where the raw-material is put through two separate production lines, one after the other. Finally, there is a quality control check. If the quality control check fails, the processes of manufacturing must be repeated for the current customer's order. For that to happen, the transaction needs to be rolled-back. In order to ensure proper rollback behavior, the first activity of the manufacturing process is modelled as a check-point activity. Essentially, this means that any subsequent failure in any of the following activities, transaction is rolled-back until this Check-Point Activity. Figure 7.1 shows the activities within the manufacturing process.

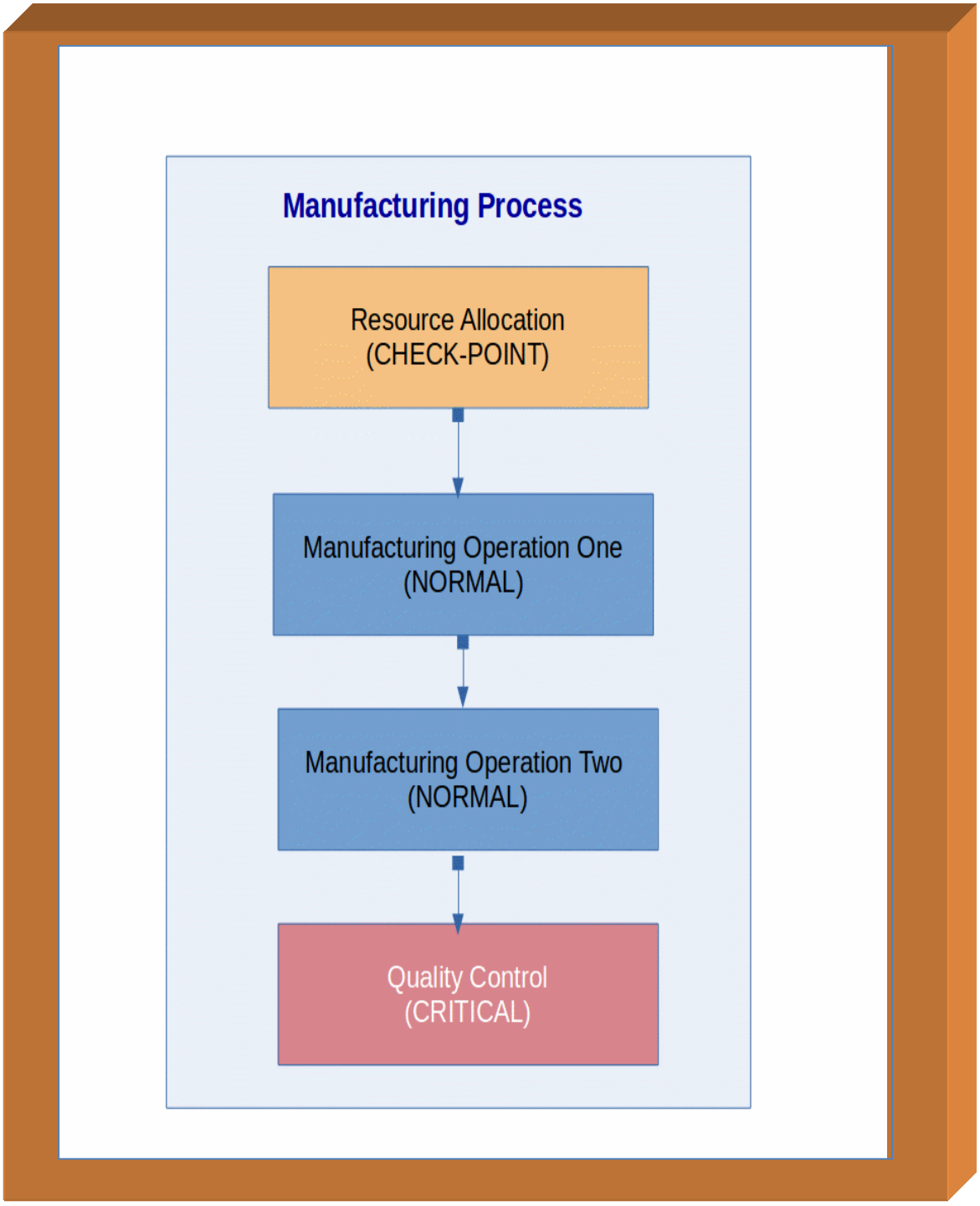

Figure 7.1 Activities within the Manufacturing Process

During the manufacturing process, if there is a failure within any of the manufacturing operations or in quality control activity, the transaction is restarted from the first manufacturing operation.

If the last activity of the manufacturing operation is successful, i.e. the quality control activity, this means that the entire manufacturing operation with respect to the current customer order is successfully executed. Thus, common-sense dictates that under no circumstances, the successfully executed manufacturing operation should not be re-executed or rolled-back. Such requirements exist not just in this particular part (i.e. Manufacturing Process) of the transaction, but at multiple points throughout the entire long running transaction. Such requirements can be accommodated within the proposed model, by categorizing them as critical activities. These activities, once successfully completed, cannot be undone or reversed. Therefore, if a transaction fails midway at a particular activity, where there is a preceding critical activity which was successfully executed prior to the failure. This means that the transaction cannot be aborted as the critical activity cannot be undone. Furthermore, the transaction can be rolled-back until the critical activity (if there are no other Check-Point activities before that) and re-executed. The current implementation does handle such rollback and retry behavior.

Table 7.1 below provides a full listing of all the activities and their activity-type for the entire case-study. The activity-type is indicated through colour codes (critical activities are marked red, check-point activities are marked yellow and normal activities have no colour).

Table 7.1 Activities within the Case-Study

Sub-Process	Activity
Client Interactions	Order Placement
	Order Completion
	Final Delivery
Finance Department	Assess Feasibility [of the] Order
	Client Payment
Manufacturing Operation	Resource Allocation
	Manufacturing Operation One
	Manufacturing Operation Two
	Quality Control
Store Management	Assess Material Requirements
	Place Orders
Supply-Chain Management	Supplier A Order
	Supplier B Order
	Supplier A Delivery
	Supplier B Delivery

Each of these activities is implemented in a separate Java class, and each one has set of parameters that are used for the simulation of the computational model. For example, the time for successful complex execution of the activity, the cost of failure, probabilities of failure and rollback delay are contained in the

Activity_Configuration_File. The mechanisms for loading each of these confutation files for the simulation are discussed later in this chapter.

The following table (Table 7.2) describes how these list of activities are sequenced together to form a long-running transactions with some additional comments.

Table 7.2 Description of the sequence of activities together

Organisation's ERP				
External Client	**Finance**	**Manufacturing**	**Stores**	**Suppliers**
Order for goods►	Assess Manufacturing plant capacity/ availability ▼			
	If spare capacity available, reach agreement with client for price/payment ◄			
Client issues a formal Purchase order and initial payment ►				

	Once the payment received, initiate the manufacturing process ►			
		Obtain bill of material for the order and request them from stores ►		
			Check internal availability ▼	
			Place orders with external suppliers for what is not available ▼	
			Issue Order for Material X from supplier A►	
			Issue order for Material Y from supplier B ►	

				◄Supplier A fulfills
				◄Supplier B fails
			Re-order Material Y from supplier C' ►	
				◄Supplier C' fulfills
			Send all raw-material for the order to manufacturing plant ◄	
		Provision machinery for the manufacturing of the order ▼		
		Execute the manufacturing ▼		
		Manufacturing Operation One		

		Manufacturing Operation Two		
		Check the final goods do meet the quality and requirements criteria ▼		
		Inform manufacturing process has completed ◄		
	Inform the Client that the goods are ready for delivery and request final payment ◄			
Client makes final				

Payment ►				
	Goods are delivered ◄			
Order completed				

7.4 IMPLEMENTATION

The implementation is partitioned into eight packages as shown in Figure 7.2 below.

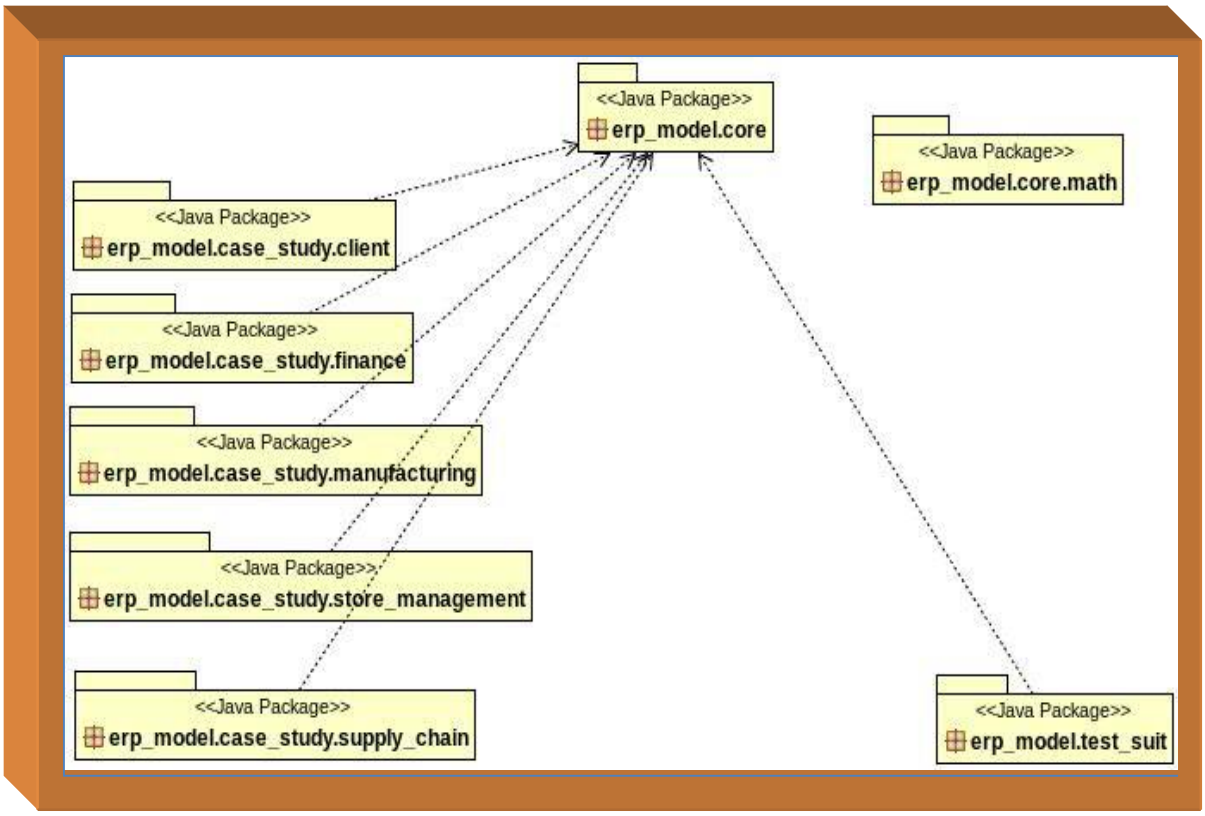

Figure 7.2 Project package hierarchy

The core transactional model is within the "erp_model.core" package. This includes the implementation classes for all the key concepts and mechanisms of the proposed transactional model that were introduced in the previous chapter, namely; activity, schedule, process and activity-type. The case study is implemented within five packages where each one represents a sub-business process. All the activities that are part of each sub-business process are implemented within that package. The "erp_model.math" contains all the mathematical probability functions that were used within the simulation such as random number generators for exponential distribution and uniform distribution. The package named "erp_model.test_suit" contains the test rig which putstogether all the activities into a Schedule (long running transaction) and creates and initiates the process thread to start the simulation.

In the next two sections the implementation of the core model and the simulation aspects of the computational model are discussed, with the aid of UML diagrams.

7.4.1 IMPLEMENTATION OF THE CORE MODEL

All the core concepts are implemented in separate Java Classes (e.g. activity, schedule, process, event generator, activity request and activity response). There are numerous support classes within the package such as logger that perform various utility functions. UML diagram for the classes within the core package is depicted in Figure 7.3.

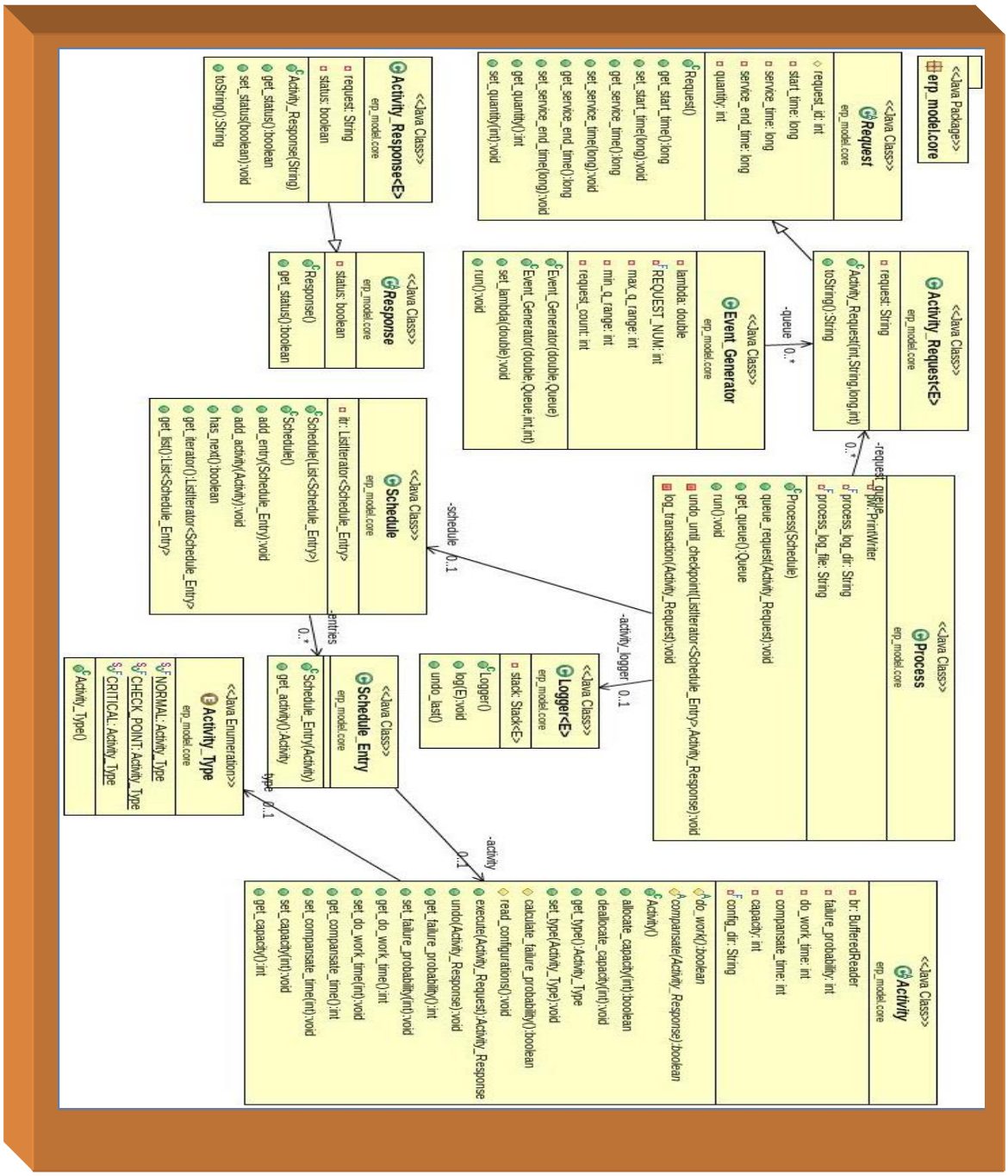

Figure 7.3 UML Diagram of ERP_Core Package

One of the noteworthy features of the implementation is the implementation construct of the concept activity. Each activity has defined within itself, how that activity is executed and how the reversal of the execution—if undoing is allowed—is performed (to be used in case of a subsequent failure). The activity class is defined as an abstract class with these two methods as abstract methods:

1. Boolean do_work () --defines how to execute the activity

2. Compensate () --how to handle subsequent failures

Therefore, any new activity that is required by an application is implemented by inheriting this class (through extends) and providing implementation for these two methods.

7.4.2 SIMULATION ASPECTS OF THE COMPUTATIONAL MODEL

All the simulation aspects of the activity are contained within the abstract activity class. Activity class contains the mechanism for loading the "configuration_file" for each Activity. This is done through a method in the activity class named "read_configurations()". This method reads a text file from the directory which contains all the configuration files for all the activities. This is done using a naming convention where activity configuration files is named the same as the corresponding class name that implements that Activity with the .txt extension.

The "read_configuration()" method is called within the constructor of the abstract activity class, thus ensuring each activity is configured with load functionality. Essentially, this ensures that all the required configuration data is initialised at the object initialisation. As mentioned earlier, all the simulation related functionality is implemented within the abstract activity class. The two methods that are called by the simulation for execution of the activity are called: execute () and the corresponding

method for undoing is: undo (). When the execute method is called, within the method, using the execution delay for this particular activity (defined within the configuration parameters) is added to the simulation timing, currently 1000 milliseconds of simulation time is taken as equivalent to one hour of real time. The delay is added through putting the activity thread to sleep for the specified delay. Once the execution delay is completed, the corresponding do_work () method is called within the activity implementation class from within the execute () method. Similarly, within the undo () method the delay for undoing work (termed compensate-time) is added. The do_work () method within the implementation of the activity uses the helper method "calculate_failure_probability()" defined within the abstract activity to decide whether the current execution is a success or failure for the purposes of the simulation.

Additionally, each activity has a notion of capacity. Some activities inherently have some capacity, for example a manufacturing process that requires a particular type of machinery is limited by the number of activates that are currently available with the company. If there are multiple concurrent orders that are being handled, some of that machinery may be already allocated for another client. Therefore, the notion of capacity is also defined for each activity. Such activities are required to allocate capacity at the beginning of the do_work () method of the activity. Correspondingly, allocated capacity is required to be released at the end of the do_work () method. The functions for this (allocate and de-allocate) is defined in the abstract activity class, which is defined as synchronized. This enables the activities to wait (in the wait queue) if the requested capacity is not available (i.e. used by another process). When the resources are released by other activity, then one activity waiting for the resources can start execution.

7.5 SUMMARY

In this chapter, the implementation of the presented long-running transactional model devised specifically at enabling efficient implementation of ERP workflow systems is discussed in detail. The chapter explains the motivation for using a computational model of an ERP workflow system as the basis for evaluation of the core concepts. Additionally, a motivating example is discussed to show how computational models of ERP workflows are being used in industry and how such tools are used for making important future business decisions.

The case-study, which forms the basis of evaluation of testing of the core framework, was then discussed. Then the scenario was decomposed into five different segments and within each segment all the sub-business processes are identified (which will later be implemented as Activities). Thereafter, how these activities were categorised into the three designated activity-types and how such categorisation would ensure the computational model would adhere to real-life physical and practical constraints of enterprise workflow was shown.

Afterwards, implementation details of the core functionality of the Transactional model were described. The process of modelling a new activity through implementation of do_work and compensate methods was discussed together with how those methods are invoked within the simulation framework. Additionally, the mechanisms used to handle various configuration parameters such as capacity, failure probabilities, execution and rollback delays, and how they are realised within the simulation was presented.

CHAPTER 8

EVALUATION AND RESULTS

This chapter introduces the evaluation of the implemented computational model performance using three scenarios; manufacturing process failure, quality control process failure and composite failure (manufacturing & supply-chain). In addition, it offers a comparison between the current model and two previous models.

8.1 INTRODUCTION

In the previous two chapters, some of the issues and avenues for improvements in current methodology of ERP workflow systems are identified. Thereafter, a new model is devised through the use of long-running transaction mechanisms. The challenges are identified which arise when applying long-running transaction concepts to the ERP context. These are addressed, first through the introduction of new semantics to facilitate the modelling of various physical and practical constraints that are important in ERP systems; and secondly, through utilisation of the concept of Communication Closed Layers (CCL) with inherent complexity that arises within ERP workflow (especially modelled as interacting sub-processes).

Thereafter, the implementation of the proposed long-running transactional model is presented, together with details of the case study which is used here as the basis of the evaluation of the core concepts as well as the performance of the computational model under a number of scenarios which is discussed in the following sections of this chapter.

The following section sets out the evaluation criteria for the proposed long-running transactional model and its implementation. The evaluation takes into account: a) the correctness of the proposed model, b) ease of use of the proposed model, and c) the performance and usefulness of the computational model and its simulation framework. In addition, a comparison is conducted among the current model and two previous models where it shows how this model outperforms other models.

8.2 EVALUATION OF THE COMPUTATIONAL MODEL PERFORMANCE

In this section, a number of scenarios are simulated to evaluate the implementation of the computational model of the ERP workflow. While the simulation framework can

perform much more complex scenarios, the chosen scenarios are simply for clarity, using only one or two variables. The rest of the configuration parameters are set to a constant. For each scenario the configuration parameters are provided in a table, with the variables highlighted.

For each data point in the simulation, 25 long-running transactions are completed and the service time is averaged to filter out anomalies introduced through a timing mechanism (as the simulation was run as real-time simulation with 100 milliseconds of simulation time equivalent to one of actual time). During each of these simulation runs, all the timing data from each of the transactions are logged. These log files are stored in the "Log" directory, with the sub-directory designated for each of the three scenarios. Then using an additional program named "Number_Cruncher", contained within the "erp_model.test_suit" package, the service time is averaged and placed in another file called the "summery.txt" for the data-set for all three scenarios. This data was then used to produce graphs for analysis.

The simulation data contains more information than that used for the graphs for the analysis below, however the information can be used to perform much more complex analysis on broader and more complicated scenarios. For example, the simulation data provides the timing information regarding the queuing delays for each of the transactions. This is essentially the time when a customer places an order with the company and the company starts servicing that order. The queuing delay happens because all the company resources are already dedicated to other customer orders. Therefore, all the customers are forced to wait in a request (FIFO) queue (First In First Out) until the company resource becomes available again. Such details can be studied through changing the customer request arrival-interval in the event generator and through varying the capacity of various activities to understand where the bottlenecks are located.

Table 8.1 shows a segment of a sample log file, where each entry represents a long-running transaction (i.e. the manufacturing workflow case-study). All the timing information is in milliseconds and currently 100 milliseconds are equivalent to one hour of actual time. For example, in the sample data set provided below, one can clearly see that the request arrival rate is much higher than the company set-up could currently serve, through observing how the request queuing delay increases gradually for each of the transactions. However, such analysis is kept for future work.

Table 8.1 Simulation Data Sample

Seq No	Requested Quantity	Order Placement Time (Unix System Time)	Service Start Time (System Time)	Service Completion Time	Service Time	Service Time + Time in Request Queue
0	12	1404186503694	1404186503698	1404186509810	6112	6116
1	18	1404186503883	1404186509811	1404186518220	8409	14337
2	20	1404186504730	1404186518220	1404186523927	5707	19197
3	11	1404186511611	1404186523928	1404186530035	6107	18424
4	13	1404186512816	1404186530036	1404186539145	9109	26329
5	16	1404186514222	1404186539145	1404186545253	6108	31031
6	12	1404186515323	1404186545253	1404186558162	12909	42839
7	18	1404186518712	1404186558162	1404186565670	7508	46958
8	10	1404186520729	1404186565670	1404186573278	7608	52549
9	13	1404186523142	1404186573278	1404186583387	10109	60245

8.2.1 SCENARIO 1 –MANUFACTURING FAILURE

In the first scenario, the failure of the "manufacturing operation two" is considered. This is done through changing the failure probability of the activity configuration parameters that corresponds to the manufacturing operation two.

Data are gathered for 6 data points that are 10 percentage points apart, starting at 10 percent failure rate which is increased to 60 percent. Then, the data is used to plot a graph depicting the service time. In order to provide a clear understanding of the service time degradation, additional reference line is plotted with raw-data (within averaging) for which failure-rates are set to the default 10percent for all the Activities. Table 8.2 contains all the configuration parameters for this simulation run.

It is worth noting that when the "manufacturing operation two" fails, the nearest preceding check-point activities is "resource allocation activity". Therefore, the transaction is required to be rolled-back until that activity and re-started from the "manufacturing operation one". This can add a significant penalty for the mean service time as the failure rates increased because each of these activities which contain significant execution delay (which will be incurred on each of the re-executions) and the sizable compensation delay. Figure 8.1 shows that the increase in the service time is exponential with the increasing failure rate.

Table 8.2 All configuration parameters for simulation run

Configuration Data			
Activity Name	Failure Probability	Execution Delay	Compensation Delay
Order Placement	10	2	1
Order Completion	10	2	1
Final Delivery	10	2	2
Assess Feasibility [of the] Order	10	2	1
Client Payment	10	1	1
Resource Allocation	10	2	1
Manufacturing Operation One	10	10	3
Manufacturing Operation Two	10-60	15	5
Quality Control	10	5	3
Assess Material Requirements	10	2	1
Place Orders	10	2	1
Supplier A Order	10	1	1
Supplier B Order	10	1	1
Supplier A Delivery	10	5	1
Supplier B Delivery	10	5	1

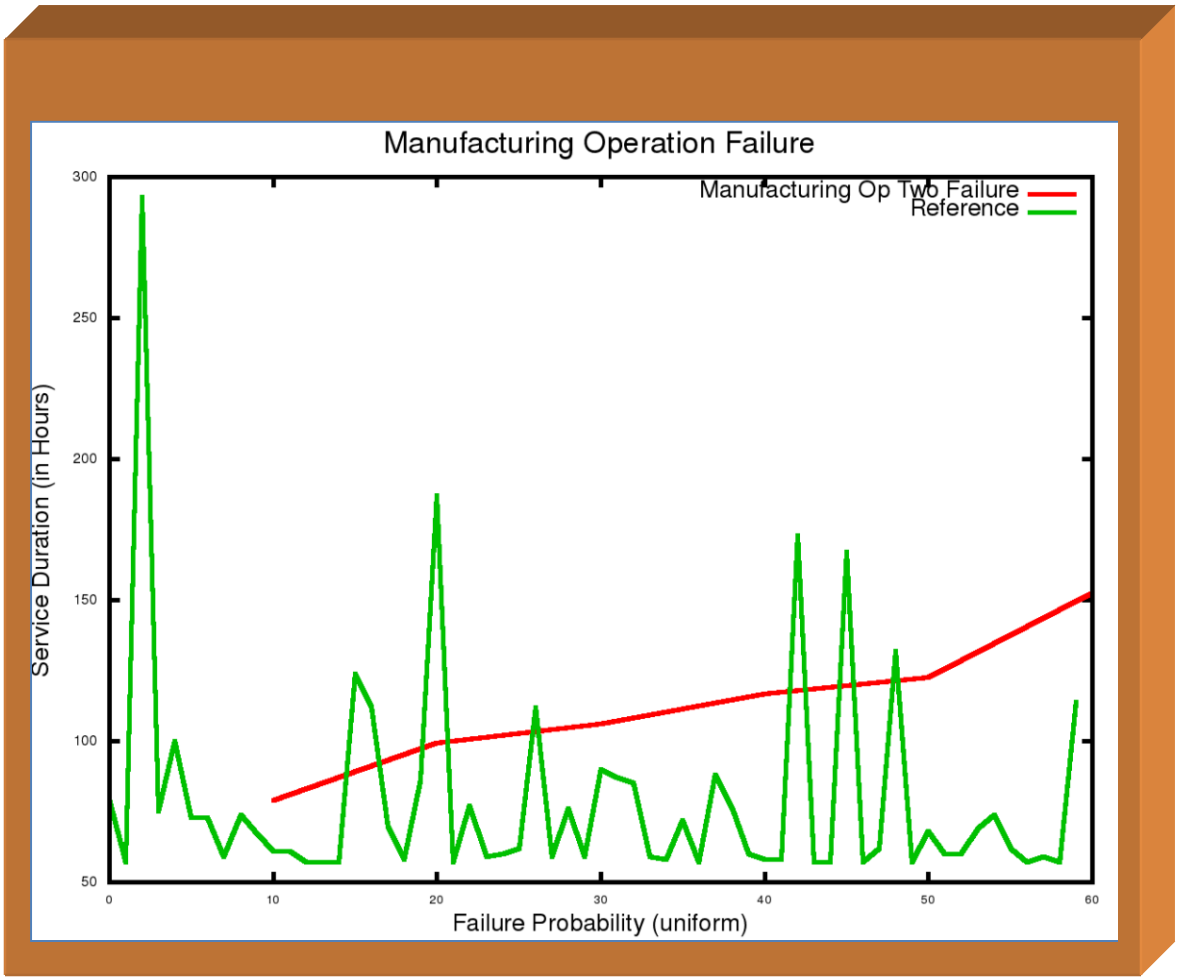

Figure 8. 1 Scenario 1 Graph (with Reference point of uniform 10% failure rate at all Activities)

The simulation data for the above scenario is plotted in Figure 8.1 with reference to the dataset of serial execution of the transaction with failure rates of all the sub-business processes (activities) which set to 10 percent (spike which can be seen in the graph represents the random failures within this 10 percent failure probability)

8.2.2 SCENARIO 2 –QUALITY CONTROL FAILURE

This scenario simulates the gradual increase of the failure probability of the "Quality Control" process. Similarly to the previous scenario, 25 transactions were performed for each data point and the mean was taken to smooth out any anomalies introduced through timing effects. As before, 6 data points were simulated with 10 percentage point increments (from 10 to 60 percent).

The failure of the "quality control" activity will result in guaranteed rollback of the transaction until the "resource allocation" activity, which is its nearest check-point activity. Unlike the previous scenario, this can add the re-execution delays for both "manufacturing operation one" and "manufacturing operation two", in addition to the compensation delay for both of those manufacturing operations as a result of the rollback. Table 8.3 illustrates the configuration parameters used for the simulation of this scenario, while Figure 8.2 shows the service time increases exponentially with the increased failure rate.

Table 8.3 Configuration parameters

Configuration Data			
Activity Name	Failure Probability	Execution Delay	Compensation Delay
Order Placement	10	2	1
Order Completion	10	2	1
Final Delivery	10	2	2
Assess Feasibility [of the] Order	10	2	1
Client Payment	10	1	1

Resource Allocation	10	2	1
Manufacturing Operation One	10	10	3
Manufacturing Operation Two	10	15	5
Quality Control	10-60	5	3
Assess Material Requirements	10	2	1
Place Orders	10	2	1
Supplier A Order	10	1	1
Supplier B Order	10	1	1
Supplier A Delivery	10	5	1
Supplier B Delivery	10	5	1

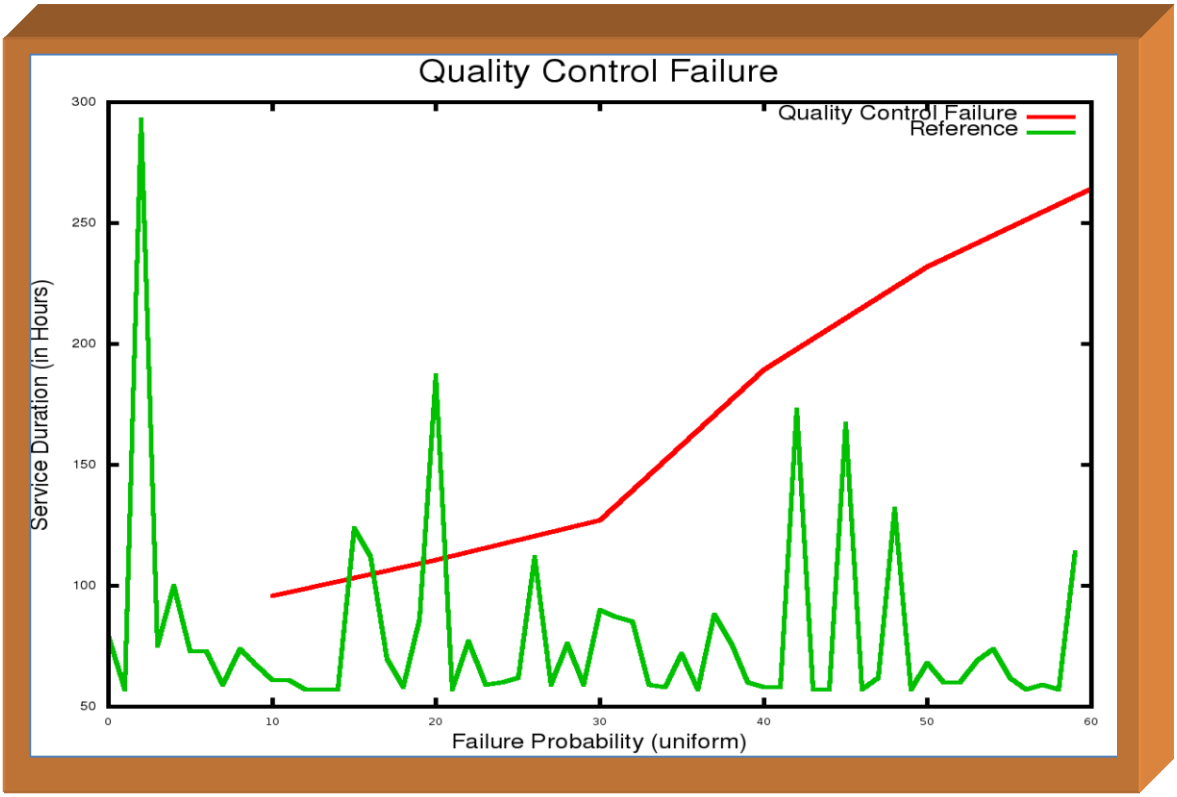

Figure 8.2 Scenario 2 Graph (with Reference line)

8.2.3 *Scenario 3 –Manufacturing and Supply-Chain Failure*

This scenario evaluates the composite failure. In this scenario, unlike the two previous scenarios, failure of two sub-processes is evaluated. Namely: the "Manufacturing Process Two" and "Supplier_A_Delivery". For simplicity, the failure-rates are adjusted in tandem in equal steps. Both failure-rate parameters for both of the mentioned activities are increased in steps of 10 percentage points, starting from the default 10 percent.

As mentioned earlier, the failure at the "manufacturing operation two" needs to be rolledback until the "resource allocation" which is the closest checkpoint activity. With respect to the failure of the "supplier A delivery", this transaction needs to be rolledback until the nearest preceding critical activity in the absence of any checkpoint activities before it. In this case, it needs to be rolledback until the "place order" operation. Table 8.4 shows the configuration parameters of the scenario where variables are highlighted. Figure 8.3 shows the graph of scenario 3 with a reference line.

Table 8.4 Configuration parameters of scenario 3

Configuration Data			
Activity Name	Failure Probability	Execution Delay	Compensation Delay
Order Placement	10	2	1
Order Completion	10	2	1
Final Delivery	10	2	2
Assess Feasibility [of the] Order	10	2	1

Client Payment	10	1	1
Resource Allocation	10	2	1
Manufacturing Operation One	10	10	3
Manufacturing Operation Two	10-60	15	5
Quality Control	10	5	3
Assess Material Requirements	10	2	1
Place Orders	10	2	1
Supplier A Order	10	1	1
Supplier B Order	10	1	1
Supplier A Delivery	10-60	5	1
Supplier B Delivery	10	5	1

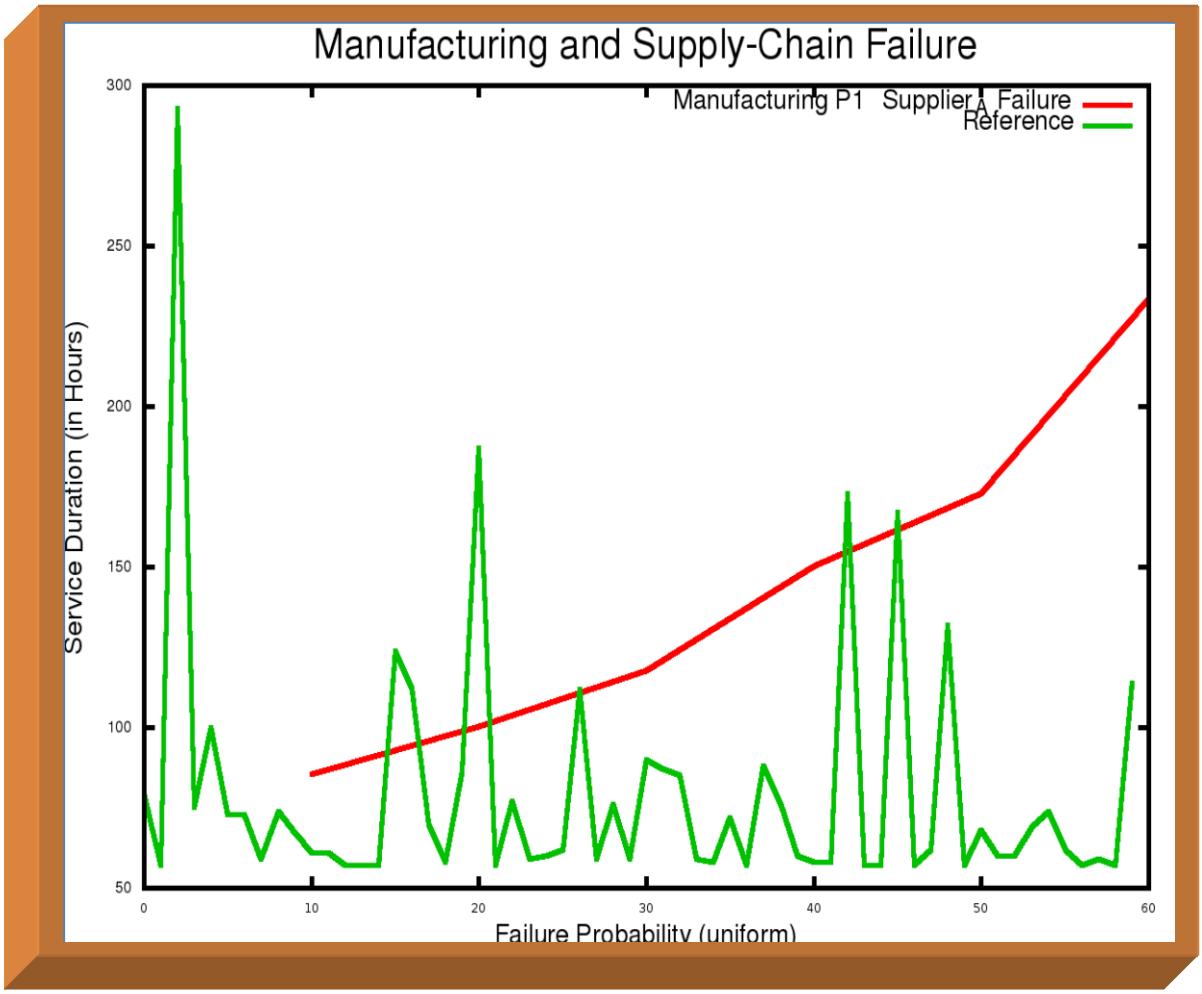

Figure 8.3 Scenario 3 Graph (with Reference line)

8.3 Evaluation Against Success Criteria

From the above evaluation of the system it is clear that the system achieved the success criteria that were established in Chapter 1. Specifically, the success criteria were formulated against the research questions of the study. Importantly, the results show that for each activity in the process, the failure probability was significantly low for the vast majority of activities at only 10 percent. The system was designed to resolve the problems that were associated with rollback check pointing, from the scenarios

presented in the above we can see that the system only allowed minimal rollback by allowing rollback to the closest checkpoint activity.

8.4 COMPARISON WITH PREVIOUS MODEL

Many transactional models are proposed in the literature to address the requirements of Long Running Transactions (LRTs), many of these relaxes the traditional ACID (Atomicity, Consistency, Isolation, Durability) properties to cope with complexities introduced through long duration and multiple sub-processes that are involved in "long running transactions".

One of the foundational models that address these issues, which form the basis of many long-running transactional models is the one that was proposed by Molina and Salem, named Sagas [108]. This model introduced the notion of splitting the long-running transaction into number of sub-transactions. Thus, for each sub-transaction, instead of a complete cancellation, otherwise used in transitional transaction processing, this model proposes the use of a compassionate procedure, where the compassionate procedure acts as a reverse of the normal execution.

The model proposed in this thesis, uses some of these key concepts introduced in the Sagas model [108], such as dividing the long-running transaction into a number of sub-transactions, and the method of "compensate" procedure for each sub-transaction. However, Sagas model is not suited to be used within the context of ERP workflow models for two main reasons: first, it does not offer convenient semantics to enable direct mapping of an Enterprise workflow into this model; second, the inherent complexities that arise from modelling each of the sub-business processes into sub-transactions is not addressed within this model. The new model, introduced in this thesis, addresses these deficiencies through introducing typing construct for each of the

sub-processes, where sub-business processes within the enterprise can be conveniently mapped to suitable types of sub-processes based on the nature of that sub-business process. For example, sub-business processes that cannot be reversed can be mapped to "Critical" type, the sub-business processes that signify a completion of a significant stage of Enterprise workflow can be mapped to "Check-point" type (i.e. processes that can be reverted in extremely rare conditions).

Furthermore, the model introduced in this work (unlike Sagas), introduces a compartmentalisation mechanism where related sub-process can be grouped together into compartments (termed processes) where sub-processes contained within a Process interacts/communicates within the Process. Since the inter-Processes communication is done through queues associated with each process, the complexity that arises due to large number of communicating sub-processes is minimised.

The work by Hu and others in [109] introduces the notion of associating a "cost" for each of the "compensate" procedures of the sub-transactions. Then through identifying the sub-transaction that has the highest "costs", the model proposes the delaying of the execution of the costliest sub-transactions to the last. There by, minimising the rollback cost in case of failure. Two issues with the approach taken in [109] were identified. First, it is not often possible to execute the sub-transactions out-of-order (to take advantage of the "delaying the costliest to last" approach proposed). For example, in the case study that was discussed in the previous sections, the "manufacturing" processes (which are the costliest) cannot be scheduled after the "quality control" process. Therefore, an explicit assumption is made within the new model that the sub-business processes are ordered identically for representation in the enterprise workflow.

Additionally, the new proposal here extends the feature used in the work in [109] where the cost is associated with the "compensate" procedure of each sub-process. This introduces "costs" for both "compensate" and "execute" procedures in the sub-transactions (termed Activities). This enables the model to be used within a simulation

framework, where it could be used to predict service times and delays under various circumstances. Also, the model introduces the notion of capacity for each sub-process which can be acquired during the "execution" of the Activity and released thereafter (this enables the representation of congestion).

8.5 SUMMARY

In this chapter, the computational model that was developed in order to evaluate the presented long-running transactional model was put through three real-life scenarios, through the use of the activity reconfigure ability feature implemented in the computational model framework. The resulting service time was plotted on a graph. Figure 8.4 shows a graph depicting the datasets from all three scenarios discussed above.

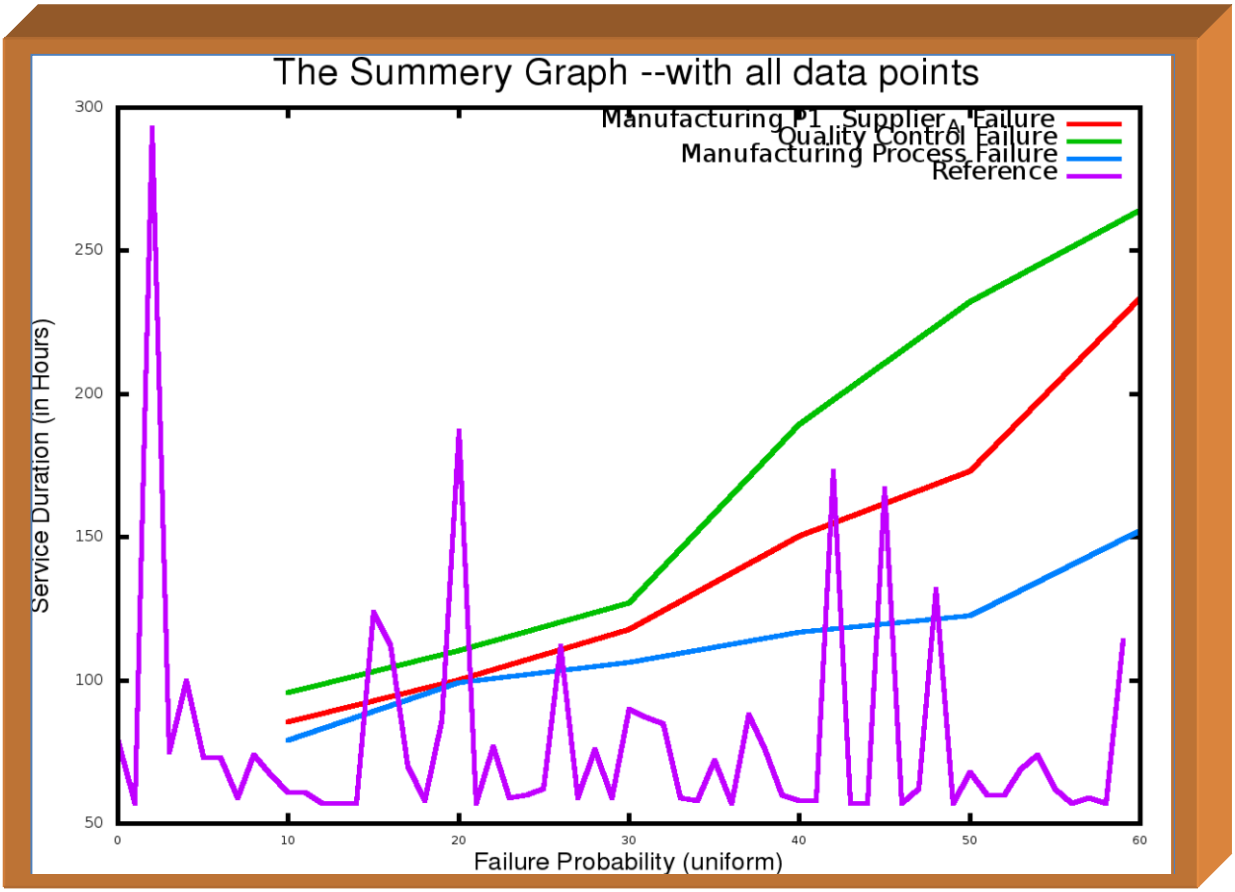

Figure 7. 4 Summer Graph

The graph shows no anomalies with regard to the data. The computational model has functioned as expected with the various changes to activity configuration parameters. This should confirm the correctness of the implementation of the long-running transactional model.

With regards to the usability of the model, it must be noted the ease of use of this computational model, with simile editing of the activity configuration files where the entire simulation framework can be configured to the desired scenario. As mentioned earlier in the document, with numerous parameters many complex scenarios can be derived and industrial settings cannot be guessed. In such environments, the ease of the use of the model is an advantage. Due to the nature of the model the implementation of the computational model of the ERP workflow and setting up a new scenario only requires editing the text files that represent the configuration parameters (contained within the activity configurations directory).

Finally, even with these three simple scenarios, with only one or two variables, the results seem slightly counter-intuitive. The summary graph clearly shows that failure at the quality control process introduces more delay into the service time than composite failure of both manufacturing process two and supply chain failure put together. Now, one could imagine how difficult it is to make guesses for real-life scenarios with hundreds of parameters.

In such scenarios, it can be concluded that ERP workflow systems, which facilitate the simulation of the current computational workflows with additional configuration parameters to mimic different anticipated conditions and choices, are extremely important tools to have at the hands of decision-makers.

In this chapter, a comparison was conducted between the presented model in this work and two previous models, where it can be summarised that the current model

outperforms the previous two models where it benefits from their features and solves their problems.

CHAPTER 9

CONCLUSION AND FUTURE WORKS

This chapter summarises the whole conducted work in this thesis, presents an evaluation, summarizes the main contributions of this work and shows some of the works that can be performed in the future to enhance the current work.

9.1 RESEARCH CONCLUSION

Corporations are structured in extremely complex ways to deal with an increasingly competitive marketplace, with offices, workforce and manufacturing plants spread over multiple countries and even continents. Managing such complex organisations is not an easy task. One of the tools that are considered essential in helping managers and executives in this task is an integrated system often termed: Enterprise Resource Planning (ERP) system, which aims to provide an integrated view of all the business processes with an organisation in real-time.

This thesis identified an approach that could bring potential benefits to the implementation methodology of these ERP workflow systems. This was through applying Long-Running Transactional (LRT) model concepts to ERP Workflow systems. The aim of the thesis was to introduce simple and elegant constructs as building blocks for implementing these complex and expensive ERP workflow systems.

There were two key challenges that were identified during the thesis, which had been addressed effectively in order to formulate the ERP workflow system as a long-running transactional model. First, there were a number of incompatibilities when applying concepts of transitional transaction processing to ERP context. This is due to the modelled entities in an ERP workflow that represents real-life processes within an enterprise, unlike virtual transactions, where on-line transaction processing techniques are often widely employed.

The second challenge was the inherent complexity of ERP workflow which needed to be addressed for it to be represented within a transactional model. Without this reduction,

the system development and verification of such systems would become exceedingly challenging as the number of sub-processes increase.

These challenges were addressed through a novel long-running transactional model that can be used as a framework for modelling ERP workflow systems specifically. The model introduced new semantics to enable the modelling of concepts such as rollbacks and checkpoints through various constraints. The complexity issue that arises from an unmanageable number of transactional paths through the system was addressed through the use of the Communication Closed Layer (CCL) approach. The new proposed model introduced a compartmentalisation mechanism that can enable sub-business processes which are related to be grouped into compartments. The compartments (termed processes) communicated/interacted with other compartments through the request queuing mechanism. The sub-process within the compartments are only able to communicate/interact with sub-process with that compartment.

9.2 STATEMENT OF EVALUATION

For the purposes of testing and evaluating the proposed generic transaction model, a realistic enterprise workflow needed to be formulated using the proposed model. Then for more numerical evaluation of the effectiveness of mechanisms, the model needed to be implemented.

Instead of implementing a dummy transactional model which would have only demonstrated its correctness, a computational model of an ERP workflow system was constructed which can simulate a real ERP workflow system. In addition, to enable gaining more understanding of the use of ERP systems in enterprise environments, reconfigure-ability was built into the system. This allows the computational model/simulation framework to be setup with various business scenarios with the use of

the activity configuration file concept, where parameters for sub-business processes can be changed through editing a simple text file. Such an executable ERP computational model/simulation framework –unlike traditional ERP work-flow system, which could only show the operational view based on the current events— can thesis into the future to help managers to make informed decisions.

To use this reconfigurable ERP computational model/simulation framework, the workflow case study was implemented. This case study was applied using different scenarios though the use of the reconfigure-ability features of the framework to simulate behaviors, such as failures and delays at various sub-business processes. The data from these simulations were then used for analysis.

Although the scenarios that were used in this analysis only had one or two variables (out of about 50 available), the complexity of the interactions between sub-process and numerous parameters is evident. It was surprising to observe that even with these extremely simple scenarios, the effects of these variables on the final customer service time could not be easily predicted. Therefore, this demonstrated quite clearly, within a large enterprise with hundreds of sub-business processes with many hundreds of variables, there is no realistic chance of predicting the final outcomes through simple guesses. Therefore, the evaluation section provides compelling evidence to support the usefulness of executable ERP computational models. It shows a simpler methodology as to how such systems can be built using a novel approach, i.e., using long running transactional model concepts.

A comparison is performed between the current model and two other previous models. Based on comparing the current model with the sagas model, the same key concepts are used in both models, like the use of “compensate” procedure for sub-transactions. However, the sagas model is not appropriate to be used in the context of ERP workflow

models since it does not provide convenient semantics for allowing direct mapping of the workflow nor addresses the generated inherent complexities from the sub-business processes modelling. These two problems are solved in the current model in addition to the use of a compartmentalisation mechanism for grouping related sub-processes. Based on comparing the current work with the work proposed to associate cost for "compensate" procedures, the new model extends the feature used in the previous work and introduces "costs" for both "compensate" and "execute" procedures in the sub-transactions, where this allows using the model within a simulation framework.

9.3 SUMMARY

The thesis is presented and divided to nine chapters as summarised as follows:

Chapter 1. Provided an overview of ERP systems has been provided, outlined the motivation, scope and methodology of the present study, set out the research questions and specified the main criteria of success.

Chapter 2. Provided a background about the transactions and the main principles of Enterprise Resource Planning (ERP), long Running Transactions (LRT) and Communication Closed Layers (CCLs), to help in performing the required work.

Chapter 3. Reviewed some of the published work about Enterprise Resource Planning (ERP) systems based on introducing the use of ERP systems in companies, effects of man-machine interaction factors on ERP Systems, user acceptance of ERP in knowledge-based industries and ERP cloud.

Chapter 4. Presented a review ofsome of the published works about both Long Running Transactions (LRTs) and Communication Closed Layers (CCLs) as well as the application of Layers in LRTs and LRTs in ERP systems.

Chapter 5. The introduced modifications for CCLs and described some of the processes used for choices and iterations in the local environment. Then, analysed the main model of CCLs using the programming language Occam.

Chapter 6. The conducted methodology and design approach have been presented based on exploring the main two challenges that have been solved in this thesis.

Chapters 7. A computational model of an Enterprise Resource Planning (ERP) workflow and a major case study, along with the core model implementation was described in detail.

Chapter 8. An evaluation of the implemented computational model performance has been explored using three scenarios and a comparison has been conducted between the current model and two previous models.

Chapter 9. The thesis has been concluded by summarising the work and providing an evaluation, making recommendations for future research and the improvement of ERP systems, and discussing several ways in which the present study might be enhanced and developed.

9.4 CONTRIBUTION TO KNOWLEDGE

The purpose of this work is to utilise Long-Running Transaction (LRT) concepts in implementing ERP workflow systems. Thus, a number of challenges were identified. This thesis has made contributions through addressing these challenges, with both the devising of the theoretical underpinnings of the core transactional model and through the implementation of the computational model of ERP workflow systems and its associated simulation framework. The following is a brief description of each of the key contributions of the thesis.

- **The Core Model**

Earlier, it was identified that the traditional transactions process methodologies were not ideally suited to be applied within ERP workflow systems. This was due to some of the physical and practical aspects of an enterprise that cannot be mapped to traditional transaction processing concepts. For example, some sub-business processes that are represented within the ERP workflow system may represent physical processes that cannot be undone or rolledback. Firstly, a novel long running transaction model specifically proposed to address these issues, through the categorisation of activities into different types. Secondly, through the use of CCL concepts, potential complexity issues that arise in the development and verification of ERP systems were addressed. This was done with the introduction of a compartmentalisation mechanism into the model, where related sub-business processes (termed Activities in the new model) are grouped together into constructs called processes. These sub-business processes can only interact with the sub-business processes with their respective process.

- **Configurable Computational Model/Simulation Framework of ERP Workflow**

For the purposes of the evaluation, a case study was modelled and implemented using the proposed model. Instead of implementing a dummy transactional model which would have only demonstrated its correctness, a computational model of an ERP workflow system was constructed to simulate a real ERP workflow system. In addition, to enable getting more understanding of the use of ERP systems in enterprise environments, reconfigure-ability was built into the system. This allows the computational model/simulation framework to be setup with various business scenarios with use of activity configuration file concept, where parameters for sub-business processes can be changed through editing a simple text file.

- **Implementation of a Realistic Case-Study using the Modelling Framework**

To demonstrate the viability of the proposed model, a real case study of a workflow from a simple manufacturing organisation was formulated using the model and then implemented in the developed computational model/simulation framework. In the case study, many of the otherwise problematic (in the traditional transactional models) concepts such as rollbacks and check-pointing were implemented with their new semantics without any issue.

- **Business Scenario execution using the Framework and data analysis**

Finally, a number of realistic business scenarios (different reconfiguration of the case study) were configured and executed using the ERP computational model/simulation framework. Through the use of a logging mechanism built into the simulation framework, the data for all the scenarios were analysed and compared with one another to understand the various effects on the service time of the customer service delivery time due to failures/delays at strategic points of the workflow. Through these simple example scenarios, the thesis has demonstrated the usefulness of such ERP computational model/simulation framework in any enterprise setting where important future decisions need to be made with many hundreds or parameters and numerous variables.

9.5 FUTURE WORK

Some of the work that can be performed in the future to enhance the system proposed in this thesis are the following:

- The use of a Workflow Control Engine (WCE) to execute the distributed workflows based on their distribution. The main problem here is the ability to run workflows on servers when applications are modelled as workflows. This requires that users must access servers to interact with the requisite applications. Practically, the convenience of users' applications must be high in context aware environments despite mobility. On the other hand, a communication link among users and servers can fail or in other cases become unusable due to the wireless communication restrictions. This in turn causes accessibility issues that can be resolved by centralised servers or completely distributed approaches. Another problem is that the configuration of any environment cannot be foreseen prior to execution, due to the highly dynamic nature of computational environments. Thus, the WCE must permit several reconfigurations to adapt the workflow distribution to the requisite state to facilitate new workflow placements and fragmentations. When the actual context state violates the current workflow distribution assumptions, adaptations can be triggered. However, in cases when the adaptation runs after a violation, a reactive approach is then followed.

- Development of a system for the proposed system workflow to permit proofing of several security, timing, context and functional workflow characteristics. In addition, the development of a background equivalence theory for confirmed characteristics, that is applicable contextually.

- The use of a formal programming language in order to facilitate expressing and proving of the required workflows and workflow characteristics. The required programming language must be significant to supply functional, mobility, performance, timing and context characteristics. In addition, there is a need to explore the hybrid nature of the workflow as related to performance characteristics. This can be performed based on temporal logic, which in turn depends on using mechanisms that can be readily integrated.

REFERENCES

[1] Davenport, T.H., 'Putting the enterprise into the enterprise system', *Harvard Bus. Rev.*, Vol. 76, No. 4, pp. 121-131, 1998.

[2] Dewett, T. and Jones, G.R., 'The role of information technology in the organisation: A review, model and assessment', *J. Manage*, Vol. 27, No. 3, pp. 313-346, 2001.

[3] Al-Mashari, M., Al-Mudimigh, A. and Zairi, M., 'ERP: taxonomy of critical factors', *European Journal of Operational Research*, Vol. 146, pp. 352-364, 2003.

[4] Mabert, V.A., Soni, A. and Menkataramanan, M.A., 'The impact of organisation size on ERP implementations in the US manufacturing sector', *Omega*. Vol. 37, pp.235-246, 2003.

[5] Hunton, J.E, Lippincott, B and Reck, J.L. 'ERP systems: comparing firm performance of adopters and no adopters', *International Journal of Accounting Information Systems*, Vol. 4, pp. 165-184, 2003.

[6] Umble, E., Haft, R.R. and Umble, M.M., 'ERP: Implementation procedures and critical success factors', *European Journal of Operational Research,* Vol. 146, pp. 241-257, 2003.

[7] Livari, J., 'The organisational fit of information systems', *Journal of Information systems,* Vol. 2, pp. 3-29, 1992.

[8] Kanellis, P., Lycett, M. and Paul, R.J., 'Evaluating business information systems fit: from concept to practical application', *European Journal of Information Systems,* Vol. 8, pp. 65-76, 1999.

[9] Dechow, D. and Mouritsen, J, 'ERP systems, management control and the quest for integration', *Accounting, Organisations and Society,* Vol. 30, pp.691-733, 2005.

[10] Genoulaz, V.B. and Millet, P.A., 'A classification for better use of ERP systems, management control and the quest for integration', *Accounting, Organisations and Society,* Vol. 54, pp. 573-587, 2005.

[11] Ko, D.G., Kirsch, L. and King, E.R, 'Antecedents of Knowledge transfer from consultants to clients in enterprise system implementations', *MIS Quart,* Vol. 29, No. 1, pp. 59-85, 2005.

[12] Skibniewski, M.J. and Ghosh, S., 'Determination of key performance indicators with ERP systems in engineering construction firms', *J. Constr. Eng. Manage,* Vol. 135, No.10, pp.965-978, 2009.

[13] Chung, B. Y., Skibniewski, M. J., Lucas, H. C. and Kwak, Y. H., 'Analyzing ERP system implementation success factors in the engineering-construction industry', *J. Comput. Civ. Eng,* Vol. 22, No.6, pp. 373-382, 2008.

[14] Tatari, O., Lacouture, D.C. and Skibniewski, M.J., 'Current state of construction enterprise information systems: Survey research', *Const. Innov,* Vol. 7, No.4, pp.310-319, 2007.

[15] Anru, F., Yi-Jun, L., Ben-Nan, S. and Qiang, Y., 'The Process of ERP Usage in Manufacturing Firms in China: An Empirical Investigation', *the16th International Conference on Management Science & Engineering,* 2009.

[16] Bryson, K. and Sullivan, W., 'Designing Effective Incentive-Oriented Outsourcing Contracts for ERP Systems', *Proceedings of the 35th Annual Hawaii International Conference on System Sciences,* 2002.

[17] Park, J. and Hossain, L., 'Social-embed-ness of ERP Systems in KM Practice', *IEEE,* 2003.

[18] Nonaka, J. and Takeuchi, H., 'The knowledge-creating company: How Japanese companies create the dynamics of innovation', *New York Oxford University Press,* 1995.

[19] Li, W. and Peng, L., 'Upgrade ERP from C/S to B/S Based on Web Service', *IEEE,* pp. 593-597, 2005.

[20] Lanotte, R., Maggiolo–Schettini, A., Milazzo, P. and Troina, A., 'Design and Verification of Long–Running Transactions in a Timed Framework', *Science of Computer Programming*, 2008.

[21] Garcia–Molina, H. and Salem, K., 'Sagas', *ACM, in: Proc. SIGMOD'87*, pp 249–259, 1987.

[22] Houston, M.C., Little, I. Robinson, Shrivastava, S. K. and Wheater, S. M., 'The CORBA Activity Service Framework for Supporting Extended Transactions', *Software – Practice and Experience,* Vol. 33, pp 351–373, 2003.

[23] Laneve, C. and Zavattaro, G., 'Foundations of Web Transaction', *Springer*, in: Proc. FOSSACS'05, Lecture Notes in Computer Science, Vol. 3441, pp. 282–298, 2005.

[24] Tygar, J.D., 'Atomicity in electronic commerce', *in Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing*, pp 8-26, May 1996.

[25] Chaum, D. Fiat, A. and Naor, M. 'Untraceable electronic cash. In advances in Cryptology,' CRYPTO 88 Proceedings, *Springer-Verlag*, pp 200-212, 1990.

[26] Colombo, C., and Pace, G. J., 'Recovery within Long Running Transactions', *ACM Transactions on Computational Logic*, Vol. V, No. N, pp 1–40., 2011.

[27] Nah, F.F.H, Lau, J.S.L and Kuang, J., 'Critical factors for successful implementation of enterprise systems', *Business Process Management Journal*, Vol. 7, No.3, pp. 285-296, 2001.

[28] Wang, B. and Nah, F. *ERP + e-business = a new vision for enterprise system*, pp. 147-64, 2001.

[29] Somers, T.M., Nelson, K. and Ragowsky, A., *Proceedings of the Americas Conference on Information Systems (AMCIS)*. 2000.

[30] Wees, S., *Juggling toward ERP success: keep key success factors high*, 2000.

[31] Rosario, J.G., *On the leading edge: critical success factors in ERP implementation theses*, 2000.

[32] Buckhout, S., Frey, E. and Nemec, J., 'Making ERP succeeds: turning fear into promise', *Engineering Management Review*. pp. 116-23, 1999.

[33] Rogers, E.M., 'Diffusion of innovations', *The Free Press*. New York, *4th Edition*, 1995.

[34] Marnewick, C. and Labuschagne, L., 'A conceptual model for ERP (ERP)', *Information Management & Computer Security*, Vol. 13, No.2, pp. 144-155, 2005.

[35] Beheshti, H.M., 'What managers should know about ERP / ERP II', *Management Research News*, Vol. 29, pp. 184-193, 2006.

[36] Shehab, E. M., Sharp, M. W., Supramaniam, L., Spedding, T. A., 'ERP – An integrative review', *Business Process Management*, Vol. 10, No.4, pp. 359-386, 2004.

[37] Brynjolfsson, E., 'The productivity paradox of information technology: review and assessment. Communications of the ACM', Vol. 36, No.12, pp.67-77, 1993.

[38] Roach, S., 'America's technology dilemma: A profile of the information economy. Morgan Stanley Special Economic Study', New York, USA, 1987.

[39] Hong, K.K. and Kim, Y.G, 'The critical success factors for ERP implementation: an organisational fit perspective', *Information & Management*, Vol. 40, pp.25-40, 2002.

[40] Bradford, M. and Florin, J., 'Examining the role of innovation diffusion factors on the implementation success of ERP systems', *International Journal of Accounting Information Systems*, Vol 4, pp.205-225, 2003.

[41] Wei, C., Chien, C. and Wang, M.J., 'AHP-based approach to ERP system selection', *Int. J. Production Economics*, Vol. 96, pp.47-62, 2005.

[42] Wagner, E.L. and Newell, S., 'Best' for whom? The tension between 'best practice' ERP packages and diverse epistemic cultures in a university context', *of Strategic Information Systems* Vol. 13, pp.305-328, 2004.

[43] Somers, T.M., Nelson, K. and Ragowsky, A., *Proceedings of the Americas Conference on Information Systems (AMCIS)*. 2000.

[44] Markus, M.L. and Tanis, C., *The enterprise system experience ± from adoption to success*, pp. 173- 207, 2000.

[45] Agarwal, R, And Prasad, J, 'The role of innovation characteristics and perceived voluntariness in the acceptance of information technologies', *Decision sciences*, Vol. 28, no. 3, pp. 557-582, 1997.

[46] Kerimoglu, O. and Bauoglu, N., 'Optimizing the Change Management of ERP Systems Implementations', pp. 2824-2831, 2006.

[47] Kumar, K. and Hillegersberg, J.V., 'ERP experiences and evolution, Association for Computing Machinery', *Communications of the ACM,* Vol. 43, No.4, 2000.

[48] Everdingen, Y.V., Hillegersberg, J.V. and Waarts, E., 'ERP adoption by European midsize companies, Association for Computing Machinery', *Communications of the ACM,* Vol. 43, No.4, 2000.

[49] Butler, M. J., and Ferreira, C., 'An operational semantics for stac, a language for modelling long-running business transactions', *In COORDINATION. Lecture Notes in Computer Science,* vol. 2949. Springer, Berlin, Heidelberg, pp. 87–104, 2004

[50] Lee, J.C. and Myres, M.D., 'Dominant actors, political agendas, and strategic shifts over time: a critical ethnography of an enterprise systems implementation', *Journal of Strategic Information Systems,* Vol. 13, pp.355-374, 2004.

[51] Bingi, P., Sharma, M.K. and Golda, J.K., 'Critical issues affecting an ERP implementation', *Information Systems Management*, Vol. 16, No.3, pp.7-14, 1999.

[52] Lucas, H.C., Walton, E.J. and Ginzberg, M.J, 'Implementing packaged software', *MIS Quarterly*, pp.537-549, 1988.

[53] Laughlijn, S.P., *An ERP game plan, Journal of Business Strategy*, pp.32-37, 1999.

[54] Helm, S.A., Hall, M.L.L. and Hall, C.A.L., 'Pre-implementation attitudes and organisational readiness for implementing an ERP system', *European Journal of Operational Research*, Vol. 146 pp. 258-273, 2003.

[55] Wu, J.H., Wang, Y.M., 'Measuring ERP success: The key-users' viewpoint of the ERP to produce a viable IS in the organisation', *Computers in Human Behavior,* Vol. 20, pp. 505-515, 2004.

[56] Calisir, F., 'The relation of interface usability characteristics, perceived usefulness, and perceived ease of use to end-user satisfaction with ERP systems', *Portland International Conference for Management of Engineering and Technology'06,* pp. 2335-2341, 2004.

[57] Ozen, C. and Basoglu, N., 'Impact of man-machine interaction factors on ERP software design', In *Portland International Conference for Management of Engineering and Technology'06*, pp. 2335-2341, 2006.

[58] Lim, E. T. K., Pan, S. L. and Tan, C. W, 'Managing user acceptance towards ERP system—understanding the dissonance between user expectations and managerial policies', *Eur. J. Inform. Syst*., Vol. 14, No.2, pp. 135-149, 2005.

[59] Amoako-Gyampah, K., 'ERP implementation factors: A comparison of managerial and end-user perspectives', *Bus. Process Manage. J*, Vol. 10, No.2, pp.171-183, 2004.

[60] Wang, E. T. G. and Chen, J. H. F., 'The influence of governance equilibrium on ERP thesis success', *Decis. Support Syst*., Vol. 41, No.4, pp.708-727, 2006.

[61] Bueno, S. and Salmeron, J. L., 'TAM-based success modelling in ERP', *Interact. Comput*. Vol. 20, No.6, pp. 515-523, 2008.

[62] Luo, W. and Strong, D. M., 'A framework for evaluating ERP implementation choices', *IEEE Trans. Eng. Manag*., Vol. 51, No.3, pp. 322-333, 2004.

[63] Migdadi, M., 'Knowledge management enablers and outcomes in the small-and-medium sized enterprises', *Ind. Manage*, Vol. 109, No.6, pp.840-858, 2009.

[64] Chung, B.Y, Skibniewski, M.J, Lucas, H.C and Kwak, Y.H., 'Analyzing ERP system implementation success factors in the engineering-construction industry', *J. Comput. Civ. Eng*., Vol. 22, No.6, pp. 373-382, 2008.

[65] Bagozzi, R. P., 'The legacy of the technology acceptance model and a proposal for a paradigm shift', *J. Assoc. Inform. Syst*., Vol. 35, No. 4, pp.367-388, 2007.

[66] Keegan, A. and Turner, J. R., 'The management of innovation in thesis based firms', *Long Range Plan*, Vol. 35, No.4, pp. 367-388, 2002.

[67] Lim, E.T.K., Pan S.L. and Tan, C.W., 'Managing user acceptance towards ERP system—understanding the dissonance between user expectations and managerial policies', *Eur. J.Inform. Syst*., Vol. 14, No.20, pp.135-149, 2005.

[68] Somersa, T.M. and Nelson, K.G, 'A taxonomy of players and activities across the ERP thesis life cycle', *Information & Management*, pp. 257-260, 2004.

[69] Tatari, O. D. Castro-Lacouture And Skibniewski, M. J., 'Current state of construction enterprise information systems: Survey research', *Const*, 2007.

[70] Amoako-Gyampah, K. and Salam, A.F, 'An extension of the technology acceptance model in an ERP implementation environment', *Information & Management*, Vol. 41, pp. 731-745, 2004.

[71] Davis, F. D., 'Perceived usefulness, perceived ease of use, and user acceptance of information technology', *MIS Quart*., Vol. 13, No.3.pp.319-240, 1989.

[72] Hong, W., Thong, J., Wong, W. and Tam, K., 'Determinants of user acceptance of digital libraries: An empirical examination of individual differences and system characteristics', *J. Manage. Inform. Syst*., Vol. 18, No.3, pp. 97-124, 2002.

[73] Hwang, Y., 'Investigating enterprise systems adoption: Uncertainty avoidance, intrinsic motivation, and the technology acceptance model', *Eur. J. Inform. Syst.*, Vol. 14, No.2. pp. 150-161, 2005.

[74] Igbaria, M., Zinatelli, N., Cragg, P. and Cavaye, A. L, 'Personal computing acceptance factors in small firms: A structural equation model', *MIS Quart*, Vol. 21, No.3, pp. 279-305, 1997.

[75] Lewis, W., Agarwal, R., and Sambamurthy, V., 'Sources of influence on beliefs about information technology use: An empirical study of knowledge workers*', MIS Quart*, Vol. 27, No.4, pp. 657-678, 2003.

[76] Lucas, H. C., Jr. and Spitler, V. K., 'Technology use and performance: A field study of Broker Workstations', *Decis. Sci.*, Vol. 30, No. 2, pp. 291-311, 1999.

[77] ORACLE, *SAP ERP in the cloud*, 2010.

[78] Abadi, M., Birrel, A., Harris, T. and Israd, M., 'Semantics of transactional memory and automatic mutual exclusion', *In Proc. 35th ACM SIGPLANSIGACT Symposium on Principles of Programming Languages*, pp 63–74, 2008.

[79] Israd, M. and Birrel, A., 'Automatic mutual exclusion', In *Proc. 11th Workshop on Hot Topics in Operating Systems*, 2007.

[80] Andrews, T. et al., 'Business process execution language for web services', 2003.

[81] Fischer, J. and Majumdar, R., *Ensuring Consistency in Long Running Transactions*, UCLA Computer Science Dept. Technical Report, 2007.

[82] Brumi, R., Nutler, M., Ferriera, C., Hoare, C.A.R., Melgratti, H. and Montanari, U., 'Comparing two approaches to compensable flow composition*',Proceedings from 16thInternational Conference, CONCUR 2005, San Francisco, CA, USA*, Vol 3653, pp.3653:383–397, 2005.

[83] Brumi, R. and Montanari, U., 'Theoretical foundations for compensations in flow composition languages', *ACM* , In POPL '05, pp.209–220, 2005.

[84] Butler, B. and Ferrira, C., 'An operational semantics for stac, a language for modelling long-running business transactions', *Springer*, in Coordination '04, Vol. 2949 of LNCS, pp 87–104., 2004.

[85] Gray, J. and Reuter, A., *Transaction processing Concepts and Techniques.* Morgan Kaufmann, 1993.

[86] Butler, M. and Ferrira, C. and Hoare, C.A.R., 'A trace semantics for long-Running transactions', *Springer*, In Communicating Sequential Processes: The First 25 Years, Vol. 3525/2005 of LNCS, pp 133–150, 2004.

[87] Kartz, B., *Protocols For Long Running Business Transactions*, 2004.

[88] Davies, J. r., C. T, 'Recovery semantics for a db/dc system'., *In Proceedings of the ACM annual conference, ACM '73. ACM, New York, NY, USA*, pp. 136–141, 1973

[89] Randell, B., Lee, P., and Treleaven, P. C., 'Reliability issues in computing system design', *ACM Comput. Surv*, vol. 10, pp. 123–165, 1978.

[90] Gray, J., 'The transaction concept: Virtues and limitations (invited paper)', *7th International Conference, September 9-11, 1981, Cannes, France, Proceedings. IEEE Computer Society, Los Alamitos, CA, USA*, pp. 144–154, 1981.

[91] Garcia-Molina, H., and Salem, K., 'Sagas', *In SIGMOD '87: Proceedings of the 1987 ACM SIGMOD international conference on Management of data. ACM, New York, NY, USA*, pp. 249–259, 1987

[92] Wil Janssen, Mannes Poel, and Job Zwiers., 'Action systems and action refinement in the development of parallel systems', *In Jos C. M. Baeten and Jan Frisco Groote, editors, Proceedings of CONCUR '91, 2nd International Conference on Concurrency Theory, Amsterdam, The Netherlands, volume527 of LNCS,* pp 298–316, 1991.

[93] Job Zwiers, ' Layering and action refinement for timed systems', *In J. W. de Bakker, CornelisHuizing, Willem P. de Roever, and Grzegorz Rozenberg,*

editors, Real-Time: Theory inPractice, REX Workshop, vol. 600 of LNCS, pp 687–723. Springer-Verlag, 1991.

[94] Wil Janssen and Job Zwiers, 'Specifiying and proving communication closedness in protocols', *In Andr´ e A. S. Danthine, Guy Leduc, and Pierre Wolper, editors, PSTV, volume C-16of IFIP Transactions*, pp 323–339. North-Holland, 1993.

[95] Wil Janssen, 'Layers as knowledge transitions in the design of distributed systems', *In Uffe H.Engberg, Kim G. Larsen, and Arne Skou, editors, Proceedings of the Workshop on Tools andAlgorithms for the Construction and Analysis of Systems, TACAS (Aarhus, Denmark, 19–20May, 1995), number NS-95-2 in Notes Series*, pp 304–318, Department of ComputerScience, University of Aarhus, May 1995.

[96] Wil Janssen, 'Layered Design of Parallel Systems', *PhD thesis, University of Twente*, 1994.

[97] Rob Gerth and Liuba Shrira, 'On proving communication closedness of distributed layers', *In Kesav V. Nori, editor, Foundations of Software Technology and Theoretical ComputerScience, Sixth Conference, volume 241 of LNCS*, pp 330–343, New Delhi, India, 18–20 December 1986.

[98] Elrad, T. and Krishna Kumar, N., 'The use of communication-closed layers to support imprecise scheduling for distributed, real-time programs', *Computers and Communications, 1991. Conference Proceedings., Tenth Annual International Phoenix Conference* , pp 226 – 231, 1991

[99] Zedan, H, DPE Build, chapter five: 'Reliable systems in occam from Actions in occam', *Micoprocessing and Microprogramming*, pp. 133-148, 1988.

[100] Zedan, H.,'Achieving atomicity in occam',*Micoprocessing and Microprogramming*, 23:261–265, 1988.

[101] Grefen, P, Vonk, J, Boertjes, E and Apers, P, 'Two-Layer Transaction Management for Workflow Management Applications', *European Commission in the WIDE project*, 2013

[102] Jinling, W, Beihong, J and Jing, L, 'Pessimistic Predicate/Transform Model for Long Running Business Processes', *TSINGHUA SCIENCE AND TECHNOLOGY*, vol. 10, no. 3, pp288-297, 2005

[103] Lea, B. R., Gupta, M. C. and Yu, W. B, 'A prototype multi-agent ERP system: an integrated architecture and a conceptual framework', *Technovation*, Vol. 25, no. 4, pp. 433-441, 2005.

[104] Wooldridge, M., Jennings, N., 'Intelligent agents: theory and practice', *The Knowledge Engineering Review*, vol. 10, no. 2, pp 115–152, 1995

[105] Bruni, R., Melgratti, H., and Montanari, U., 'Theoretical foundations for compensations in flow composition languages', *In POPL '05: Proceedings of the 32nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages. ACM, New York, NY, USA*, pp. 209–220, 2005

[106] Li, J., Zhu, H., Pu, G., and He, J., '*Looking into compensable transactions. Software Engineering Workshop', Annual IEEE/NASA Goddard*, pp. 154–166, 2007

[107] Laneve, C., and Zavattaro, G., 'Foundations of web transactions', *In FoSSaCS. Lecture Notes in Computer Science, Springer*, vol. 3441, pp. 282–298, 2005

[108] Molina, G.H and Salem, K, "Sagas", ACM SIGMOD Record, vol. 16, no. 3,pp. 249−259, 1987

[109] Hu, Q, Zhuang, Y, Liu, J and Liu, Y, "A Long-running Transaction Model of Workflow", 3rd International Conference on Biomedical Engineering and Informatics, 2010.

APPENDICES

APPENDICES

APPENDIX A: IMPLEMENTATION CODE

```
package erp_model.test_suit;
import erp_model.core;*.
import erp_model.core.Process;
import erp_model.case_study;*.

import java.util;*.

class Simple_Request extends Request}

        String job;

        public Simple_Request(String s}(
                job = s;
        {

{

public class Test_Bench
{       public static void main(String[] args}(

                Activity act1 = new Order_Material_Activity;()
```

```
                Activity act2 = new Payment_Activity;()
                Activity act3 = new Manufacturing_Activity;()
                Activity act4 = new Supplier_Delivery_Activity;()
                Schedule_Entry s1, s2, s3, s4;
                s1 = new Schedule_Entry(  act1;(
                s2 = new Schedule_Entry(  act2;(
                s3 = new Schedule_Entry( act3;(
                s4 = new Schedule_Entry( act4;(

                List<Schedule_Entry> lst = new LinkedList<Schedule_Entry;()<
                lst.add(s1;(
                lst.add(s2;(
                lst.add(s3;(
                lst.add(s4;(

                Process p = new Process(new Schedule(lst;((
                Event_Generator e = new Event_Generator(1.0/12.0, p.get_queue;(()

                new Thread(e).start;()
                new Thread(p).start;()
                {

}
package erp_model.core;
```

```
import java.io.FileWriter;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.Queue;

import java.util.concurrent.ArrayBlockingQueue;

import java.util.ListIterator;


public class Process implements Runnable}


        private Schedule                                    schedule;

        private Logger<Activity_Response>   activity_logger;

        private ArrayBlockingQueue<Activity_Request>
        request_queue;

        private PrintWriter pw;

        private final String process_log_dir = "Logs;"/

        private final String process_log_file= "process_log.txt;"



        public Process(Schedule s}(

                schedule                = s;

                activity_logger  = new Logger<Activity_Response;()<

                request_queue           =                                   new
ArrayBlockingQueue<Activity_Request>(20;(


                try}

                        pw              =           new             PrintWriter(new
FileWriter(process_log_dir+process_log_file;((
```

```
            {catch (IOException e} (
                    //TODO Auto-generated catch block
                   e.printStackTrace;()
            {

{

public void queue_request(Activity_Request req}(
            request_queue.add(req;(
{

public Queue get_queue}()

            return request_queue;
{

public void run}()
            Activity_Request req = null;

            while(true}(
             */check the queue to see whether there are request/*

            try}
                    */dequeue the first/*
```

```
                req = request_queue.take;()
                req.set_service_time(System.currentTimeMillis());  /*  service
start/*


            {catch (InterruptedException e} (
                 //TODO Auto-generated catch block
                e.printStackTrace;()
            {

            System.out.println("**** Transaction for request id: "+req.request_id+"
START;(" ***

            ListIterator<Schedule_Entry>  itr  =  schedule.get_list().listIterator ;()

             */as long as execute method succeeds move forward through the
             * list of scheduled activities
             *
            /*
            Schedule_Entry ent;
                while(itr.hasNext }(()


                ent= itr.next;()
                 */if one fails, move backwards through the schedule using the
                 * activity log until a activity of type check-point is
                 * detected
                /*
```

```
Activity_Response resp ;
if(( resp = ent.get_activity().execute(req)).get_status}(()
        */  activity executed successfully/*
        activity_logger.log(resp;(

{  else{ /*activity failed/*
         */ pass the undo method the activity response from the log/*
        undo_until_checkpoint(itr, resp;(

{

 */
 * we we reach the end of list of scheduled activities the
 * then add a log to the process logger and
 *
 * initiate the call-back to the requester
 *
/*
{

req.set_service_end_time(System.currentTimeMillis());  /* transaction completed/*
System.out.println("****  Transaction  for  request  id: "+req.request_id+" END;(" ***

log_transaction(req;(
```

```
        {


        {
         */this functional will roll-back activities until it finds a check-point or a
         *       critical activity
        /*
        private void undo_until_checkpoint(ListIterator<Schedule_Entry> it,
Activity_Response r}(


                Schedule_Entry ent;
                it.previous(); /* skip the failed activity/*
                while (it.hasPrevious&& ()
                                )  )!ent  =  it.previous()).get_activity().get_type()  ==
Activity_Type.CRITICAL
                                        ||ent.get_activity().get_type()          ==
Activity_Type.CHECK_POINT((
                                }


                        ent.get_activity().undo(activity_logger.undo_last;(()
                {
                it.next;()
        {


        private void log_transaction(Activity_Request r}(


                pw.print(r.request_id +"\t;("
                pw.print(r.get_quantity()+"\t;("
```

```
                pw.print(r.get_start_time()+"\t;("

                pw.print(r.get_service_time()+"\t;("

                pw.print(r.get_service_end_time()+"\t;("

                pw.print((r.get_service_end_time()  -  r.get_service_time())+"\t");  /*
service duration/*

                pw.print((r.get_service_end_time() - r.get_start_time())+"\t\n");  /* entire
duration including the

                                                queueing delay/*


                pw.flush;()


        {



{
package erp_model.core;


import java.util.Queue;


import erp_model.core.math;*.



 */we are simulating 1 hour in real time as 1000 milliseconds, and we will

 * change the lambda to simulate different rates of arrivals of new events

 *

 * This class will create request events at intervals exponentially distributed
```

```
 * in relation to given lambda
 *
 * It will also create a random request quantity between the provided max and min
 * quntity
   *
/*

public class Event_Generator implements Runnable}

        private double lambda = 1.0/12.00; /* 1 divided by the arrival mean-time, default
is
                                                                        12  hour
arrival mean/*
        private final int REQUEST_NUM = 50;
        private int max_q_range;
        private int min_q_range;
        private int request_count = 1;
        private Queue<Activity_Request> queue;

        public Event_Generator(double lambda,Queue queue}(

                this.lambda = lambda;
                this.queue  = queue;
                this.max_q_range = 20;
                this.min_q_range = 10;
```

```
        {

public Event_Generator(double lambda,Queue queue, int max_q, int min_q}(

                this.lambda = lambda;
                this.queue  = queue;
                this.max_q_range = max_q;
                this.min_q_range = min_q;

        {



        public void set_lambda(double l}(
                lambda = l;

        {

        public void run}()
                Activity_Request req;
                 */generate the specified number of requests/*
                for(int i=0; i< REQUEST_NUM; i}(++

                        long start = System.currentTimeMillis;()
```

```
int quantity = Number_Gen.uniform(min_q_range, max_q_range+1;(

*/create a new quest/*

req = new Activity_Request(i, "Order Number "+i, start, quantity;(

*/generate the arrival interval/*

long interval = Math.round(Number_Gen.exp(lambda) * 1000); /* multiplied by 1000 to

convert into milliseconds/*

this.queue.add(req;(

*/sleep for the arrival interval/*

try}

Thread.sleep(interval;(

{catch (InterruptedException e} (

//TODO Auto-generated catch block

e.printStackTrace;()

{
```

```
                {


        {


{
package erp_model.core;

import java.util.*;


publicclass Schedule{

        private List<Schedule_Entry> entries;
        private ListIterator<Schedule_Entry> itr;

        public Schedule(List<Schedule_Entry> lst){

                entries = lst;
                itr     = entries.listIterator();

        }

        publicboolean  has_next(){
                return itr.hasNext();
        }

        public ListIterator<Schedule_Entry> get_iterator(){

                return itr;

        }

        public List<Schedule_Entry> get_list(){
                return entries;
        }


}

package erp_model.core;
```

```
publicclass Schedule_Entry {

        private Activity activity;


        public Schedule_Entry(Activity act){

                activity = act;
        }

        public Activity get_activity(){
                return activity;
        }

}
package erp_model.core;


import java.io;*.

import java.util.StringTokenizer;


import erp_model.core.math;*.



public abstract class Activity}


        private Activity_Type type;

        private BufferedReader br;


        private int failure_probability; /*as a percentage within range of 0 - 100 /*

        private int do_work_time;                /* simulated time delay for do work/*

        private int compansate_time;   /* simulated time delay for failure/*
```

```
        private int capacity;                   /* this a generic capacity for each activity
                                                                                interpretation
depends on Activity
                                                                                definition
and implementation
                                                                        /*
        private final String config_dir = "Activity_Configurations;"/

        protected abstract boolean do_work;()
        protected abstract boolean compansate(Activity_Response r;(

        public synchronized boolean allocate_capacity(int n}(
                if (n <= capacity} (
                        capacity -=n;
                        return true;
                {
                else return false;

        {

        public synchronized void deallocate_capacity(int n}(

                capacity+=n;
        {

        public Activity_Type get_type}()
```

```
        return type;
    {

    public void set_type(Activity_Type t}(
        type = t;
    {

    protected boolean calculate_failure_probability}()

        if (failure_probability == 0) return true;
        else if (failure_probability == 100) return false;

        else
            return !(Number_Gen.uniform(101) <= failure_probability;(
    {

    protected void read_configurations}()

        try}

            br = new BufferedReader(new
FileReader(config_dir+this.getClass().getSimpleName()+".txt;(("
        //  pw = new PrintWriter(new FileWriter(out;((

            String  line;""=
```

```
                StringTokenizer tk;

                line = br.readLine(); /* failure probablity/*

                tk = new StringTokenizer(line, "-"); tk.nextToken;()

this.set_failure_probability(Integer.parseInt(tk.nextToken().trim;(((()

                line = br.readLine(); /* do work time delay/*

                tk = new StringTokenizer(line, "-"); tk.nextToken;()
                this.set_do_work_time(Integer.parseInt(tk.nextToken().trim;(((()

                line = br.readLine(); /* Compensate time delay/*

                tk = new StringTokenizer(line, "-"); tk.nextToken;()

this.set_compansate_time(Integer.parseInt(tk.nextToken().trim;(((()

                line = br.readLine(); /* capacity/*

                tk = new StringTokenizer(line, "-"); tk.nextToken;()
                this.set_capacity(Integer.parseInt(tk.nextToken().trim;(((()


        {catch (FileNotFoundException e} (
                 //TODO Auto-generated catch block
                e.printStackTrace;()
```

```
                {catch( IOException e}(

                        e.printStackTrace;()
                {
        {

        public Activity_Response execute(Activity_Request req}(

                Activity_Response r = new Activity_Response(req.toString()+ "done;("

                if(do_work}(()
                        try}
                                Thread.sleep(this.do_work_time * 1000;(
                         {catch (InterruptedException e} (
                                 //TODO Auto-generated catch block
                                e.printStackTrace;()
                        {
                        r.set_status(true;(
                        System.out.println("Subtransaction:  SUCCESS");  /*  successful
execution/*
                {
                else{ r.set_status(false;(
                        try}
                                Thread.sleep(this.do_work_time * 1000;(
                         {catch (InterruptedException e} (
                                 //TODO Auto-generated catch block
```

```
                    e.printStackTrace;()
                {
                System.out.println("Subtransaction: FAILURE;("

        {

        return r;


{
public void undo(Activity_Response res}(

        compansate(res;(
{
public int get_failure_probability} ()

        return failure_probability;
{

public void set_failure_probability(int failure_probability} (

        if (failure_probability < 0 || failure_probability > 100(
                return;
        this.failure_probability = failure_probability;
{
```

```
public int get_do_work_time} ()

        return do_work_time;
{

public void set_do_work_time(int do_work_time} (

        this.do_work_time = do_work_time;
{

public int get_compansate_time} ()

        return compansate_time;
{
public void set_compansate_time(int compansate_time} (

        this.compansate_time = compansate_time;
{

public void set_capacity(int c}(
        capacity = c;
{

public int get_capacity}()
        return capacity;
```

```
        {



{
package erp_model.core;

publicclass  Activity_Request<E>extends Request{

        private String request;

        public Activity_Request( int request_id, String s, long start, int quantity ){
                request = s;
                super.request_id = request_id;
                super.set_quantity(quantity);
                super.set_start_time(start);
        }

        public String toString(){
                returnthis.request_id+", \t"+ this.get_quantity()
                                                        +", \t"+ this.get_start_time()
                                                        +", \t"+ this.get_service_time()
                                                        +",                                     \t"+
this.get_service_end_time();
        }

}
package erp_model.core;

publicclass  Activity_Response<E>extends Response{

        private String request;
        privateboolean status;

        public Activity_Response(String s){
                request = s;
                status  = false;
        }

        publicboolean get_status(){
                return status;
        }
```

```
	publicvoid set_status(boolean s){
		status = s;
	}

	public String toString(){
		return request;
	}
}
```

```
package erp_model.case_study;


import erp_model.core.Activity;

import erp_model.core.Activity_Response;


public class Manufacturing_Activity extends Activity}


	public Manufacturing_Activity}()


		this.read_configurations;()

	{


	@Override

	protected boolean do_work} ()
		 //TODO Auto-generated method stub


		System.out.println("Manufacturing activity do work;("



		return this.calculate_failure_probability;()
```

```
        {

        @Override
        protected boolean compansate(Activity_Response r} (
                //TODO Auto-generated method stub

                System.out.println("Manufacturing activity undo work;("
                return true;
        {



{
package erp_model.case_study;

        import erp_model.core.Activity;
        import erp_model.core.Activity_Response;
        import erp_model.core.Activity_Type;

        public class Order_Material_Activity extends Activity}

                public Order_Material_Activity}()

                        set_type(Activity_Type.CRITICAL;(

                        this.read_configurations;()
```

```
                {


                @Override
                protected boolean do_work} ()
                        //TODO Auto-generated method stub


                        System.out.println("Order_Material_Activity activity do work;("



                        return this.calculate_failure_probability;()
                {


                @Override
                protected boolean compansate(Activity_Response r} (
                        //TODO Auto-generated method stub


                        System.out.println("Order_Material_Activity activity undo
work;("
                        return true;
                {
        {
                package erp_model.case_study;
import erp_model.core.*;
publicclass Payment_Activity extends Activity {
        public Payment_Activity(){
```

```
		this.read_configurations();
	}

	@Override
	protectedboolean do_work() {
		// TODO Auto-generated method stub

		System.out.println("Payment activity do work");


		returnthis.calculate_failure_probability();
	}

	@Override
	protectedboolean compansate(Activity_Response r) {
		// TODO Auto-generated method stub

		System.out.println("Payment activity undo work");
		returntrue;
	}


}
package erp_model.case_study;




import erp_model.core.Activity;

import erp_model.core.Activity_Response;

import erp_model.core.Activity_Type;


public class Supplier_Delivery_Activity  extends Activity}


	public Supplier_Delivery_Activity}()


		this.read_configurations;()
```

```
        {

        @Override
        protected boolean do_work} ()
                //TODO Auto-generated method stub

                System.out.println("Supplier_Delivery_Activityy activity do work;("

                return this.calculate_failure_probability;()
        {

        @Override
        protected boolean compansate(Activity_Response r} (
                //TODO Auto-generated method stub

                System.out.println("Supplier_Delivery_Activityy activity undo work;("
                return true;
        {
{
```