AN INTEGRATED TEST ENVIRONMENT PROCESS MODEL TO CONTROL SOFTWARE FAILURES

By

MOHAN BHEEMASENARAO

Bachelor of Mechanical Engineering

University of Mysore

Davangere, Karnataka State, India

1988

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December 2007

AN INTEGRATED TEST ENVIRONMENT

PROCESS MODEL TO CONTROL

SOFTWARE FAILURES

Thesis Approved:

Dr. M. H. Samadzadeh

_____

Advisor

Dr. G. E. Hedrick

_____

Dr. Venkatesh Sarangan

_____

Dr. A. Gordon Emslie

_____

Dean of the Graduate College

PREFACE

Software is important to virtually any business in today's world. The most challenging issue that the software industry is facing today is how to produce quality software consistently. When there are multiple applications involved in a project in an integrated environment, controlling software quality becomes more complex. Even though many methodologies have been suggested to address the issue of software quality, there is always a demand for new and innovative software testing processes and methodologies in the face of changing technologies.

Part of the objective of this thesis work was to analyze the reasons for failures in software projects. In particular, the main focus was reasons for failures during the various stages of the testing process in an Integrated Test Environment (ITE). Based on this analysis, the main thrust of this thesis work was the development of an Integrated Test Environment Process Model (ITEPM) to control software failures. The proposed model includes effective software practices that were developed based on practical industry data. This model was tested for efficacy, applicability, viability, and practicability. A software development process that utilizes the proposed model will be able to control the software failures in an Integrated Test Environment and improve the software quality.

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to my graduate advisor Dr. M. H. Samadzadeh for his continued guidance, support, and encouragement throughout my thesis work. This thesis work would not have been possible without his valuable insights and directions.

My sincere appreciation also extends to Dr. G. E. Hedrick and Dr. Venkatesh Sarangan for their advice and serving on my graduate committee.

My sincere thanks to Dr. John P. Chandler for his valuable advice during the thesis proposal presentation meeting.

Finally, my sincere thanks to my parents, my wife, and my son for their encouragement and patience throughout my studies.

TABLE OF CONTENTS

Chapter Page

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

1.1 Importance of Software and Quality

Production-quality software is an indispensable technology and has become a powerful business asset. It has zero reproduction cost, can be distributed worldwide in seconds, does not wear out or deteriorate (unless it undergoes modification), and is the most economical and flexible way to implement almost any complex function [Humphrey 02].

The most challenging aspect that the software industry is facing today is how to deliver defect-free quality software. Software quality is defined as conformance to explicitly-stated functional and performance requirements, explicitly-documented development standards, and implicit characteristics that are expected of all professionally developed software [Pressman 05].

1.2 Importance of Software Testing

Software testing is a critical part of the software development process and impacts the delivery of high quality software [Eickelmann and Richardson 96]. The Standish Group, an IT consulting firm, reported that 65 percent of software projects started in 2006 were failures, meaning they were either not completed on time, on budget, or meet user

requirements. This included 19 percent of the projects that were outright failures, meaning that they were abandoned without implementation [Rubinstein 07]. Failing to understand and manage risks can lead to project failure, a costly problem that has not been completely addressed in almost three decades since such outcomes were first described in the literature [Wallace and Keil 04].

The rest of this thesis is organized as follows: Chapter II provides information about the importance of an Integrated Test Environment (ITE). Chapter III discusses reasons for software project failures in an Integrated Test Environment. Chapter IV describes the design and development of an Integrated Test Environment Process Model (ITEPM). Chapter V includes analysis of practical data from various software projects involving an Integrated Test Environment. Chapter VI contains the implementation and testing of an Integrated Test Environment Process Model with test plans, desk checks, and test cases. Chapter VII provides a summary, conclusion, and suggestions for future work on an Integrated Test Environment Process Model.

CHAPTER II

INTEGRATED TEST ENVIRONMENT (ITE)

An Integrated Test Environment validates functionality across multiple systems. ITEs provide a means to integrate different applications from different software technologies and provides for the end-to-end flow of a product. When testing involves an Integrated Test Environment, it becomes more complex and challenging to control the quality of the resulting software. The input runs through the different applications in the ITE and undergoes various stages of validations in each application before coming out as the expected defect-free end-product. Since ITEs involve many major applications in the flow, failure of one application may impact the entire chain resulting in revenue losses, taking more time to analyze the failures, and hampering the growth of the business. So it is critical for businesses to address these failures.

A number of applications may work well individually but when they are put together in an ITE, the end-product might fail. Data can be lost across an interface, one module can have an inadvertent adverse affect on another, sub-functions when combined may not produce the desired major functions, individually acceptable imprecision may be magnified to unacceptable levels, global data structures can present problems, etc. [Pressman 05].

Studies strongly suggest that formal project management practices have the power to reduce software project risks. The value of such practices lies largely in the well-defined patterns and directives they create for coordinating interactions and integrating inputs from various project constituents. Formal milestones also help in monitoring progress and spotting discrepancies throughout the project trajectory [Tiwana and Keil 04].

CHAPTER III

REASONS FOR SOFTWARE PROJECT FAILURES IN AN ITE

This chapter discusses the various reasons for software project failures in an ITE, namely planning failures, requirement failures, process failures, technology failures, management failures, and uncontrolled factors which are described in the following sections.

3.1 Planning Failures

Planning is an important phase in software projects that defines various tasks involved in the project and outlines the project schedule. The project plan is developed at the beginning of a software development undertaking and is continually refined and improved as the project progresses. It can be useful to the management as a framework for rigorous review and it can play a significant role in the process of developing quality software. The following three items constitute the common planning failures in an ITE.

- Unrealistic Schedules [Humphrey 02]: Unrealistic release schedules without any rational plan are bound to fail and will result in defective software. This may be due to factors such as poor project planning, underestimation of tasks, ineffective project head, and poor communication.

- Out-of-Sync Release Schedules and Code Delivery Dates: Testing process will have to wait until all applications become available, resulting in missing the schedules.
- No Dedicated ITE: Some applications, though required in an ITE, are not even in the ITE since they do not have a test environment. These applications are tested independently but not as a part of the ITE. This could be risky in the production phase.

3.2 Requirement Failures

To begin any software project, it is essential to have a set of Business Requirements. Business Requirements are transformed into Business Related Documents (BRD) based on which the System Related Documents (SRD) is developed. These documents must be finalized and accepted by all applications in an ITE. The following four items constitute the common requirement failures in an ITE.

- Misunderstanding of Requirements [Wallace and Keil 04]: Requirements define what the product should be and how it should perform. Thus, a clear understanding of the requirements could be responsible for producing a good product outcome, even as problems with process outcome persist.
- Incorrect Requirements: Giving incorrect requirements to the applications will result in defective products delivery to the market and can adversely impact business prospects.

- Frequent Requirement Changes: Such changes impact the flow across an ITE since applications not impacted by requirement changes have to wait until applications impacted by these are modified to complete their implementation, resulting in delays in the final implementation across the ITE. While the requirements or objectives normally change during the early phases, there is a point beyond which changes will do more harm than good [Humphrey 02].
- Requirements Conflict Among Applications: Requirements for one application may conflict with the functionality of some other application in an ITE, thus causing failures.

3.3 Process Failures

The end-to-end testing process has to be finalized and accepted by all applications in an ITE. Individual applications have to make sure that this will not conflict with their own testing process. If the end-to-end testing process is not followed, it may result in the following failures.

- Poor Quality: The poor quality work of one application may result in failure across the entire ITE.
- Data Configuration Mismatch across ITE: Given two communicating applications, data found in one application is not found in the other application, thus resulting in a failure.
- Data Sharing Issues: Many testing groups share the data generated from one instance of an application in an ITE thus causing duplicate, redundant, and conflicting data issues across the ITE.

- Lack of Progress Tracking System: This could be critical in any software project. When there is no tracking system to monitor the progress of each application and mitigation strategies to avoid failures, the projects will generally fail.

3.4 Technology Failures

Applications in an ITE may be operating on different software technologies due to various business needs. It is essential to have proper software interfaces clearly defined to avoid the following failures.

- Software Technology Configuration Mismatch across ITE: Some applications running on the latest software technologies and some applications running on old software technologies may result in complex interface issues.
- Instance Issues across ITE: Some applications in an ITE operate on multiple instances whereas other applications operate on a single instance in an ITE, thus causing failures.

3.5 Management Failures

Top management controls all applications in an ITE. Top management should clearly define the objectives, the scope of operation, and the goals of all applications in an ITE. What follows are some of the management failures in an ITE.

- Inappropriate Staffing [Humphrey 02]**:** When management fails to provide timely, adequate, and properly trained resources for a project, the projects will generally fail.

- Lack of Coordination Among Applications in an ITE: This may be due to different applications in an ITE coming under different departments in the organization, different regions with varying working culture, etc.
- Lack of Top Management Support for the Project [Wallace and Keil 04]: Management support for a project is essential in terms of providing the required budget, resources, motivation, and encouragement.

3.6 Uncontrolled Factors

Software projects may fail due to many uncontrolled factors including factors such as turn-over of the skilled people, organizational restructuring resulting in the shifting of people, a product becoming obsolete, the software technology utilized becoming obsolete, organizational changes due to acquisitions resulting in business changes, changes in business due to governmental regulations, and so on. Since these factors are difficult to predict, top management must have mitigation strategies in place to counter these factors in the event they occur.

CHAPTER IV

DESIGN AND DEVELOPMENT OF AN INTEGRATED TEST ENVIRONMENT PROCESS MODEL (ITEPM)

An Integrated Test Environment Process Model (ITEPM) consists of the following models to address various failures in an ITE such as planning failures, requirement failures, process failures, technology failures, and management failures.

- Planning Model
- Requirements Model
- Process Model
- Technology Model
- Management Model
- ITEPM Team Model

These models establish a technical and management framework for applying methods, tools, and people to the software development task. Also, these models identify roles, specify tasks, establish measures, and provide input and output criteria for the major failures impacted. Well-defined process models help in identifying the problem areas and suggest various methods for improvement [Humphrey 05a].

When new or improved processes are developed, these models can be reused or modified and incorporated into the process models. These models provide a solid foundation for software process improvement in an ITE and help to improve software quality.

4.1 Planning Model

The template for the planning model is given below.

TEMPLATE FOR PLANNING MODEL	
Purpose	Model to prevent planning failures in an ITE
Input criteria	List of common planning failures in an ITE • Unrealistic schedules • Out-of-sync release schedules and code delivery dates • No dedicated ITE
Input source	Planning failures in all applications in an Integrated Test Environment
Factors	
Release schedule	
Capacity planning and tracking	
Level of effort	
Release listing report	
Release development plan	
Status reporting	
High level risk assessment	
Environment setup plan	
Output criteria	• Release schedules of all applications in an ITE is documented • Capacity model completed • Level of effort estimation completed

	• Release listing report ready for distribution • Release development plan ready for distribution • Milestone dates reported • Risk assessment review completed • Environment setup plan completed

4.1.1 Planning Model Factors

The factors considered in the planning model are described in this subsection.

Release Schedule:

Release management develops a schedule for an ITE testing project. The schedule is discussed with all impacted applications in an ITE to ensure alignment with the delivery of the release to production in support of the end-to-end testing process. All the subordinate schedules developed by different application testing groups must be based on the major milestones included in the release management calendar.

Capacity Planning and Tracking:

Capacity of resources is tracked throughout an ITE testing lifecycle. Capacity planning includes manpower planning and any software and hardware tools required for the project. Each application in an ITE maintains its individual capacity roster. The capacity model translates the roster of a team by skill and function into hours available for future testing work.

Level of Effort (LOE):

Level of effort is an estimation of number of hours required to complete a software testing lifecycle in an ITE. All applications must use their Level of Effort model process

to estimate the time required for testing applications in an ITE. To estimate the number of hours required to test an application, the complexity of the testing must be determined. The complexity level depends on the type of application and also the number of changes required to be made to the existing data. Different applications may be based on different software technologies in an ITE, so each application must create its own estimated Level of Effort model. The LOE model consists of a matrix of software components and modules comprising the application, and an estimate of the number of hours needed to complete the given task.

Release Listing Report:

This report lists the number of releases of all applications in an ITE throughout the year. This report will ensure that all applications in an ITE are congruent with the release schedules well in advance.

Release Development Plan (RDP):

This plan is developed for each release by all applications in an ITE. It includes application-specific plans, schedules, resources, status, risks, and the software quality assurance functions [Humphrey 05b]. If new requirements are to be introduced into a release, the impacted applications in an ITE will be notified to update their respective plans.

Status Reporting:

Status reports help in periodically reviewing the status of an ITE project and assess progress. Frequent meetings are conducted to review the progress. Information reviewed at the meetings is captured in the various status reports, ranging from the application

level to the ITE level tracked by release management. Completion of milestone dates throughout the course of an ITE testing release is also tracked.

High Level Risk Assessment:

The purpose of risk management during the planning phase is to identify any possible events that would have adverse effect on the successful completion of the software project [Tiwana and Keil 04]. Risk assessment has to be done at the application level as well as at the ITE level. Risk status may be reviewed on a weekly basis. Risk management has to be done continuously throughout the lifecycle of the software project.

Environment Setup Plan:

This plan includes all applications impacted by the release including hardware, software, servers, and network configuration. The plan may also contain the availability of all applications impacted in an ITE. The plan also outlines the down-time required for the applications during the environment setup phase.

4.2 Requirements Model

The template for the requirements model is given below.

TEMPLATE FOR REQUIREMENTS MODEL	
Purpose	Model to prevent requirement failures in an ITE
Input criteria	List of common failures in an ITE • Misunderstanding of requirements • Incorrect requirements • Frequent requirement changes • Requirements conflict among applications

Input source	Requirement failures in all applications in an Integrated Test Environment

Factors	
Receiving requirements	
Reviewing requirements	
Candidate listing	
Analyzing requirements	
Requirement modifications	
Output criteria	• Requirements of all applications in an ITE received • Requirements reviews completed • Candidate listing finalized • Requirements analysis completed • Requirements modifications documented • Matrix to trace the requirements completed

4.2.1 Requirement Model Factors

The factors considered in the requirements model are described in this subsection.

Receiving Requirements:

A work request (WR) is a business client's request for development of a new product or the client's requested modifications to an existing product. A work request is composed of requirements that define the development and testing work to be done by the organization. Complete and accurate requirements are to be collected before starting the work. Requirements can be from internal sources or from clients in the organization.

Reviewing Requirements:

The release management group described in the planning model reviews all work requests and assigns a date by which all applications in the ITE must perform an impact assessment. The assessment will determine whether or not the new work requests impact the different applications that exist in an ITE. The information is documented by each work request as part of the project profile. After it is determined that the application is impacted by a work request, a separate project profile is created. The order of magnitude (OOM) [Murthy 07], which estimates the needed capacity to complete the work request based on the information available in the project profile, is recorded in the LOE. Finally, authorization may be needed by the release development group to proceed with the work requests.

Candidate List:

A candidate list basically contains the number of ITE work requests approved by the release management group after prioritization. The candidate list may be distributed to the impacted ITE application groups. If there are changes in the requirements, the candidate list may have to be reevaluated. All the application groups impacted by the changes may have to be notified of the changes in the requirements and changes in the candidate list.

Requirement Analysis:

The detailed client requirements based on the candidate list are captured and may be stored in a requirements database. The key business objectives, business function, client traceability, dependencies among application groups, and contact information may be

recorded in the requirements database. A team meeting has to be conducted with all application groups in the ITE to analyze the requirements documents.

Requirement Modifications:

Whenever a client submits changes to the requirements, the ITE testing team will be notified and the impact to the ITE testing schedule needs to be analyzed. The modifications and the impacted applications are reviewed by the ITE testing team. Once the applications concur with the modifications, the requirements documents are updated.

Requirement Traceability Matrix:

Business requirements are transformed into Business Related Document (BRD), based on which the System Related Document (SRD) is developed. The application groups use the SRD and the general design document to create the detailed design in order to ensure traceability to the original work request. The Traceability matrix is utilized by the test teams to map test cases and the test results to the requirements.

4.3 Process Model

The template for the process model is given below.

TEMPLATE FOR PROCESS MODEL	
Purpose	Model to prevent process failures in an ITE
Input criteria	List of common process failures in an ITE • Poor quality • Data configuration mismatch • Data sharing issues • Lack of tracking system
Input Source	Process failures in all applications in an Integrated Test

	Environment
Factors	
ITE test plan and test cases	
Application readiness	
Interface readiness	
Data setup	
Test execution	
Verification and validation	
Test status tracking	
Software quality assurance	
Reviews and audits	
Quality assurance reporting	
Quality assurance training	
Output criteria	• ITE testing procedure documented • Readiness of ITE is checked • Data setup for testing across all applications in ITE • Tracking system established • Quality policy is setup

4.3.1 Process Model Factors

The process model factors are described in this subsection.

ITE test plans and test cases:

The ITE test plan containing the details of the testing process, test approach, and test methodology has to be designed as a part of the ITE testing process. The set of test cases can be derived from the test plan. The number of test cases depends upon the input parameters for which the software has to be tested.

Application Readiness:

All applications in an ITE have be tested in conformance to the system requirements and kept ready before the integrated testing is started. Application testing is to test the logical paths of the code that have been added, modified, or deleted. Application tests ensure that the proper changes were made to the code. The application group executes the unit test stage to test the code they created. The application groups analyze the problems as they surface, making code changes as needed.

Interface Readiness:

Interface readiness can be checked by doing a shake-out test which ensures the connectivity of all applications in an ITE. This test will ensure that all applications are working properly before the actual integration testing begins.

Data Setup:

During the test data setup phase, the test team, in coordination with the ITE applications, has to make sure that the required data to carry out the integrated testing is made available to all applications. Also, if multiple applications share a common data, the data configuration checks have to done to avoid data mismatch across ITE systems.

Test Execution:

ITE test execution includes regression testing and progression testing. Regression testing is done to check the existing functionalities whenever a change is made to any portion of the software [Prasad 04]. Regression test is conducted to ensure that the previous code works with the new changes, and that the new functionality does not break the procedures related to the existing functionality. Regression test may also include the

testing of existing production software issues. Progression testing is the testing of new functionalities with the new software.

Verification and Validation [Pressman 05]:

Verification in an ITE involves verifying the test results for each work request requirement already defined with respect to the corresponding system related requirement. This will ensure that we are building the product right. Validation in an ITE involves verifying the test results for each work request requirement already defined with respect to the corresponding business related requirement. This will ensure that we are building the right product.

Test Status Tracking:

A test status tracking mechanism has to be established to track the test results and also for reporting purposes. A suitable software test reporting tool may be used for this purpose. The details to be reported on the tracking report may be negotiated with all applications in an ITE so that all applications will be made accountable for the testing progress. The frequency of reporting, e.g., daily, weekly or monthly, may be discussed with the release development group and others in an ITE. Status review meetings may also be conducted as part of progress tracking with other applications in an ITE. At the completion of integration testing, client traceability outputs are released for review. The accuracy and completeness of the output may be reviewed with the release management group and then forwarded to the client for final review and approval.

Software Quality Assurance (SQA):

The purpose of software quality assurance is to provide visibility to practices and processes followed by all application teams in an ITE and to ensure that the key deliverables and design articrafts meet the defined standards [Hower 07]. The SQA team is part of the business planning unit. The SQA team reports the SQA related data to the management. The SQA team performance and all release activities are subject to audits by an outside SQA team to ensure that all SQA and release-related activities are compliant with the processes described in the SQA plan and the SQA procedure document [Prasad 04]. The SQA team performs a role in establishing and improving a well-defined process for delivering high-quality software products. The SQA process is essential in planning and performing each activity associated with the software testing work request, application, or release. The key factors necessary to integrate SQA successfully into all aspects of the software development and testing are listed below.

- Use the guidelines described within this process document.
- Define and document the standards and procedures, plans, and schedules.
- Audit the deliverables and articrafts produced during the software lifecycle.
- Review the processes and procedures established for software development and testing.

The software quality policy forces the independent SQA group to review the software testing activities and regularly report quality assurance metrics to the management. Establishment of an effective software quality process, one that is practiced, documented, trained, enforced, measured, and improved upon, results in improved productivity and quality [Prasad 04].

Reviews and Audits:

The SQA manager ensures that the SQA process reviews of different applications are scheduled and conducted. The purpose of the review is to ensure compliance with the established policy and operational procedures and to recommend any necessary changes. The reviews are listed in the SQA audit and review schedule. The SQA manager ensures that the SQA work product audits of the articrafts produced are scheduled and conducted. The audits are conducted to determine whether the procedures and standards set by the organization are being met in the development of a specific product. SQA team members perform the audit reviews.

Quality Assurance Reporting:

The SQA manager collects and reports the data regarding process compliance and cost in hours, schedule, and status of the SQA activities. The SQA manager collects data from reviews, audits, and presents SQA review report.

Quality Assurance Training:

The skills and knowledge needed by each member of the organization to perform their functions are identified by the manager, and any skill or knowledge gaps are addressed through a training plan. Manager works with members of their teams to develop individual objectives and training plans. Skill gaps for individuals are addressed in the individual training plans and documented in the employee development, growth, and education system. Training is targeted only for employees. The objective of a training program is to ensure that all employees receive the training needed in order to perform their job.

4.4 Technology Model

The template for the technology model is given below.

TEMPLATE FOR TECHNOLOGY MODEL	
Purpose	Model to prevent Technology Failures in an ITE
Input criteria	List of common failures in an ITE • Software technology configuration mismatch across ITE • Instances issues across ITE
Input source	Technology failures in all applications in an Integrated Test Environment
Factors	
Name and location	
Ownership and contact details	
Database server specifications	
Application server specifications	
Main frame specifications	
Instance details	
Input type	
Source and destination	
Response times	
Access details	
Sub system and others	
Output criteria	• All applications in ITE are configured. • All testing instances documented. • Technical specifications of all applications documented

4.4.1 Technology Model Factors

The factors considered in the technology model are described in this subsection.

Name and Location:

The Technology model should have all application names and their address location.

Ownership and Contact Details:

The technology model should have details such as application owner information and contact information.

Specifications:

The technology model should have database server specifications, application server specifications, and main frame specifications. The specifications should include details such as dbms type, operating system details, memory size, host name, and program size. This information will help in interfacing different applications in an ITE by designing different intermediate programs to take care of applications operating on different software technologies.

Instance Details:

There may be several different testing instances for different user groups in an organization. The technology model should have details of all applications instances required in an ITE such as production instance, test instance, user acceptance testing instance (UAT), and staging test environment (STE) instance.

Input Type:

The input file received by different applications may be in different formats depending upon the type of technology they use. The technology model should list the type of input files coming into the system such as text file format, XML format, batch

type, transactional type, frequency, estimated daily traffic, and transaction direction (send or receive or both).

Source and Destination:

The technology model should list the source application sending the data and the destination application where data is being transmitted.

Response Times:

The most common fall-out noticed in an ITE is the timing issue because of the different response times taken by different applications in an ITE. So, the technology model should have details about the typical response times of all applications for both best case and worst case scenarios.

Application Access Details:

Details of how the application should be accessed with userid, password, privileges, and permissions, etc. should be included. Special software tools may be required in some cases to access the systems.

Others:

The technology model should list any special program or hardware device used as a subsystem to establish communication between applications in an ITE operating on different software technologies.

4.5 Management Model

The template for the management model is given below.

TEMPLATE FOR MANAGEMENT MODEL	
Purpose	Model to prevent management failures in an ITE
Input	List of common failures in an ITE • Inappropriate staffing • Lack of coordination among applications in an ITE • Lack of top management support for the project
Input source	Management failures in all applications in an Integrated Test Environment
Factors	
Project management	
Financial management	
Development management	
Test management	
Team management	
Configuration management	
Implementation management	
Release management	
Support management	
Environment	
Verification and audits	
Output criteria	• Adequate staffing for the project is finalized • Coordinated testing process is documented • Top management support is ensured.

4.5.1 Management Model Factors

The factors considered in the management model are described in this subsection.

Project Management:

Project management is responsible for launching a new project, decides the goals with the marketing department, and coordinates with the release department. They work with the management team to select team members for the project. Application managers are responsible for checking the capacity and level of effort as well as the risk level, and prepare the work plan for their applications. They are also responsible for document control. All staff members are responsible for maintaining any documentation under their individual control.

Financial Management:

Financial management is responsible for budget planning for the organization. It includes hardware capacity planning, technical support, disaster recovery planning, and overall resource management. Budget managers would estimate the cost of the release based on the initiatives on the candidate list and LOE's. They are responsible for getting funding approval from the clients.

Development Management:

They are responsible for the all development activities that are involved in the development of the project. They manage the development team. They monitor the capacity and develop the release development plan. A development manager will assign the work areas for each developer who is maintaining or enhancing configuration items. Developers create a private work area and use it to check out the required configuration item for editing and developing.

Test Management:

Test management is responsible for test-related activities that include preparation of test plans, test procedures, and test cases which meet the test standards. It has to ensure that the system is thoroughly tested and performs the required functions properly.

Team Management:

Team management is responsible for setting goals for the team. Team management is responsible for building the team. They are responsible for defining strategies, tasks, and plans for the team. They define the roles and responsibilities of the team members. They also decide the training programs to be conducted. Basically, they are responsible for overall team activities.

Configuration Management (CM) [Pressman 05]:

They are responsible for the overall configuration management activities, and prepare the configuration management plan. The CM plan is developed in conjunction with the project release plan, and it identifies the resources, CM check-in and check-out procedures, configuration item identification and naming procedure, application build procedure, CM schedule, change control procedures, and CM audit procedures. Configuration management is responsible for assigning work areas for each developer maintaining or enhancing configuration items. New configuration items are placed under configuration control when developers check in new configuration items to the CM build work area. If the build is successful, CM promotes the configuration items in this area to an intermediate testing area. Otherwise, CM coordinates with the development group for

corrections. This area is not available for normal check out. CM also provides an audit trial of change activity with the appropriate level of documentation.

Implementation Management:

The responsibilities of the implementation management are to produce a high-quality product, and to ensure that the implementation is fully confirmed to the design. It identifies and resolves all the implementation issues. It is responsible for training the team in producing, refining, and verifying the product implementation.

Support Management:

They are responsible for helping the team use proper tools and methods, and handle the team's configuration management and change control functions [Humphery 05b].

Release Management:

They are responsible for the overall management and coordination of the releases. They are responsible for the client interface and coordination, and implementing releases successfully and on schedule. The release manager will coordinate and communicate the release status and activity. They report the release activities and its progress and publish executive status reports.

Environments:

Environments are locations where the latest versions of the code are stored. Configuration control manages the environments by checking the levels of code being migrated, ensuring that the code is being installed is exactly the way it was tested and

received signed-off. It ensures that the code is production ready before it is installed via test environment certification.

Verifications and Audits:

Verification and audits provide traceability throughout the software development life cycle.

4.6 ITEPM Team Model

The template for the team model is given below.

TEMPLATE FOR ITEPM TEAM MODEL	
Purpose	To build an ITEPM team
Input criteria	• Goals • Strategy • Roles • Team Plans • Reviews
Input source	ITPEM team failures in all application teams in an ITE
Factors	
Attributes	
Goals	
Define strategy	
Define roles	
Define plans	
Reviews and meetings	
Training	
Team Building	
Post mortem	

Output criteria	• Team roles, goals, processes, and responsibilities are defined • Well-defined ITEPM team is established

4.6.1 Team Model Factors

The team model factors are described in this subsection.

Attributes:

The ITEPM team members need to have the following attributes in order to form a successful team [Humphery 05b].

- The team members are skilled and have the knowledge of all applications in an ITE.
- The team members are committed to achieving quality.
- Commitment to a common goal.
- Ownership of the process and plan.
- Following a disciplined personal process.
- Planning, managing, and reporting on their personal work.
- Cooperating with the team and all team members to maintain and effective and productive working environment.
- Dedication to excellence.

Goals:

The members of the teams are committed to a common set of goals to maintain the team's motivation and energy. The team agrees on the goals during project launch that can be stretched from several months to years. All members of the team know their

individual goals and understand where they stand against these goals. The team goals should be measured frequently and the members should regularly see the results of the measures. Team charts can be used to post the updates of goal progress. If there are any issues with the team, measures should be taken immediately. This will help motivating the team. All measures should be discussed, progress goals should be set and the status of each goal should be reviewed. Aggressive long-term goals should be set but they should be broken down into realistic and measurable short-term goals. Each team member should be assigned a specific job which is measurable.

Strategy:

The plans and strategies should be clearly defined for achieving the goals which are set. The members of the team would strive hard to meet the team's quality goals for every module created and tested. The team should have a strategy for reviewing every module before code inspections. The team should have to set plans to test each and every module created. The issues found during the reviews should be solved with recommendations for what to do about them. Within the context of an overall strategy, the team should make improvements to their team plans. Plans for achieving the goals should be clearly defined. The tools required to achieve the goals should be available. The team should be trained in using the ITEPM process models such as the Planning model, Requirements model, Process model, Technology model, Quality model, Testing model, and Management model.

Team Roles [Humphery 05b]:

Team roles are tasks performed by members of the team. The tasks are assigned based on the individual skill sets. The roles of the team members are defined before the project launch. Team's prior performance is reviewed and compared with the suggested roles. Discuss the roles of the members with the team to make sure that they understand the importance of their role. Team members will be committed to their task when they realize, how important their role is in making the project successful. Different team roles include Application manager, Implementation Manager, Release Manger, Test Manager, and Test teams such as Application Test team and Integration Test team. The team roles include:

- Communicating with the environment group regarding tracking/controlling the test environment.
- Preparing and maintaining test data.
- Preparing reports and summary data.
- Preparing test cases and scenarios.
- Test case execution and validation.
- Documenting errors.
- Reviewing problem reports for completeness and prioritization.

Team Plans:

The team plan is to define how the job should be done within the team. The plan defines the overall project schedules, daily targets, and weekly targets that are measurable. The plan represents the resources the team has and the strategy the team intends to follow. The teams should be motivated consistently to maintain their plan after

the project launch. The teams should follow the plans and perform the tasks specified in their plans. The plan provides the reference for everything the teams do. The plans will represent the way team members work. The team members should always update their plans to reflect their current work. If there are frequent changes in the plan or the plan is inaccurate, then the plan should accommodate for the dynamic changes and workload balancing. Team members should have a detailed and accurate plan to track their progress. The plan should define the, overall project schedules, the daily targets, and the weekly targets that are measurable. The plan should accommodate for the dynamic changes and workload balancing that may occur during the execution of the project.

Team Reviews and Meetings:

Effective team goals are defined during project launch and the team plan is discussed with all the members of the team. Tasks are assigned to each member of the team. Once the project begins, the progress will be reviewed daily or weekly. Team reviews are done at every stage of the software process. Each and every document will be reviewed. All modules created are reviewed. All test documents are reviewed. If any problems are found at any stage, recommendations should be made to rectify the errors found. A problem module will be re-reviewed, re-developed, and re-tested until defect-free software is developed and a quality product is produced. Team meetings are held frequently to track the team activities. The teams should have frequent reviews and meeting to track the progress. If progress is not being made, alternate measures should be taken to meet the target.

Team Training Programs:

Team members sometimes will not be fully productive. Some team members may not have proper skills or they might lack the required skills. Training will solve many skill problems. If team members are not trained, they take longer to do their jobs and they produce poorer quality products. With training, the quality of a developer's work will be improved and the testing time will be reduced. Proper training in the project's tools and methods will also generally shorten the overall schedule by more than enough time to compensate for the training time.

4.7 Test Plan Template

The test plan template designed for ITEPM is described in this section. This test plan format can be utilized throughout the testing of various failures in ITEPM.

TEST PLAN TEMPLATE FOR ITEPM	
Failures	Test plan (TP)
User enters type of failure	User enters test plan title:
	User enters test plan title:
	User enters test plan title:
User enters type of failure	User enters test plan title:
	User enters test plan title:
	User enters test plan title:

4.8 Test Case Template for ITEPM

The test case template designed for ITEPM is described in this section. This test case format can be utilized throughout the testing of various failures in ITEPM.

TEST CASE TEMPLATE FOR ITEPM	
Test case number	User enters test case number
Test case title	User enters test case title
Test plan reference	User enters test plan reference number from the corresponding ITEPM template model
Failure	User enters type of failure
Test planner	User enters test planner name
Test executor	User enters test executor name
Test environment	Integrated Test Environment
Application name	User enters application name
Test instance name	ITE 01
Test data	User enters test data
Test data source	User enters model name of ITEPM
Execution method	Manual/Automatic
Execution steps	Uses enter test execution steps
Expected results	User enters expected test results
Test results	User enters actual test results
Tester comments	User enter comments
Test case status	Complete/In validation/Fail/Not executed
Tester action	No action required / Trouble ticket opened
Test case history	Previous history of test case if any
Test cycle number	Cycle 1/ Cycle 2

CHAPTER V

ANALYSIS OF PRACTICAL DATA

In this chapter, the various applications involved in an ITE are described and the practical issues gathered from various ITE projects are analyzed and classified into various ITEPM failures (see Chapter III for a list of the failures). The practical ITE projects described in this chapter integrate different applications from different software technologies and provide for the end-to-end flow of a product. The applications used in this chapter were obtained from a Software Services Organization that provides various testing solutions to telecommunication industries.

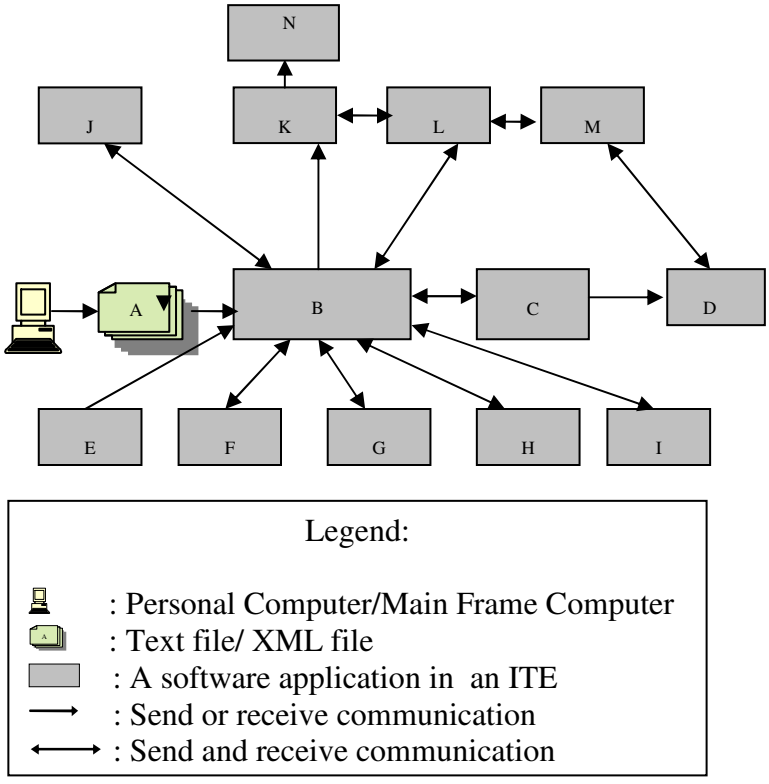

Figure 1: Example of an Integrated Test Environment

5.1 Description of Applications in Figure 1

The various applications involved in Figure 1, their function, and the technology used by them are given in the following table.

Application	Name	Function	Technology Used
A	Input file	Receives input files	Controlled text file, XML file
B	Service order generator	Generates service order	Java, Oracle database
C	Service order validator	Validates service orders	Main frame
D	Distributor	Distributes service order	Main frame
E	Account system	Stores customer account information	Java
F	File validator	Validates input files	Main frame, XML, Oracle database
G	Address validator	Validates addresses	Java
H	Facility design application	Handles all facility design information	Java
I	Reporting application	Used as a reporting tool	Oracle
J	Project application	Handles special projects	Java
K	Work force application	Handles information of all actual field work	Oracle
L	Facility router	Handles all actual facility routing	Main frame
M	Service order router	Routes service orders to corresponding regions	Main frame
N	End customer	Sends information to end customer	Java

5.2 Type of Failures in ITE Project– 1

The practical issues gathered from ITE Project-1 are given in the table below. These issues are further classified as per failures listed in Chapter III.

Type of Issue	Total Number Reported	Classification of issues as per ITEPM
Business As Usual (BAU)	9	Planning Failures
Code	27	Process Failures
Data setup	14	Process Failures
Environment	13	Technology Failures
External inputs	1	Uncontrolled Failures
Explainable Differences	1	Management Failures
Requirements	22	Requirement Failures
Unknown (TBD)	19	Requirement and Management Failures
Table entry	15	Process Failures
Testing	9	Process Failures
Total	**130**	

5.3 Type of Failures in ITE Project– 2

The practical failures data gathered from ITE Project-2 are listed in the following table.

Type of Issue	Total Number Reported	Classification of issues as per ITEPM
Business As Usual (BAU)	5	Planning failures
Code	32	Process Failures
Data setup	3	Process Failures
Environment	16	Technology Failures
External inputs	1	Uncontrolled Failures
Explainable Differences	1	Management Failures
Requirements	20	Requirement Failures
Unknown (TBD)	6	Requirement and Management Failures
Table entry	4	Process Failures
Testing	3	Process Failures
Total	**91**	

5.4 Type of Failures in ITE Project– 3

The practical failures data gathered from ITE Project-3 are given in the table below.

Type of Issue	Total Number Reported	Classification of issues as per ITEPM
Business As Usual (BAU)	7	Planning Failures
Code	50	Process Failures
Design	3	Process Failures
Data setup	5	Process Failures
Environment	17	Technology Failures
External inputs	2	Uncontrolled Failures
Explainable Differences	0	Management Failures
Requirements	16	Requirement Failures
Unknown (TBD)	22	Requirement and Management Failures
Service Order	0	
Table entry	8	Process Failures
Testing	9	Process Failures
Total	**140**	

5.5 Type of Failures in ITE Project– 4

The practical failures data gathered from ITE Project-4 are listed in the following table.

Type of Issue	Total Number Reported	Classification of issues as per ITEPM
Business As Usual (BAU)	4	Planning Failures
Code	48	Process Failures
Data	1	Process Failures
Design	2	Process Failures
Environment	12	Technology Failures
Explainable Differences	5	Management Failures
Requirements	11	Requirement Failures
Testing	9	Process Failures
Table Entry	7	Process Failures
Others	53	Requirement and Management Failures
Total	**152**	

CHAPTER VI

IMPLEMETATION AND TESTING OF THE INTEGRATED TEST ENVIRONMENT PROCESS MODEL (ITEPM)

In this chapter, various test plans, test cases, and desk checks are described for the planning model, requirements model, process model, technology model, and management model for implementation and testing of Integrated Test Environment Process Model (ITEPM).

6.1 Test Plan for Planning Model

A test plan for the planning model for testing the planning failures is given in the table below.

TEST PLAN FOR PLANNING MODEL	
Planning failures	Test plan (TP)
1. Unrealistic schedules	TP6.1.1: Release schedule checking
	TP6.1.2: Resource capacity verification
	TP6.1.3: Compare capacity model with LOE
2. Out-of-sync release schedules and code delivery dates	TP6.1.4: Release management deliverables checking
	TP6.1.5: Risk analysis checking
3. No dedicated ITE	TP6.1.6: Check environment requirement for all applications

6.2 Test Plan for Requirements Model

A test plan for the requirements model for testing the requirements failures is given in the table below.

TEST PLAN FOR REQUIREMENTS MODEL	
Requirement failures	Test Plan(TP)
1. Misunderstanding of requirements	TP6.2.1: Requirements checking
2. Incorrect requirements	TP6.2.2: Requirements review
3. Frequent requirement changes	TP6.2.3: Requirement modifications checking
4. Requirements conflict among applications	TP6.2.4: Requirements conflict among different application groups checking

6.3 Test Plan for Process Model

A test plan for the process model for testing the process failures is given in the table below.

TEST PLAN FOR PROCESS MODEL	
Process failures	Test Plan(TP)
1. Poor quality	TP6.3.1: Interface readiness checking
	TP6.3.2: Regression testing
	TP6.3.3: Progression testing
	TP6.3.4: Test case verification
	TP6.3.5: Test status tracking
	TP6.3.6: Verify SQA practices and procedures followed
	TP6.3.7: Validate SQA reviews and audits

2. Data configuration mismatch	TP 6.3.8: Data checks verification
3. Data sharing issues	TP6.3.9: Report the data sharing issues
4. Lack of progress tracking system	TP6.3.10: Check the progress tracking system

6.4 Test Plan for Technology Model

A test plan for the technology model for testing the technology failures is given in the table below.

TEST PLAN FOR TECHNOLOGY MODEL	
Technology failures	Test Plan(TP)
1. Software technology configuration mismatch across ITE	TP6.4.1: Check the technical specifications
	TP6.4.2: Check of hardware and sub-systems involved
2. Instances issues across ITE	TP6.4.3: Instances issues to be checked

6.5 Test Plan for Management Model

A test plan for the management model for testing the management failures is given in the table below.

TEST PLAN FOR MANAGEMENT MODEL	
Management failures	Test Plan(TP)
1. Inappropriate staffing	TP6.5.1: Check project management plan
	TP6.5.2: Check financial management plan
2. Lack of coordination among applications in an ITE	TP6.5.3: Check Configuration management plan
	TP6.5.4: Check environment conflicts
	TP6.5.5: Check implementation management

3. Lack of top management support	TP6.5.6: Check release management

6.6 Test Plan for ITPEM Team Model

A test plan for ITEPM team model is given in the table below.

TEST PLAN FOR ITPEM TEAM MODEL	
Team factors	Test Plan(TP)
1. Goals	TP6.6.1: Check team goals
2. Strategy	TP6.6.2: Check team strategy
3. Roles	TP6.6.3: Check team roles
4. Team plans	TP6.6.4: Check team plan
5. Reviews	TP6.6.5: Check for team reviews
	TP6.6.6: Check the team building activities
	TP6.6.7: Check training programs

6.7 Test Cases and Desk Checks for Planning Model

Test cases and desk checks considered for testing the planning failures are given in the tables below.

Test case number 6.7.1	
Test case title	Check if the release schedule of all applications match with that of ITE release schedule
Test plan reference	Planning model : TP6.1.1
Failure	Planning failure : Unrealistic schedule
Test planner	Mohan
Test executor	Mohan
Test environment	Integrated Test Environment
Application name	All applications in an ITE
Test instance name	ITE 01
Test data	

Test data source	Planning model reference number 4.1.1
Execution method	Manual
Execution steps	• Go to planning model • Check the release schedules of all applications • Check the ITE release schedule. • Check if the release date for applications in an ITE are earlier than ITE release date
Expected results	Test passes if release dates of all applications are earlier than ITE release date
Test results	
Tester comments	
Test case status	
Tester action	
Test case history	No history available
Test cycle number	First cycle

Test case number 6.7.2	
Test case title	Compare level of effort and capacity available
Test plan reference	Planning model : TP6.1.3
Failure	Planning failure: Unrealistic schedule
Test planner	Mohan
Test executor	Mohan
Test environment	Integrated Test Environment
Application name	All applications in an ITE
Test instance name	ITE 01
Test data	
Test data source	Planning model reference number 4.1.1
Execution method	Manual
Execution steps	• Go to planning model • Check the level of effort • Check capacity model • Check if level of effort is less than capacity
Expected results	Test passes if level of effort in number of hours is less the capacity in number of hours.
Test results	
Tester comments	
Test case status	
Tester action	
Test case history	No history available
Test cycle number	First cycle

Test case number 6.7.3	
Test case title	Check if the required test environment is available for all applications in an ITE
Test plan reference	Planning model: TP6.1.6
Failure	Planning failure: No dedicated ITE
Test planner	Mohan
Test executor	Mohan
Test environment	Integrated Test Environment
Application name	All applications in an ITE
Test instance name	ITE 01
Test data	
Test data source	Planning model reference number 4.1.1
Execution method	Manual
Execution steps	• Go to planning model • Check the environment required to test the applications • Check if the test environment is available to test the applications
Expected results	Test passes if the test environment is available to all applications in an ITE
Test results	
Tester comments	
Test case status	
Tester action	
Test case history	No history available
Test cycle number	First cycle

Desk checks number 6.7.4		
	Yes	No
1. Is the release date of the project as per the release management calendar date		
2. Is the hardware capacity sufficient for the applications in an ITE		
3. Is the estimated hours for the project sufficient		
4. Is the cost estimation done		
5. Has the release management released the list of releases in an year		
6. Are all work requests included in the release development plan		
7. Are the reviews conducted at every stage of the software cycle for all applications in an ITE		
8. Are status reports updated at all the stages of the software cycle for all applications in an ITE		
9. Is risk analysis done regularly		

6.8 Test Cases and Desk Checks for Requirements Model

Test cases and desk checks considered for testing the requirements failures are given in the tables below.

Test case number 6.8.1	
Test case title	Check the requirements with all the application groups in an ITE to verify if there is any misunderstanding of requirements
Test plan reference	Requirements model: TP6.2.1
Failure	Requirements failure: Misunderstanding of requirements
Test planner	Mohan
Test executor	Mohan
Test environment	Integrated Test Environment
Application name	All applications in ITE
Test instance name	ITE 01
Test data	
Test data source	Requirement model reference number 4.2.1
Execution method	Manual
Execution steps	• Go to the requirements model • Check with all application groups in an ITE if the requirements can be implemented • Review the BRD and SRD • Verify that the requirements are understood correctly by applications
Expected results	Test passes if requirements can be implemented by all applications in an ITE
Test results	
Tester comments	
Test case status	
Tester action	
Test case history	No history available
Test cycle number	First cycle

Test case number 6.8.2	
Test case title	Check if the requirements are correct
Test plan reference	Requirements model: TP6.2.2
Failure	Requirements failure: Incorrect requirements
Test planner	Mohan
Test executor	Mohan
Test environment	Integrated Test Environment
Application name	All applications in ITE

Test instance name	ITE 01
Test data	
Test data source	Requirement model reference number 4.2.1
Execution method	Manual
Execution steps	• Go to the requirements model. • Review the requirements with all application groups in an ITE • Verify that the requirements are correct
Expected results	Test passes if the requirements are correct
Test results	
Tester comments	
Test case status	
Tester action	
Test case history	No history available
Test cycle number	First cycle

Test case number 6.8.3	
Test case title	Check the requirement modifications for all applications in an ITE
Test plan reference	Requirements model: TP6.2.3
Failure	Requirements failure: Frequent requirement changes
Test planner	Mohan
Test executor	Mohan
Test environment	Integrated Test Environment
Application name	All applications in an ITE
Test instance name	ITE 01
Test data	
Test data source	Requirement model reference number 4.2.1
Execution method	Manual
Execution steps	• Go to the requirements model • Review the modified requirements with all application groups in an ITE • Check if the candidate list is updated with the modified requirements • Verify that the modified requirements are correct and can be implemented
Expected results	Test passes if the modified requirements can be implemented
Test results	
Tester comments	
Test case status	
Tester action	
Test case history	No history available
Test cycle number	First cycle

Test case number 6.8.4	
Test case title	Check the requirement conflicts among different applications when requirement changes are frequently asked
Test plan reference	Requirements model: TP6.2.4
Failure	Requirements failure: Requirement conflict among applications
Test planner	Mohan
Test executor	Mohan
Test environment	Integrated Test Environment
Application name	All applications in an ITE
Test instance name	ITE 01
Test data	
Test data source	Requirement model reference number 4.2.1
Execution method	Manual
Execution steps	• Go to the requirements model • Review the modified requirements with all application groups in an ITE • Check how it impacts different applications • Verify that there are no requirement conflicts among different applications
Expected results	Test passes if the requirements can be implemented without any conflicts among applications in an ITE
Test results	
Tester comments	
Test case status	
Tester action	
Test case history	No history available
Test cycle number	First cycle

Desk checks number 6.8.5		
	Yes	No
1. Are the requirements complete and accurate		
2. Are the requirements of all applications in an ITE reviewed		
3. Is the candidate list from all application groups in an ITE ready		
4. Are the requirements analysis done with all the application groups in an ITE		
5. Do the modified requirements impact all the applications in an ITE		
6. Are the modified requirements notified to all impacted application groups in an ITE		
7. Are the test cases mapped to requirements using traceability matrix		

6.9 Test Case and Desk Checks for Process Model

Test cases and desk checks considered for testing the process failures are given in the tables below.

Test case number 6.9.1	
Test case title	Check if all the applications are tested and reviewed individually before testing in an ITE
Test plan reference	Process model: TP6.3.1
Failure	Process failure: poor quality
Test planner	Mohan
Test executor	Mohan
Test environment	Integrated Test Environment
Application name	All applications in an ITE
Test instance name	ITE 01
Test data	
Test data source	Process model reference number 4.3.1
Execution method	Manual
Execution steps	• Go to the process model • Check if all interfaces are ready for ITE testing • Check if the testing is completed for all applications in an ITE • Verify if reviews are completed for tested applications
Expected results	Test passes if all applications in ITE are ready for testing in an ITE
Test results	
Tester comments	
Test case status	
Tester action	
Test case history	No history available
Test cycle number	First cycle
Test case number 6.9.2	
Test case title	Report data mismatch errors for all the applications in an ITE
Test plan reference	Process model: TP6.3.8
Failure	Process failure: Data configuration mismatch
Test planner	Mohan
Test executor	Mohan
Test environment	Integrated Test Environment
Application name	All applications in an ITE
Test instance name	ITE 01
Test data	
Test data source	Process model reference number 4.3.1
Execution method	Manual

Execution steps	• Go to the process model • Check the test data for all applications in an ITE • Check for any data configuration mismatch between the applications in an ITE
Expected results	Test passes if there is no data mismatch among applications in an ITE
Test results	
Tester comments	
Test case status	
Tester action	
Test case history	No history available
Test cycle number	First cycle

Test case number 6.9.3	
Test case title	Report the data sharing issues among applications in an ITE
Test plan reference	Process model: TP6.3.9
Failure	Process failure: Data sharing issues
Test planner	Mohan
Test executor	Mohan
Test environment	Integrated Test Environment
Application name	All applications in an ITE
Test instance name	ITE 01
Test data	
Test data source	Process model reference number 4.3.1
Execution method	Manual
Execution steps	• Go to the process model • Check the test data for all applications in an ITE • Check if any test data is shared with other applications • Check if the data sharing process is finalized among applications in an ITE
Expected results	Test passes if there is no data sharing issues among applications in an ITE
Test results	
Tester comments	
Test case status	
Tester action	
Test case history	No history available
Test cycle number	First cycle

Test case number 6.9.4	
Test case title	Check if the data from reviews and audits are tracked and

	reported
Test plan reference	Process model: TP6.3.10
Failure	Process failure: Lack of progress tracking system
Test planner	Mohan
Test executor	Mohan
Test environment	Integrated Test Environment
Application name	All applications in an ITE
Test instance name	ITE 01
Test data	
Test data source	Process model reference number 4.3.1
Execution method	Manual
Execution steps	• Go to the process model • Check the review dates • Check the review results • Check the auditing process • Check if all the details are reported
Expected results	Test passes if reviews and audits are tracked and reported
Test results	
Tester comments	
Test case status	
Tester action	
Test case history	No history available
Test cycle number	First cycle

Desk checks number 6.9.5		
	Yes	No
1. Is SQA practices and processes followed by the all the application teams in an ITE		
2. Are the guidelines of the SQA followed by the all the application teams in an ITE		
3. Has the defined standards met by the all the application teams in an ITE		
4. Has reviews done by the SQA team for all application groups in an ITE		
5. Is the software quality assurance metrics reports updated for the project		
6. Are audits to check the procedures and standards set by the organization done by all the application teams in an ITE		
7. Are audit reviews done for the project		
8. Do you have non-compliance issue in an ITE		
9. Is the SQA review report given for the project ITE		
10. Is the necessary training organized for the team members		
11. Are team members in an ITE trained regarding the data sharing operations among applications		

6.10 Test Cases and Desk Checks for Technology Model

Test cases and desk checks considered for testing the technology failures are given in the tables below.

Test case number 6.10.1	
Test case title	Check software technology configuration mismatch across all the applications in an ITE
Test plan reference	Technology model: TP6.4.1
Failure	Technology failure: Software technology configuration mismatch across ITE
Test planner	Mohan
Test executor	Mohan
Test environment	Integrated Test Environment
Application name	All applications in an ITE
Test instance name	ITE 01
Test data	
Test data source	Technology model reference number 4.4.1
Execution method	Manual
Execution steps	• Go to technology model • Check the technical specifications • Check if all applications in an ITE are configured • Conduct shake-out test to ensure connectivity • Report if there is any configuration mismatch
Expected results	Test passes if there is no configuration mismatch across ITE
Test results	
Tester comments	
Test case status	
Tester action	
Test case history	No history available
Test cycle number	First cycle

Test case number 6.10.2	
Test case title	Check instance issues across applications in an ITE
Test plan reference	Technology model: TP6.4.3
Failure	Technology failure: Instances issues across ITE
Test planner	Mohan
Test executor	Mohan
Test environment	Integrated Test Environment
Application name	All applications in an ITE
Test instance name	ITE 01
Test data	

Test data source	Technology model reference number 4.4.1
Execution method	Manual
Execution steps	• Go to technology model • Check applications running on multiple instances • Check applications running on single instance • Check if configuration process is finalized
Expected results	Test passes if proper interfaces are established to match one to many relation of instances
Test results	
Tester comments	
Test case status	
Tester action	
Test case history	No history available
Test cycle number	First cycle

Desk checks number 6.10.3		
	Yes	No
1. Is application owner details like name, location, and contact information available		
2. Do you have the details of dbms type, and operating system for database server		
3. Do you have the details of operating system, memory size, host name, and program size for application server		
4. Do you have the details of operating system, memory size, host name, and program size for mainframe server		
5. Is production instance details available		
6. Is test instance details available		
7. Is user acceptance test instance details available		
8. Is staging test environment instance details available		
9. Do you have details of type of input coming into the system		
10. Is the source of data coming into the system available		
11. Is the destination source going out of the system available		
12. Is the response time available for all the scenarios in the all the applications		
13. Are access details like login and password available		
14. Do you have list of hardware and sub-system involved in all the		

application		
15. Is there is any data masking involved		

6.11 Desk Checks for Management Model

Desk checks considered for testing the management failures are given in the table below.

Desk checks number 6.11.1		
	Yes	No
Project Management		
1. Is project team selection done		
2. Is training required		
3. Has application manager done capacity planning and level of effort		
4. Has application management done risk assessment		
5. Is work plan made ready		
6. Is document control done		
Financial Management		
7. Is approved budget sufficient for the project		
8. Is hardware capacity planning done		
9. Is technical support provided for the project		
10. Is disaster recovery planning done		
11. Are resources sufficient for the project		
Development Management		
12. Are development manager responsibilities defined		
13. Is the release plan ready		
14. Are the design documents ready		
15. Is the tracking system for the development process ready		
Test Management		
16. Are they using the right test methodologies		
17. Are test documents ready on time		
18. Have they done functionality testing		
19. Is the overall testing complete		
Configuration Management		

20. Is configuration management plan ready		
21. Is code control done for development work		
22. Is change control taken care of		
23. Is configuration manager coordinating with different groups		
Implementation Management		
24. Are all the measures taken to produce the high quality product		
25. Is implementation fully confirmed with the design		
26. Are all the implementation issues resolved		
Support Management		
27. Are they helping the team with the proper software tools		
28. Are they helping the team to use proper methods to handle the configuration management		
29. Are they supporting the team in handling the team's configuration management and change control functions		
Release Management		
30. Is the implementation ready as per release schedule		
31. Is release manager tracking the release status and activity		
32. Has release manager given progress reports to the executives		
Environment		
33. Is environment made available for testing		
Verifications and Audits		
34. Is verification and audits done for the project		

6.12 Desk Checks for ITEPM Team Model

Desk checks considered for testing the ITEPM team model is given in the table below.

Desk checks number 6.12.1		
	Yes	No
1. Are the team goals set		
2. Is the individual goal set for all the team members		
3. Is the team goal measured frequently		
4. Do the teams have strategy to execute the team plan		
5. Do the teams have strategy to improve the team plan		
6. Are the roles of the individual team members defined		

7. Do the teams have team plan for the project		
8. Are the teams following the plan		
9. Are the teams updating the plan		
10. Is the team plan accommodating for the dynamic changes		
11. Are the team reviews done frequently		
12. Are the code reviews done before code implementation		
13. Are all the test documents reviewed		
14. Are the team meetings done frequently for team building activities		
15. Are the team members trained with proper skills		
17. Is the training sessions arranged for the team members		

CHAPTER VII

SUMMARY AND FUTURE WORK

7.1 Summary

Part of the objective of this thesis work was to analyze the reasons for failures in software projects. Reasons for failures during the various stages of the testing process in an Integrated Test Environment (ITE) were discussed in detail. Based on this investigation, the main thrust of this thesis work was the development of an Integrated Test Environment Process Model (ITEPM) to control software failures. The proposed model includes effective software practices that were developed based on practical industry data. This model was tested for efficacy, applicability, viability, and practicability.

Chapter I discussed the importance of software quality and software testing. Chapter II provided information about the importance of an Integrated Test Environment (ITE). Chapter III discussed reasons for software project failures in an Integrated Test Environment. Chapter IV described the design and development of an Integrated Test Environment Process Model (ITEPM). Chapter V included analysis of practical data from various software projects involving an Integrated Test Environment. Chapter VI described the implementation and testing of an Integrated Test Environment Process Model with test plans, test cases, and desk checks.

It is widely known that creating good production-quality software is quite challenging and the software industry must use the best available methods to improve quality. A software development process that utilizes the proposed model should be able to manage if not control the occurrence of software failures in an Integrated Test Environment and help to produce production-quality software consistently.

7.2 Future Work

For future work, the factors listed in the planning model, requirements model, process model, technology model, management model, and ITEPM team model can be further broken down into individual models to cover more failure points than listed in this thesis report. For each application in an Integrated Test Environment an Application Process Model can be developed similar to the Integrated Test Environment Process Model proposed in this thesis work. More test cases and desk checks can be considered for a more thorough testing of the model.

REFERENCES

[Eickelmann and Richardson 96] Nancy S. Eickelmann and Debra J. Richardson, "An Evaluation of Software Test Environment Architecture", *Proceedings of the 18th International Conference on Software Engineering (ICSE-18)*, pp. 353-364, Berlin, Germany, March 1996.

[Hower 07] Rick Hower, "Software QA and Testing Resource Center", URL: http://www.softwareqatest.com
date created: July 2007, access date: September 2007

[Humphrey 02] Watts S. Humphrey, *Winning with Software: An Executive Strategy*, pp. 17-18, 70-71, 115-116, Addison-Wesley, Boston, MA, 2002.

[Humphrey 05a] Watts S. Humphrey, *PSP: A Self-Improvement Process for Software Engineers*, *SEI Series in Software Engineering*, pp. 109-132, 225-240, 287-294, Addison-Wesley, Upper Saddle River, NJ, 2005.

[Humphrey 05b] Watts S. Humphrey, *TSP: Leading a Development Team*, *SEI Series in Software Engineering*, pp. 47-60, 93-105, 115-124, 133-150, 251-255, Addison-Wesley, Upper Saddle River, NJ, 2005.

[Murthy 07] Sanjay Murthy, "Useful Estimation Techniques for Software Projects", URL: http://www.developer.com/ mgmt/article.php/1463281
date created: unknown, access date: September 2007

[Prasad 04] K.V.K.K Prasad, *Software Testing Tools*, pp. 69-96, Dreamtech Press, Daryaganj, New Delhi, 2004.

[Pressman 05] Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, pp. 181, 498-499, McGraw-Hill, New York, NY, 2005.

[Rubinstein 07] David Rubinstein, "Standish Group Report: There Is Less Development Chaos Today", URL:
http://www.sdtimes.com/article/story-20070301-01.html
date created: March 2007, access date: May 2007.

[Tiwana and Keil 04] Amrit Tiwana and Mark Keil, "The One-Minute Risk Assessment Tool", *Communications of the ACM*, Vol. 47, No. 11, pp. 75-76, November 2004.

[Wallace and Keil 04] Linda Wallace and Mark Keil, "Software Project Risks and Their Effect on Outcomes", *Communications of the ACM*, Vol. 47. No. 4, pp. 68-72, April 2004.

APPENDICES

APPENDIX A

GLOSSARY

BAU	Business As Usual indicates that the part of the business functionality that is not defined in the new functionality requirements, has to work the same way it is currently working.
BRD	Business Related Document refers to the business requirement common for all systems in an ITE.
CM	Configuration Management is a process to establish and maintain the integrity of the components of a software application throughout the project's lifecycle.
ITE	An Integrated Test Environment provides a testing environment where individual software applications are combined and tested as a group. The connectivity and flow between the different applications are tested to achieve pre-determined results.
ITEPM	Integrated Test Environment Process Model, a structured process model to effectively manage, monitor, and implement ITE software projects.
LOE	Level of Effort is a detailed estimate of the number of hours required to complete the analysis, design, build, and testing a specific work request.
OOM	The Order Of Magnitude is an estimation of the needed effort to complete the work request based on the information available in the project profile [Murthy 07].
PROGRESSION TESTING	Testing of new functionalities in conformance with the requirements.

PROJECT HEAD	A person who is in charge of a software project. The primary responsibility of a project head is to ensure that all work related to the project is completed on time, within budget and scope, and in conformance to the requirements.
PSP	Personal Software Process [Humphrey 05a], a process that claims to convert capable software engineers into disciplined and highly efficient software engineers.
RDP	Release Development Plan includes specific plans, schedules, status, risks, and software quality assurance functions [Humphrey 05b].
REGRESSION TESTING	Regression Testing is carried out to ensure that modifications to one portion of software have will not impact other portions of the software [Prasad 04].
SHAKE-OUT TESTING	This test is done to check the interface readiness to ensure the connectivity of all applications in an ITE.
SQA	Software Quality Assurance will add visibility to practices and processes to ensure that the key deliverables and design articrafts meet the defined standards [Prasad 04].
SRD	A System Related Document defines the individual system-related requirements in conformance with a BRD.
STE	A Staging Test Environment provides a production-like test environment for integration testing before the software is implemented in production.
TBD	To Be Decided later.
TC	A Test Case defines the set of input parameters for which software is tested.
TP	Test Plan provides the framework of required resources, test approaches, and test methodologies for defining detailed test cases. All related test cases are grouped under one test plan.
TSP	Team Software Process [Humphrey 05b], a process that claims to build and maintain motivated and committed engineering teams. Also, it is claimed that it addresses the growing need for capable software teams that deliver quality products on schedule and within their committed costs.
UAT	User Acceptance Testing is the testing generally performed by the

	end-user clients who are actually going to use the application.
WR	A Work Request is a business client's request for development of a new product or modifications to be made to an existing product.

APPENDIX B

RESULTS OF TESTING

This appendix contains test data for testing ITEPM and eight sample test case results that were referenced in Chapter VI. These test cases can be executed on every application in an ITE.

1. Test Data for Testing ITEPM

The sample practical test data was obtained from various applications involved in ITE projects described in Chapter V.

Work Request 1AZ234							
ITE testing start date 10/11/07							
ITE testing end date 11/07/07							
ITE Release date: 11/11/07							
Appli-cation	Instance	Release date	LOE (Hours)	Capacity (Hours)	Application testing end date	Data sharing	SRD finalized date
A	ITE	11/10/07	828	1200	10/10/07		9/10/07
B	ITE01	11/10/07	2350	2500	10/10/07	With J and L	9/10/07
C	ITE	09/03/07	530	600	10/10/07		7/10/07
D	UAT	03/07/08	2200	1500	02/02/08		TBD
E	ITE	11/10/07	225	300	10/15/07		5/10/07
F	ITE	11/10/07	430	500	10/25/07		TBD
G	ITE, UAT, STE	11/10/07	422	500	08/10/07		7/10/07
H	ITE	11/10/07	328	400	10/06/07		7/10/07
I	ITE	11/10/07	226	240	Not tested		7/10/07
J	ITE	11/10/07	536	580	09/10/07	With B	
*K	ITE	None	None	None	None	With B	None

*L	ITE	None	None	None	None		None
*M	ITE	None	None	None	None		None

* Applications are not impacted by the work request 1AZ234

2. Test Results

The test results, after executing the test cases described in Chapter VI, are listed in the following table.

Test case	Application tested	Test results	Comments
Test case number 6.7.1	A	Pass	ITE release date same as application A release date
Test case number 6.7.1	D	Fail	ITE release date earlier than application A release date
Test case number 6.7.2	B	Pass	LOE hours less than capacity hours
Test case number 6.7.2	D	Fail	LOE hours more than capacity hours
Test case number 6.7.3	E	Pass	Test environment is available in ITE
Test case number 6.7.3	G	Fail	No test environment is available in ITE
Test case number 6.9.1	A	Pass	Application testing will be completed before ITE testing
Test case number 6.9.1	I	Fail	Application testing will not be tested before ITE testing

3. Conclusions Based on Test Results

- Work request 1AZ234 can only be partially implemented until Application D is ready if that is acceptable to the business client, otherwise it cannot be implemented.
- Application D needs more resources to implement work request 1AZ234.
- Application G needs a test environment in an ITE.

- ITE testing cannot be started until Application I is ready and this may result in missing the ITE release schedule.

VITA

Mohan Bheemasenarao

Candidate for the Degree of

Master of Science

Thesis: AN INTEGRATED TEST ENVIRONMENT PROCESS MODEL TO CONTROL SOFTWARE FAILURES

Major Field: Computer Science

Education: Bachelor's degree in Mechanical Engineering from Government B.D.T College of Engineering, University of Mysore, Davangere, Karnataka State, India in January 1988; completed the requirements for the degree of Master of Science in Computer Science at the Computer Science Department of Oklahoma State University in December 2007.

Experience: Worked in Computer Aided Design and Computer Aided Manufacturing Applications from 1988 to 1999 in India. Working as a Software Consultant from 2002 in the US and involved in providing testing solutions, testing automations, and process improvements.

Name: Mohan Bheemasenarao Date of Degree: December 2007

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: AN INTEGRATED TEST ENVIRONMENT PROCESS MODEL TO CONTROL SOFTWARE FAILURES

Pages in Study: 68 Candidate for the Degree of Master of Science

Major Field: Computer Science

Scope and Method of Study: Software is important to virtually any business in today's world. The most challenging issue that the software industry is facing today is how to produce quality software consistently. When there are multiple applications involved in a project in an integrated environment, controlling software quality becomes more complex. Even though many methodologies have been suggested to address the issue of software quality, there is always a demand for new and innovative software testing processes and methodologies in the face of changing technologies.

Findings and Conclusions: Part of the objective of this thesis work was to analyze the reasons for failures in software projects. In particular, the main focus was reasons for failures during the various stages of the testing process in an Integrated Test Environment (ITE). Based on this analysis, the main thrust of this thesis work was the development of an Integrated Test Environment Process Model (ITEPM) to control software failures. The proposed model includes effective software practices that were developed based on practical industry data. This model was tested for efficacy, applicability, viability, and practicability. A software development process that utilizes the proposed model will be able to control the software failures in an Integrated Test Environment and improve the software quality.

ADVISOR'S APPROVAL: Dr. M. H. Samadzadeh