

ATHABASCA UNIVERSITY

IMPACT OF DISTANCE / DISTRIBUTED PROJECT MANAGEMENT

ON

DIFFERENT SOFTWARE DEVELOPMENT METHODOLOGIES

BY

GERALD LAURENT CAISSY

A thesis project submitted in partial fulfillment

Of the requirements for the degree of

MASTER OF SCIENCE in INFORMATION SYSTEMS

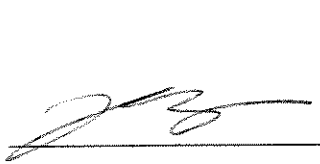
Athabasca, Alberta

March 2005

© Gerald L. Caissy, 2005

## ATHABASCA UNIVERSITY

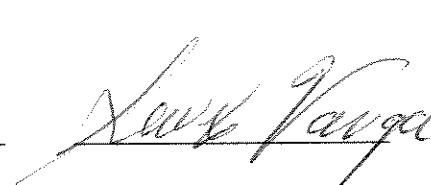
The undersigned certify that they have read and recommend for acceptance the thesis project  
“IMPACT OF DISTANCE / DITRIBUTED PROJECT MANAGEMENT ON DIFFERENT  
SOFTWARE DEVELOPMENT METHOLOGIES” submitted by “GERALD L. CAISSY” in  
partial fulfillment of they requirements for the degree of MASTER OF SCIENCE in  
INFORMATION SYSTEMS.



Xiaokun Zhang, Ph.D.

Associate Professor

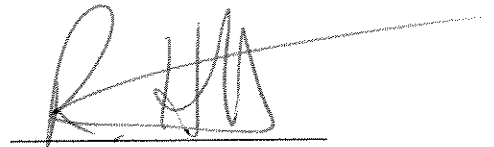
Master Project Supervisor



Lewis Varga, Ph. D.

Assistant Professor

Review Committee Chair



Richard Huntrods, M. Eng.

Academic Coordinator

Review Committee Examiner

Date: May 25, 2005

Date: June 2005

Date: June 16, 2005

## **DEDICATION**

To Deb, you are the love of my life; I am so fortunate to have found you 20<sup>+</sup> years ago. To our children Tyler, Danielle, and Mitchell work hard, stay committed, and have fun achieving your goals. Thank you for your support. I love you!

## **ABSTRACT**

This paper explores the question of possible impact(s) caused by Distributed Project Management on Software Development Methodologies. Project Management adopters claim benefits from the application of such practices in software development projects from the perspective of control. Software Development Methodologies provide us with models, which allow for efficient software construction. The methodologies and/or approaches profiled in this paper are: Waterfall, Object-Oriented, Rapid Prototyping, and Extreme Programming. Although some have a more extensive historical track record, they are all accepted approaches in attempting to build a software solution to a given development problem.

Distributed Project Management is the application of standard project management over a distributed venue. A key observation made with regards to this distributed application of project management is that communication is the single most important influencing factor. In the absence of good communication protocols and/or standards, projects find themselves at greater risk of negative impact. This impact is very difficult to isolate as solely a Distributed Project Management issue or solely a Software Development Methodology issue.

As part of this paper, an industry survey provides interesting ‘state of affairs’ information for consideration, not intended as statistically defensible data, but to be used in an observatory manner only. What the survey does is provide a sampling that indicates the largest portion of the local, regional industries are applying Waterfall and/or Object-Oriented Methodologies.

Of the Software Development Methodologies and/or approaches profiled, Extreme Programming is possibly the most at risk of negative impacts, primarily due to the unique physical characteristics of Classical XP Methodology. To say that the other three methodologies are exempt of any impact by Distributed Project Management would be premature, but their impacts are not as noticeable as in the case of the Extreme Programming Methodology.

It is conceivable that the process of managing a distributed software development project may not only incur negative impacts, but may introduce positive impacts, such as the ability to draw from a more specialized pool of development professionals.

Regardless of negative or positive impacts, the distributed application of project management to software development projects, as well as methodologies, is a relatively young application domain. The youthfulness of this domain's application suggests its evolution is not complete and may not have progressed far enough to yield solid evidence with regards to the impacts imposed on Software Development Methodologies.

## **ACKNOWLEDGMENTS**

First, I would like to thank Deb who has stood by me throughout this educational undertaking, which has challenged both recreational time and family logistics within a two career / three teenager home. To my children Tyler, Danielle, and Mitchell, I want to thank you for giving me the time and space to study and for always turning down the stereos when I studied.

I would like to take this opportunity to acknowledge my employer, Holland College, for their support, via financial, time, and technology resources, provided to me for my educational undertaking. A special note of thanks in recognition of their support to, Dr. Brian McMillan (Vice-President Academic) and Dr. Sandy MacDonald (former Executive Director of Programs), also to Sylvia Poirier (Registrar at Holland College) for informing me of the Athabasca University Master of Science in Information Systems program. Professional development is a critical aspect of all technology practitioners in today's fast moving technology cycles, to be supported by one's employer is a tremendous professional and personal advantage. Thank you!

I would like to thank all Athabasca University Master of Science in Information Systems faculty and staff for their support during my educational experience with them. Special thank you to Dr. Xiaokun Zhang faculty advisor and paper supervisor. As well as a note of thanks to Jackie Terrien, who always clarified administrative questions about the program. Finally, I would like to thank Gaylene Carragher, technical reader; your assistance was invaluable.

## TABLE OF CONTENTS

<b>ATHABASCA UNIVERSITY.....</b>	<b>I</b>
<b>DEDICATION .....</b>	<b>II</b>
<b>ABSTRACT .....</b>	<b>III</b>
<b>ACKNOWLEDGMENTS.....</b>	<b>V</b>
<b>LIST OF TABLES.....</b>	<b>VIII</b>
<b>LIST OF FIGURES .....</b>	<b>IX</b>
<b>CHAPTER I.....</b>	<b>1</b>
<b>INTRODUCTION .....</b>	<b>1</b>
<i>Statement of Purpose .....</i>	<i>1</i>
<i>Research Problem or Question.....</i>	<i>2</i>
<i>Assumptions of the Paper.....</i>	<i>3</i>
<i>Significance.....</i>	<i>3</i>
<i>Limitations .....</i>	<i>4</i>
<i>Delimitations.....</i>	<i>4</i>
<i>Definition of Terms .....</i>	<i>4</i>
<i>Organization of Paper .....</i>	<i>5</i>
<b>CHAPTER II.....</b>	<b>6</b>
<b>PROJECT MANGEMENT / SOFTWARE DEVELOPMENT METHODOLOGY</b>	
<b>ENVIRONMENT OVERVIEW.....</b>	<b>6</b>
<i>Project Management Body of Knowledge Summarization .....</i>	<i>6</i>
<i>Project Life Cycle Models Review .....</i>	<i>9</i>
<i>Software Development Methodology Summarization.....</i>	<i>13</i>

<b>CHAPTER III .....</b>	<b>23</b>
DISTRIBUTED PROJECT MANAGEMENT (DPM) STATE OF THE ART .....	23
<i>Current Distributed Project Management State of the Art Overview.....</i>	23
<i>Review of Supporting Literature in Distributed Project Management.....</i>	25
<i>Exploration of Distributed Project Management Tools.....</i>	28
 <b>CHAPTER IV .....</b>	 <b>33</b>
INDUSTRY SURVEY .....	33
<i>Scanned Environment Review.....</i>	33
<i>Data Collection Techniques.....</i>	33
<i>Data Analysis .....</i>	34
<i>Survey Findings Summarization .....</i>	44
 <b>CHAPTER V .....</b>	 <b>47</b>
DISTRIBUTED PROJECT MANAGEMENT IMPACT ON SOFTWARE DEVELOPMENT METHODOLOGIES.....	47
<i>Impact on Software Development Methodology by Distributed Project Management ..</i>	47
<i>Review of Specific Software Development Methodologies Impact from DPM.....</i>	51
<i>Distributed Project Management Benefits / Impediments in Software Development.....</i>	57
<i>Distributed Project Management Impediment Elimination Strategies.....</i>	60
<i>Best Practices of Distributed Project Management on Software Development .....</i>	62
<i>Distributed Project Management Evaluation Criteria .....</i>	64
 <b>CHAPTER VI .....</b>	 <b>66</b>
CONCLUSIONS AND RECOMMENDATIONS.....	66
<i>Conclusion .....</i>	66
<i>Further Research Suggestions .....</i>	68
 <b>REFERENCES .....</b>	 <b>69</b>
 <b>APPENDIX A ~ SURVEY TOOL.....</b>	 <b>71</b>
 <b>APPENDIX B ~ ASSESSMENT MATRIX.....</b>	 <b>74</b>



## LIST OF TABLES

	<b><u>Page</u></b>
1. Geographical / Affiliation Influence on Project Model	24
2. Benefits / Drawbacks of Distributed Projects on Software Development Methodologies	58

## LIST OF FIGURES

	<u>Page</u>
1. Project Management Knowledge Management Areas Content	8
2. Traditional Software Development Phases	13
3. WaterFall Development Methodology Diagram	16
4. Object-Oriented Development Methodology Diagram	18
5. Rapid Prototyping Development Methodology Diagram	19
6. Extreme Programming Development Methodology Diagram	21
7. Economic Forces Pushing Distributed Project Management	23
8. Distributed Project Management Framework 8 Steps	26
9. Distributed Project Management Impacts over SD Methodology	32
10. Employment Sector Distribution Chart	35
11. Job Classification Breakdown Chart	35
12. Years Experience in IT / IS / CS Chart	36
13. Credential Distribution Chart	36
14. Team Project Exposure Chart	36
15. Experience in Distributed Projects Chart	37
16. Current Project Management Status Chart	37
17. Software Development Methodology Distribution Chart	38
18. Collaborative Tools Usage Chart	38
19. Collaborative Tool Technologies Distribution Chart	39
20. Software Development Methodology Application	39
21. Communication Plan Documentation Existence Chart	40

22. Privacy Policy Documentation Existence Chart	40
23. Security Policy Plan Documentation Existence Chart	41
24. Virtual Trust Documentation Existence Chart	41
25. Project Management Leadership Existence Chart	41
26. Occasional Team Collocation Chart	42
27. Technical Infrastructure Existence Chart	42
28. Conflict Resolution Process Plan Documentation Existence Chart	42
29. Cultural Diversity Awareness Documentation Existence Chart	43
30. Clear Software Development Methodology Identified Chart	43
31. Ability to Change Software Development Methodology Chart	43
32. Ability to Modify Software Development Methodology Chart	44

# CHAPTER I

## INTRODUCTION

### Statement of Purpose

Software Development Methodologies are respectively interesting realms unto themselves. Over the past 50 years, there have been significant paradigm shifts in software development, such as structured programming, object-oriented, and now agent-oriented development approaches. Each evolutionary shift introduced new ways to view similar problems, as well as introducing strengths and weaknesses respectively to software development. In the pioneer days of software development the idea or presence of project management would not have been as documented or structured as it currently is in today's environment. In current times, the presence of software development projects without some sort of applied project management presents a higher probability of project failure, which is supported by the 1999 Standish Group Chaos Report, where it is reported that only 16.2% of software projects are completed on-time and on-budget, an unbelievably low percentage. [1] Although project management is applied to most development projects today, versus the pioneer days, we are still plagued with these high failure rates. The reader will find that this paper has examined particular software development methodologies (approaches) and how distance/distributed project management may impact on the software development process itself, as opposed to the overall project. Although one may be able to argue that a methodology impacted by distance/distributed project management automatically should result in the project deliverable(s) being impacted as well.

The outcomes of this paper were primarily to document current practices in software development methodologies and project management. The influence(s) that distance/distributed project management may impose on a given software development methodology has also been explored. An evaluation matrix was developed to assist software developers in determining whether distance/distributed project management impacted the respective software development methodology. At the conclusion of this paper, a summary of research findings and survey results have been provided.

#### Research Problem or Question

Study of the impacts on software development methodologies resulting from distance/distributed project management is a new area for academic research. As the distance/distributed project management approach is a young domain, and the software development methodology domain is in a continuous state of evolution, it is not surprising that the globalization of software development projects is impacting heavily on these two domains. The research conducted for this paper was based on the following two questions:

- What issues / opportunities present themselves in a particular software development methodology when managed via distance/distributed project management approaches?
- What best practices should be observed to minimize negative project impact, as it pertains to distance/distributed project management during software development?

### Assumptions of the Paper

It is assumed that a software development project will be created using only one methodology and/or approach, since it would be almost impossible to determine in a blended software development methodology project what methodology aspects, if any, were impacted by distance/distributed project management. This paper also assumes that all projects are distributed geographically beyond a city or local community, making face-to-face situations an extreme exception. As well, for the purposes of this paper, the terms distance and distributed will be synonymous, although assuming the meaning of distributed. It is also assumed that Object-Oriented software development, although technically referred to as a development approach and not a methodology, will be viewed as a methodology for the purpose of this paper.

### Significance

As we move towards a global economy, so are more of our development efforts. Therefore, understanding the impacts on software development methodology by distance/distributed software development is a step towards mastering the application of distance/distributed projects.

### Limitations

This paper references a local survey administered to a small ‘information system’ industry group. The intent of the survey was only to ascertain a local ‘state of affairs’; it was not intended to be statistically defensible data, but to be used only as an observation tool.

### Delimitations

This paper is restricted to four software development methodologies, which are: Waterfall (structured), Object-Oriented, Rapid Prototyping, and Extreme Programming. The Project Management Body of Knowledge, as put forward by the Project Management Institute, is the basis for the project management review.

### Definition of Terms

This section defines terms used in the paper and provides context for their usage:

- |                                 |  |
|---------------------------------|--|
| Distant Project Management:     | The management of a ‘project’ as defined below, where the team members are geographically separated but contained within the same organizational unit.                     |
| Distributed Project Management: | The management of a ‘project’ as defined below, where the team members are not only geographically separated but they are composed of members from multiple organizations. |

Project: Schwalbe, K. states that a project “is a temporary endeavor undertaken to accomplish a unique purpose.” [12] And continues, “constrained in different ways by its scope, time goals, and cost goals.” [12]

### Organization of Paper

This paper commences with the introduction to the topic of software development, followed by a section reviewing the current project management body of knowledge and project management life cycles. The current practices in distributed project management, as well as tools used by software development practitioners are examined in “Distributed Project Management (DPM) State of the Art” section three. While the results of an industry survey for information purposes only, and not intended as validated statistical data, are reviewed in section four, “Industry Survey”. Under the “Distributed Project Management Impact on Software Development Methodologies” section, a review of the impact(s) distance/distributed project management has had on given software development methodologies will be discussed based on a review of current literature. A conclusion summarizing research findings and presenting future questions, which could form the basis of further research, are offered up to the reader for consideration.



## CHAPTER II

### PROJECT MANGEMENT / SOFTWARE DEVELOPMENT

#### METHODOLOGY ENVIRONMENT OVERVIEW

##### Project Management Body of Knowledge Summarization

Project Management adopters claim they benefit from: better control of financial, physical, and human resources, improved customer relations, shortened development time, lower costs, higher quality, higher profit margins, improved productivity, better internal coordination, and higher moral. Logically, more structured monitoring of a software development process should result in better software development outcomes. Many information system practitioners today reference the Project Management Institutes ~ Project Management Body of Knowledge (PMBOK), as a sound approach to project management.

The PMBOK frames processes and knowledge elements, which software developers can adopt during their respective development life cycle methodology. In reviewing the PMBOK process, first we find the following project management areas: Initiation, Planning, Executing, Controlling, and Closing. Each is supported by one or many of the nine knowledge elements, identified in the PMBOK, which may or may not all be contained in each management area, but are blended throughout the entire process at different levels and stages.

Knowledge elements found in the PMBOK are contained within the first knowledge element, which is Project Integration Management. The Integration Management knowledge element focuses on the coordination between all other eight knowledge elements. Greater coordination of elements should translate to a higher probability of project success. Of the remaining eight knowledge elements, we find they are clustered in two functional groups, which are, Core Functions, and Facilitating Functions. Refer to Figure 1 for a detailed breakdown of each knowledge element and the components contained within each element.

[2]

Core Functions address 'project objectives' directly, which includes the following four project key objectives: [12]

1. Scope Management ~ Focuses on project parameters and their control.
2. Time Management ~ Focuses on the time required to complete each task.
3. Cost Management ~ Focuses on the project budget and associated resources.
4. Quality Management ~ Focuses on the project satisfying the stated / implied needs.

Facilitation Functions address 'project means', i.e. how the clients project objectives are resourced. These four project objectives to support the Core Functions are:

1. Human Resource Management ~ Focuses on the allocation of people to tasks.
2. Communication Management ~ Focuses on documenting the project's life and communicating to all project information to the appropriate group or individual.

3. Risk Management ~ Focuses on the identification, analysis, and response to project risk factors.
4. Procurement Management ~ Focuses on obtaining goods and services for the project from outside the organization.

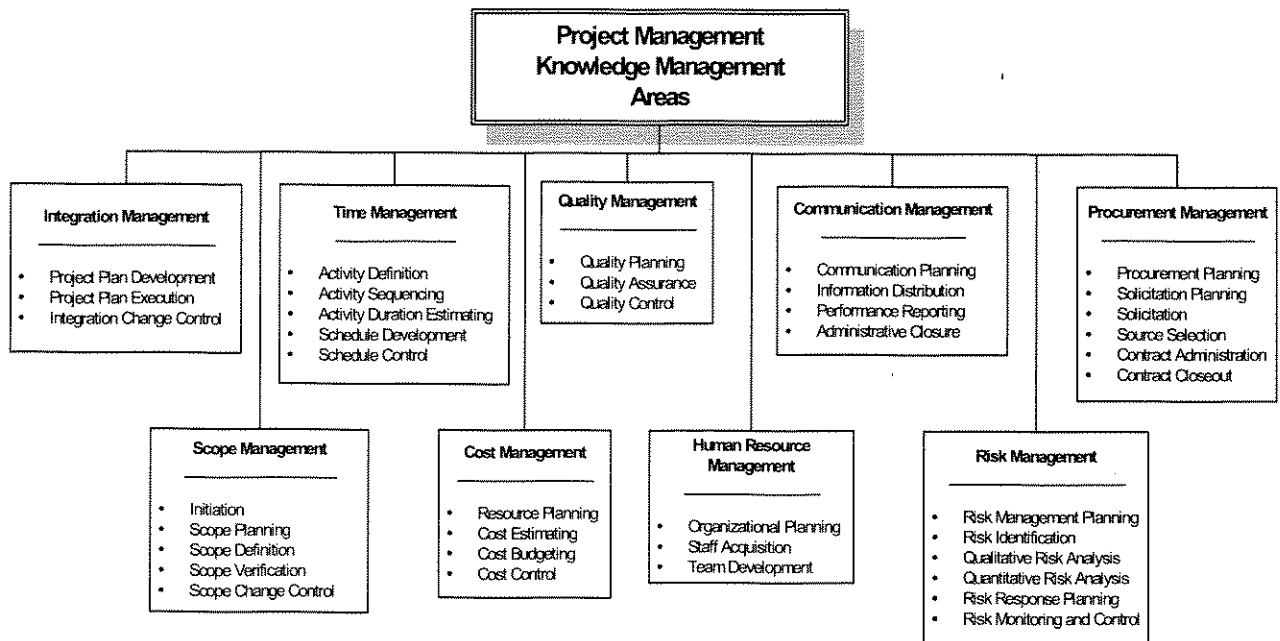


Figure 1 ~ Project Management Knowledge Management Area Content Sections

In addition to the project management knowledge management areas, we find that there are project management models, i.e. underlying approaches or philosophical frameworks, which globally govern how we apply the process of project management onto a project. The project management models are the following: [13]

1. Ad hoc ~ Characteristically department initiatives managed on an informal basis.
2. Bureaucratic ~ Typically a project found in the public sector where bureaucratic process has greater importance over project process outcomes.

3. Normative ~ Commonly a project that exist within an environment of stability capable of providing defined goals and a stable framework which allows for an organized project execution.
4. Creative-Reflective. ~ Typically projects created and executed within a complex environment that are managed by practitioners who promote life-long learning.

### Project Life Cycle Models Review

Projects being of a defined duration, composed of many interrelating pieces organized commonly into phases. These phases, when grouped together, form a project life cycle. A number of project life cycles have evolved over the past 50 years, as it pertains to software development. It is possible that they existed within other contexts or domains. As well, project life cycles may be metamorphic, which suggests they adapt to the specific domain in which they are applied. As an example, a project life cycle for a software development project will vary when compared to the same project life cycle applied to a new pharmaceutical drug development project. Foundational elements, nonetheless, would be present but the phases and sequences of phases would differ from one domain to the other. Project life cycles reviewed in this paper are: PMBOK Project Life Cycle, Straightforward, Control-Oriented, Quality-Oriented, Risk-Oriented, and the Fractal project life cycle approaches. The following text will review each of the previously mentioned project life cycles.

The Project Management Body of Knowledge Project Life Cycle may be interpreted as a generic model. A model that could apply to any project in principle, but may not be as effective given a particular project domain. This model, according to Schach, R. S., is composed of four phases, which are organized into two-phase groupings. First, the 'project feasibility phase grouping' oversees the planning and building of the software product. Second, we find the 'project acquisition phase grouping' overseeing the roll out and closure of the developed software product. Hence, within the two-phase groupings we find the following four phases: Concept, Development, Implementation, and Closeout. [3] In the concept phase, our efforts are concerned with management plans, preliminary cost estimates, and initial work breakdown structures (i.e. main tasks and key subtasks). The development phase then focuses on detailed budgetary controls and a detailed work breakdown structure. While the implementation (roll out) phase looks for definitive cost estimates and system performance reports. The closeout phase reviews the completed work, lessons learned, and customer acceptance in an attempt to capture and document the experiences for the next project. [3]

Bonnal et al. put forward five project life-cycle models with regards to managing a project, which appeared to be more suited towards technical project. The models are Straightforward, Control-oriented, Quality-oriented, Risk-oriented and Fractal, which are reviewed in the following text.

The Straightforward Project Life Cycle as described by Bonnal et al. is composed of three phases that frame the model used to manage a project. These phases are initiation,

feasibility, and implementation explained as follows. The initiation phase focuses on the concepts, economics, customers, trends, etc, in an attempt to assist in determining if the project should receive a go-ahead. Afterwards, the feasibility phase looks at organizational needs, environmental realities, and resources available to the project. It is during the feasibility phase that a project receives the go-ahead from being a proposed project request to being an assigned (resourced) project. Finally, the implementation phase will occur once the project receives official go-ahead, initiating a number of sub-phases. These sub-phases would then focus on the design, development, integration, and acceptance of the project deliverables. [2]

The Control-Oriented Project Life Cycle as described by Bonnal et al. is composed of three phases, which are planning, execution, and operation. In the planning phase the main outcomes are the project concept (i.e. need, economics, technology), project feasibility, and project definition. During the execution phase outcomes such as procurement of supplies and services, project implementation, project controlling, as well as project redefinition are the main tasks worked on. The operation phase is the official software product turnover or the point-in-time where the system leaves the test environment to be relocated on-line in the production environment with users, and clients accessing the system, to meet business needs. [2]

The Quality-Oriented Project Life Cycle as described by Bonnal et al. is composed of three phases, conceptualization, materialization, and the turnover phase. The idea behind these three phases is that while time passes during the project, we move seamlessly from one

phase to another. Beginning with the highest level of conceptualization (specifications), and as we move throughout the project timeline the level of conceptualization will decrease as we move towards the materialization phase. In materialization, we create (build) the product, hence the conceptualization level stabilizes during materialization to once again rise once materialization ends, and we the turnover phases begins. The turnover phase moves concerns from construction to reflection pertaining to the software product created. [2]

The Risk-Oriented Project Life Cycle as described by Bonnal et al. is composed of only two phases, which are pre-project and project. In pre-project we identify requirements, evaluate feasibility, evaluate environmental factors, and determine the impact of a go or no-go decision. Followed by the project phase composed of three sub components, which are: planning, execution, and closeout. [2]

The Fractal Project Life Cycle as described by Bonnal et al. is also composed of only two phases, similar to the risk-oriented project life-cycle, which are called: Pre-project and Project. In the pre-project (feasibility) phase activities center around the determination of project feasibility allowing a go/no-go decision to occur, followed by the realization of a project prototype. In the project (execution) phase the completion or refinement of the prototype, writing of the specifications, building the application, and delivering the application are the main activities. [2]

## Software Development Methodology Summarization

Development methodologies are models by which developers build solutions to a given software problem. Developers, normally academically trained, are proficient on at least two development methodologies, and have acquired industry experience in each. Ideally, developers have worked in industry developing with one of the methodologies. A traditional software development methodology, whose phases are recognized within all other methodologies, is illustrated in Figure 2 below.

This foundational approach (methodology) to a software development project begins with the definition of requirements, such as what the customer needs and/or wants coupled with what should be recommended by the information system professional determining the requirements. The next phases of the methodology guide the software developer through a system design, followed by the actual coding of the application. Afterwards, a developer's focus turns towards integrating the new system, first release or upgrade release, within the existing business systems. As the integration maps out within a test environment, the new application undergoes system and unit tests at a minimum. Upon passing the testing phase, the application is installed to the production environment, to be employed by users accessing live data. In this traditional model, each phase ends before the subsequent phase commences. Emphasis placed on large start-and-stop points in time, versus iteratively approaching all the phases of the application. A visual representation of this methodology is found in Figure 2.

<b>Requirements Analysis</b>	<b>Design</b>	<b>Code</b>	<b>Integrate</b>	<b>Test</b>	<b>Installation / Acceptance</b>
----------------------------------	---------------	-------------	------------------	-------------	--------------------------------------

Figure 2 ~ Traditional (generic) Development Life-Cycle Phases



The Traditional Methodology having been explained, the discussion can now move towards the Waterfall, Object-Oriented, Rapid Prototyping, and Extreme Programming Methodologies.

Waterfall Methodology. This methodology has similarities to the Traditional Methodology previously reviewed, but does have very important distinctions. The Waterfall Methodology was first presented by Royce in 1970 [3], indicating by the date how adolescent the domain of software development truly is. Royce documented a process (methodology) that incorporated new elements such as: documentation standards, quality assurance reviews, and progressive evolution of the system development process within the given project.

Requirements were determined as with the foundational methodology, but in contrast, this model verifies with the client that the requirements are correct. Moving to the next phase is not possible until client approval (sign-off) is obtained for the particular phase in question. Once approval is obtained, and only then, the specification phase will begin where the “what” is determined, which is what the application is actually expected to do. Once again prior to the completion of this phase, client sign-off approval is required; this process continues throughout all phases. Another important distinction between the Waterfall Methodology and the Traditional Methodology which exist is that it is permissible to revisit a previous phase in the Waterfall Methodology. The ability to revisit a phase could possibly result from: requirements initially missed, elements contradicting one another, items simply being unclear, or upon normal phase completion quality control phase outcome review. Therefore, given the above reasons for revisiting a previous phase, as illustrated in Figure 3,

one should review and update the documentation pertaining to that phase. After which a re-pass of the current phase should occur to document quality assurance of the evolving software product. Emphasizing that this methodology's distinction of not progressing onto the next phase, until all documentation of the current phase has been finalized and approved, allows for stronger controls during the software development process. Hence, a sequentially unfolding methodology, which does promote phase iteration to rework previous phases when required.

In this methodology upon moving the software from the integration/testing environment, to the systems production environment, all system changes from this point are viewed as a software maintenance effort. The software maintenance effort, as Figure 3 demonstrates, could loop the maintenance programmer back to various phases within the methodology. Where all the original documentation and software quality assurance protocols are expected as the system undergoes a maintenance revision and/or upgrade.

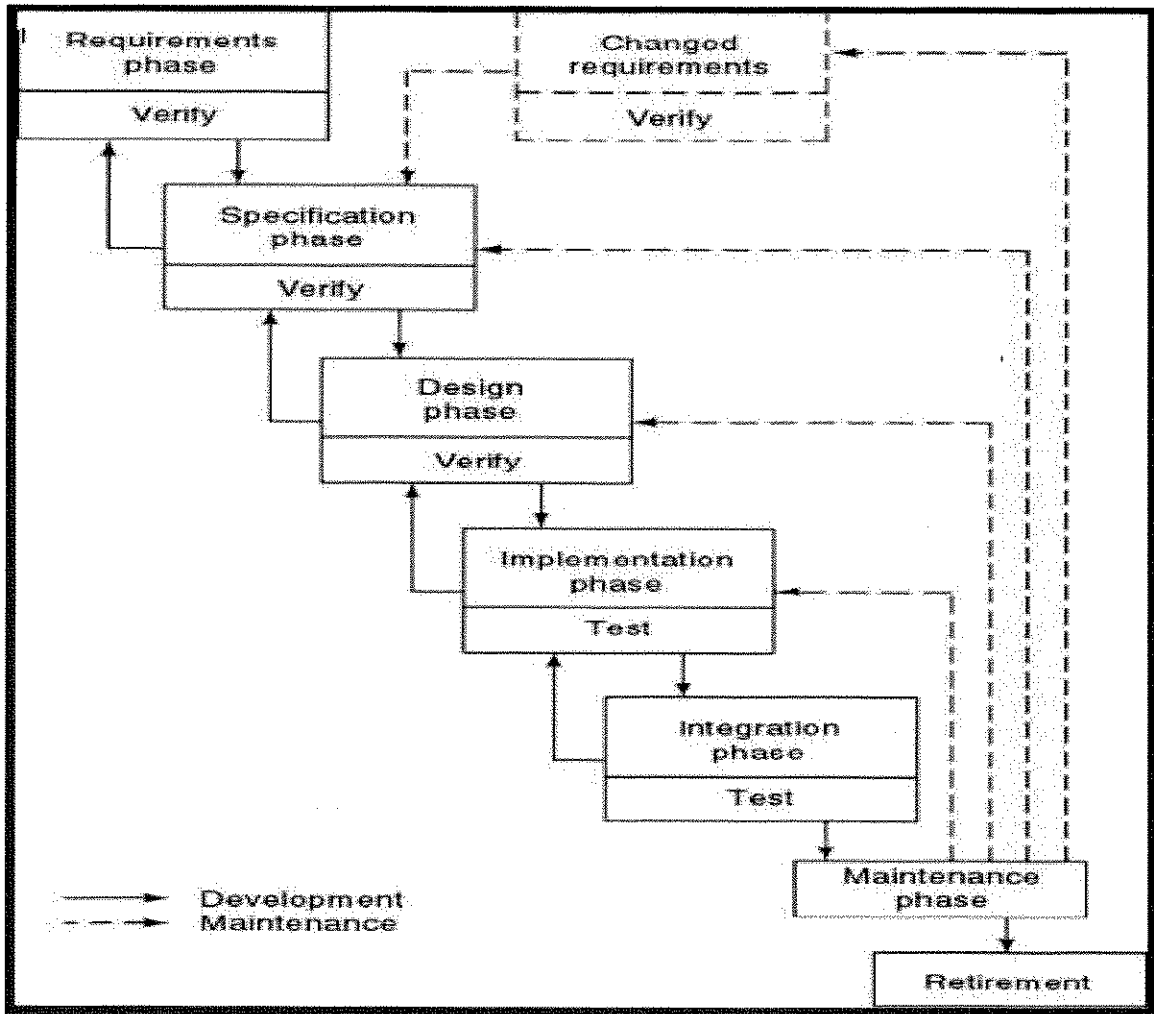


Figure 3 ~ Waterfall Development Methodology [3]

Object-Oriented Methodology. In 2001 Dr. Takuya Katayama, former professor at Tokyo Institute of Technology, defined Object-Oriented Methodology as a “Software developing methodology based on Object-Oriented Paradigm” [17]. Professor Katayama further expanded his explanation by stating that “Our world is a collection of collaborating agents/objects” and “Software has to be organized according to the structure of our world”. [17] There are a number of models supporting the object-oriented approach to software development. This paper will focus on the Fountain Model, as illustrated in Figure 4. Once again, we see the utilization of phases referred to in previously reviewed software

development methodologies, such phases are, requirements, analysis, design, implementation, integration, operation, maintenance. It is understandable that the same terminology resurfaces given that it makes sense, for example, to call the design phase 'design' in all methodologies.

Important distinctions of object-oriented versus other methodologies are that object-oriented promotes a spiral approach from one phase to another versus a linear approach. This means, for example, that within the development phase, one can theoretically restart the phase numerous times, given information that would present itself while you were in the phase. Object-oriented promotes parallelism between phases, allowing multiple phases to occur simultaneously, which means that an interface can be under design as the specifications are being defined. To take this further, one could receive user acceptance input in parallel with system construction, because today's development environments allow for a more rapid cycle from design to functional sample. Caution is required so that one does not jump, or be perceived in such a manner as jumping, from one phase to another in an un-disciplined manner. This approach is different when compared to the Waterfall approach, but still requires sound documentation and justification for actions undertaken during the iterative and parallel activities. Figure 4 below, illustrates the idea of iteration, as well as, phase parallelism approach in the execution of the Object-Oriented approach to software development.

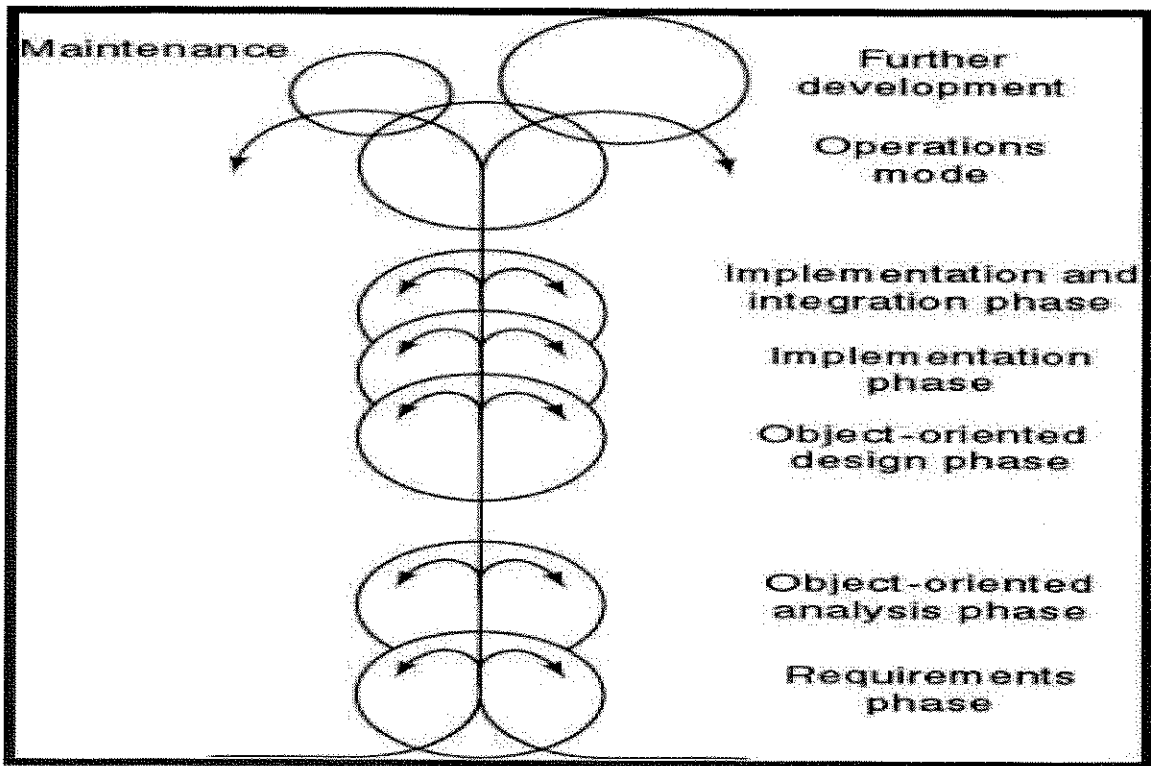


Figure 4 ~ Object-Oriented Development Methodology [3]

Rapid Prototyping. The name of the methodology defines its role clearly; build the program rapidly. Developers first assemble a functional sample of the system for the client. The reasoning here is that the client has an actual model to use and experience in order to provide feedback. The feedback from the client, someone who actually used the prototype, is invaluable to the developers. It is for this reason that the rapid prototyping methodology does not require the level of recursive steps as compared to the Waterfall methodology. The undisclosed requirements found stepping through the Waterfall methodology are for the most part identified during the single prototype review. In Figure 5, one can notice how the prototype phase has replaced the requirement phase of the Waterfall methodology and that the 'verify' loop backs are not required as client feedback has been received at the prototype phase.

It is important nonetheless to view this approach as valid and organized for software development. The creation of a prototype is not the creation of the final software product. Ideally, a prototype should not be the baseline development activity used for the actual software build. Therefore, the actual software product should be built from the initial phase, and be developed outside of the prototype code. The prototype is only utilized to obtain client feedback. As illustrated in Figure 5, once prototyping is completed, developers start moving through the specification, design, implementation, and integration phases.

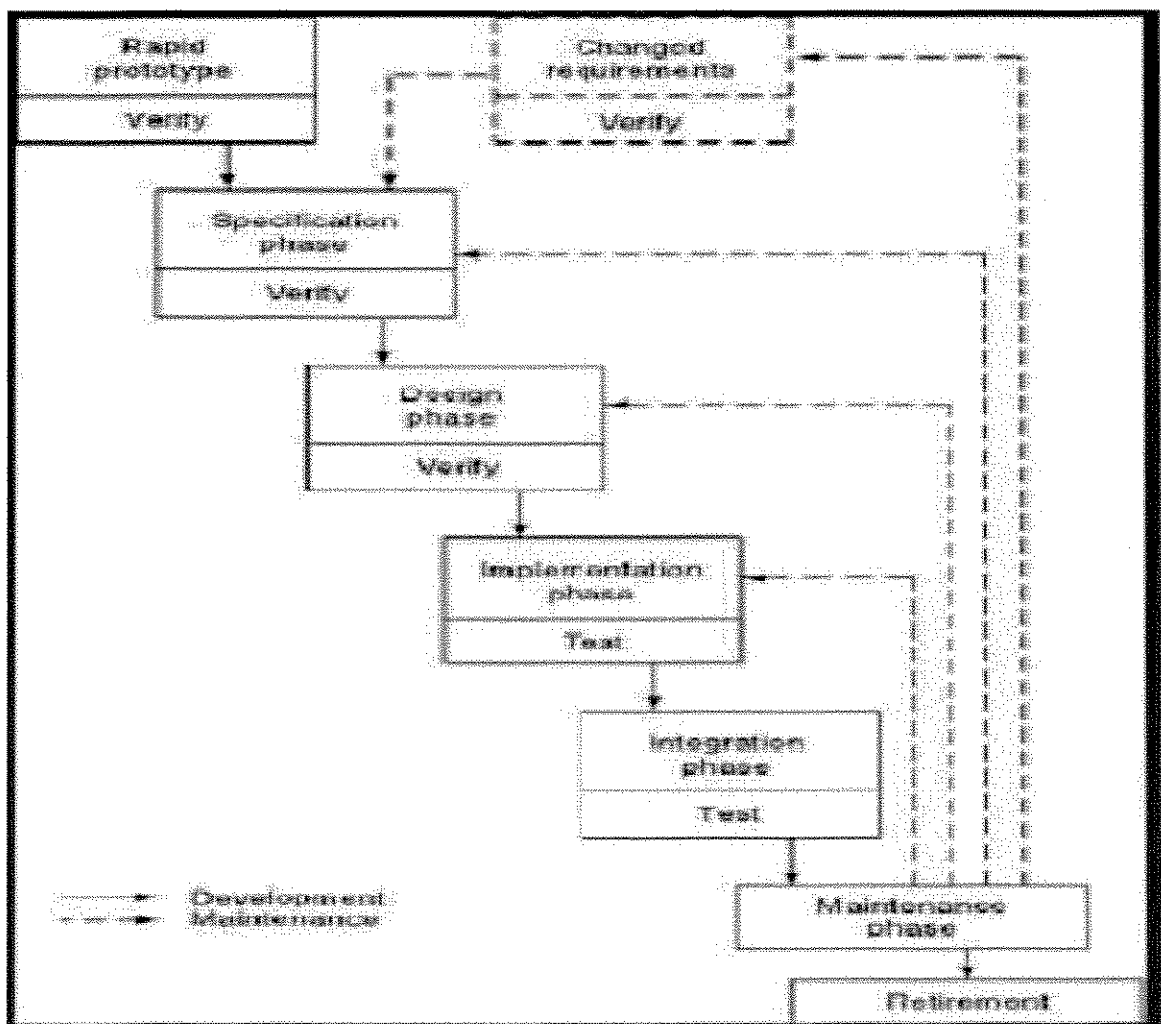


Figure 5 ~ Rapid Prototyping Development Methodology [3]

Classical Extreme Programming (XP). The Extreme Programming is referred to as a lightweight methodology by some, as it has few rules, simple practices, and is easy to implement. Considering for a moment Kanbay's explanation of software methodology as: "the set of rules and practices used to create computer programs. A heavyweight methodology has many rules, practices, and documents requiring discipline and time to follow correctly. A lightweight methodology has only a few rules and practices or ones which are easy to follow." [18] Hence, the first step taken by XP adopters is to determine the requirements that the software must accomplish. After which, the XP team selects the function(s) to be developed during the first build iteration and proceeds to do so. Four phases, illustrated below in figure 6, are used during the XP build iteration. These four phases are specification, design, implementation, and integration.

One notes that the phases are occurring as the software is under construction via multiple build iteration as illustrated in Figure 6. A comparison to this process could be a gardener who creates beautiful landscaped flowerbeds, but never designs prior to actually getting out there in the dirt creating and designing as a dynamically integrated process.

Schach expresses that XP is composed of the following unique features:

"A number of features of extreme programming (XP) are somewhat unusual:

1. The computers of the XP team are set up in the center of a large room lined with small cubicles.
2. A client representative works with the XP team at all times.
3. No Individual can work overtime for two successive weeks.

4. There is no specialization. Instead, all members of the XP team work on specifications, design, code, and testing.
5. As in the more risky incremental model ---- there is no overall design phase before the various builds are constructed. This process is termed refactoring.” [3]

As indicated above, the methodology is very different in its phase approach, i.e. multiple builds with design & code seemingly viewed as one activity. Physical logistics requirements make this methodology unique, i.e. collocation of developer in one area. Another unique aspect of this methodology is pair programming. Two programmers work on one screen as a pair, which allows for immediate human backup should one team member or programmer become unavailable.

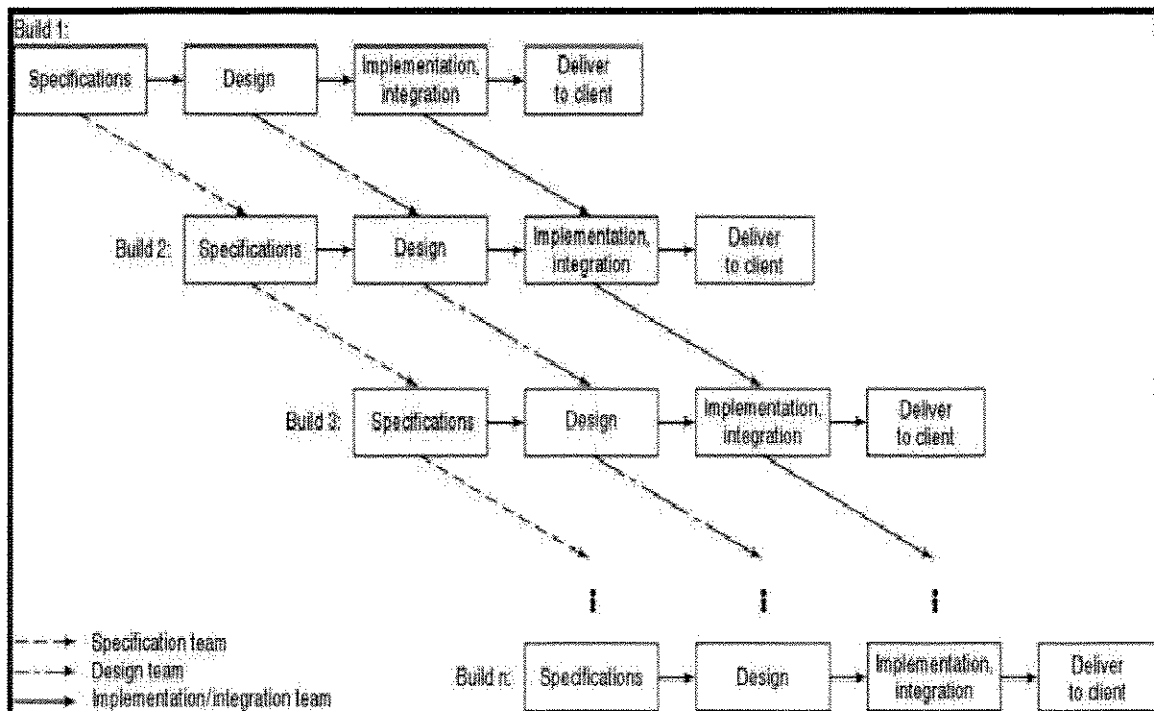


Figure 6 ~ Extreme Programming Development Methodology [3]



In the January 2002 edition of Serverworld Magazine Kanbay presented a white paper titled 'Extreme Programming (XP)' that presented XP as being composed of 12 core best practices. Listed below are the 12 practices and a brief explanation of each: [18]

1. Planning: Lists the requested features, estimates the effort required, and prioritizes
2. Small Releases: Continually supply the user with working models to obtain feedback
3. System Metaphor: Use project metaphors to create naming conventions
4. Simple Design: Keep it simple, because it could all change tomorrow
5. Continuous Testing: Similar to Test-Driven Programming, i.e. create the test case first then write the code that meets the test case specification
6. Refactoring: Allows you to capitalize on moveable components
7. Pair Programming: One machine, two programmers together, coding and reviewing
8. Collective Code Ownership: No developer owns a code section, all team members can work on all sections of the software build
9. Continuous Integrations: Changes integrate as immediately possible, i.e. at least daily
10. Forty-Hour Work Week: Developers work a normal forty-hour workweek. Only in isolated exception is up to 1 week of overtime allowed
11. On-Site Client: The client is on-site and available for consultation
12. Coding Standards: One project, one XP team, and one code standard

XP is unique and relatively new to the software development forum. Its pair programming and client on-site requirements define the methodology as one that requires a collocated team. The challenge will be to determine strategies to evolve the Classical XP model into a Distributed XP model.

## CHAPTER III

### DISTRIBUTED PROJECT MANAGEMENT (DPM) STATE OF THE ART

#### Current Distributed Project Management State of the Art Overview

As the Internet has increased its presence significantly over the past 20 years, a spin-off has been the evolution of software development project management from a fixed / single geographical location towards a globally distributed process. Numerous economic forces have pressured organizations to move toward a distributed project management forum, as it pertains to software development. A sampling of these forces, illustrated in Figure 1, are identified as the increase in market complexity for which software is developed. The human resource base becoming more specialized, therefore organizations cannot realistically have all the specialists in-house as they once may have. Compounding this situation is the global shortage of IS/IT practitioners accelerating the need to collaboratively work together in large virtual groups.

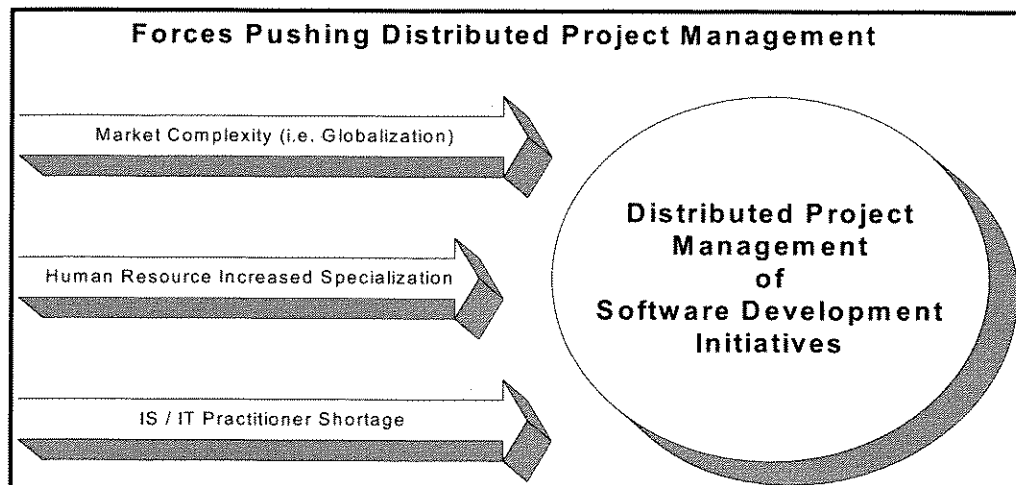


Figure 7 ~ Economic Forces Pushing Distributed Project Management

The U.S. Bureau of Labor Statistics reports that the average practitioner will change their job approximately every five years. The day of receiving '*THE GOLD WATCH*' after 30 years of service with one organization, is rapidly becoming the exception. This employment shift is occurring while organizations face an increase in project complexity and scope, demanding the required skill set to manage the complex software development project becoming more specialized. Hence, McMahon et al. found organizations are challenged in recruiting and retaining experienced staff to accomplish the numerous tasks at-hand. [4] Distributed Project Management addresses this concern by allowing organizations to globally seek out the employee with the required skill set to best meet the requirements of the task on a project-by-project basis.

An issue of the 2004 *International Journal of e-Collaboration* contained the following statement, "These so-called "virtual projects" involve people cooperating from internationally distributed sites and even different organizations. Professionals working geographically distributed participate in multi-cultural and functional projects with a global focus. These virtual projects pose new challenges to project management practitioners and researchers." [5] Bernhard et al. put forward for consideration the following project topology found in Table 1. [16]

Affiliation Dispersion of Team Members	Geographical    Distribution of    Team Members	
	Low	High
		Distributed
	Low	Traditional
	High	Inter- Organizational
		Virtual

Table 1 ~ Geographical / Affiliation Influence on Project Model

Table 1 above suggests that the lower the *geographical distribution* and *team member affiliation distribution* the more traditional the project management organization or model

will be. In contrast, the higher the geographical distribution and team member affiliation distribution, the more virtual a project management organization or model will be. Hence, in the blended approach we find distributed and inter-organizational project organization. It is plausible that there would be examples of projects that would fit within each of these four quadrants, and that a greater number of projects would move from quadrant to quadrant during a project's life cycle.

#### Review of Supporting Literature in Distributed Project Management

The November 2001 issue of the *Journal of Defense Software Engineering* presented an eight-step framework plan, as illustrated in Figure 2, to be used in assembling a distributed project, or in verifying if an existing project was assembled correctly. These steps are as follows:

1. Select a team leader with good conflict management skills and who is open minded.
2. Determine your architecture, work splits, and assign tasks.
3. Plan the individual builds and site-specific infrastructure.
4. Define / document / distribute the virtual communication rules.
5. Manage the lower level of the virtual project organization.
6. Develop a detailed plan.
7. Validate (test) the virtual operation concept of the Virtual Organization.
8. Execute the virtual project.

It is apparent that the ideas of organization and collaboration are promoted in the above list.

[4]

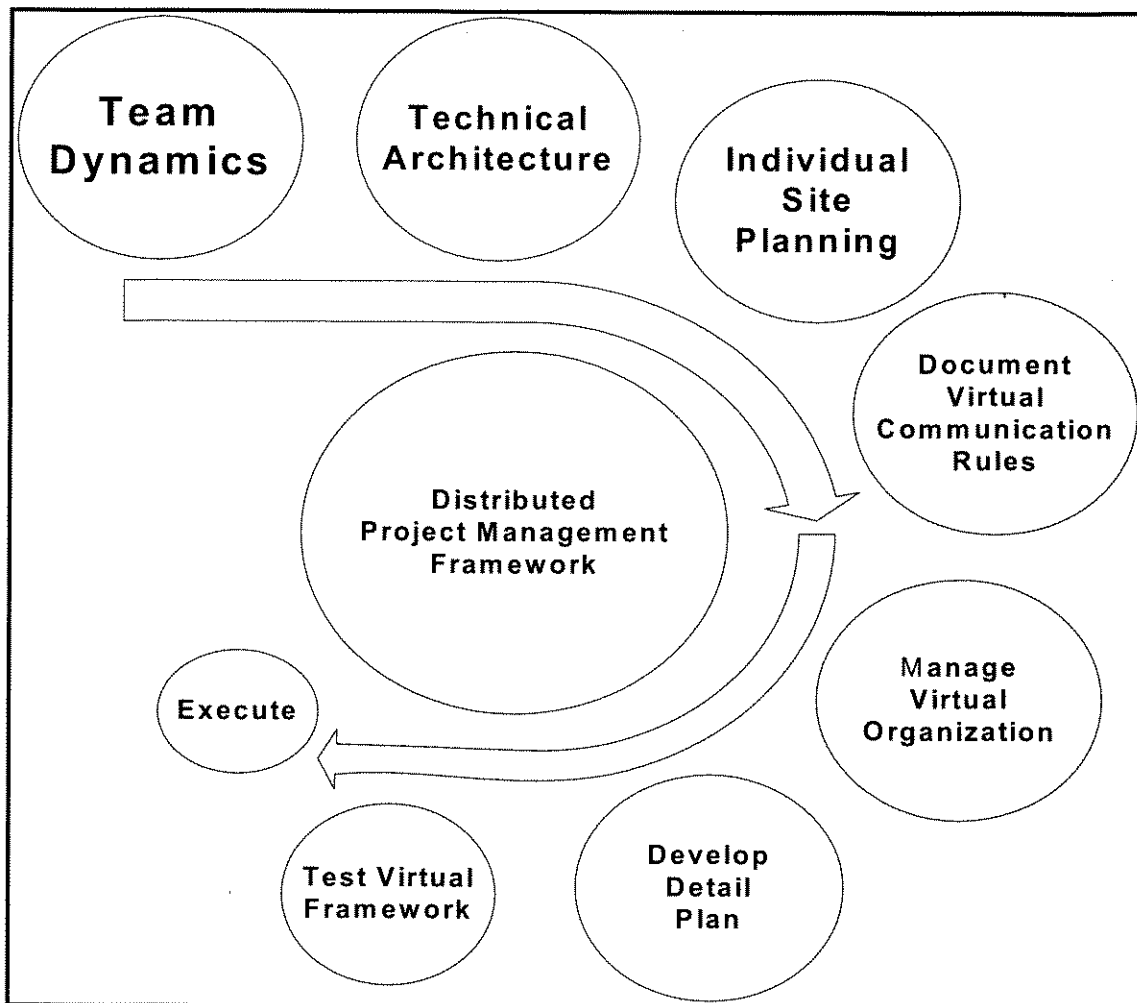


Figure 8 ~ Distributed Project Management Framework 8 Steps

Within any documented framework, opportunities and/or challenges are often experienced. In virtual project management, the following advantages exist: 1. A global software development resource pool, 2. An effective matching of resources with the required tasks, and 3. A reduction in project cost directly connected to travel. The challenges which exist in a virtual project management forum include: 1. Cultural incompatibility of project team members and/or clients, 2. Leadership struggles, 3. Lack of trust, and 4. Inherent notion

of competition. [6] Damian, D. and Zowghi, D. expanded on the Distributed Project Management Framework challenges, with the following challenges:

1. Inadequate Communication ~ Dependent on good asynchronous or synchronous tools.
2. Knowledge Management ~ Often the lack of a management system to support the project.
3. Cultural Diversity ~ Teams develop their own identity based on numerous factors, as well as working with team members from different cultural backgrounds.
4. Time Difference ~ Working in different time-zones may mean that one persons daytime is another's nighttime. [7]

Salisbury University Professor, Dr. Catherine M. Beise, articulates that Distance/Distributed Project Management is a dynamic undertaking when she states "the globalization of project teams has increased demographic and cultural diversity, which can create obstacles to the smooth functioning of team processes, but also can provide benefits in creativity, innovation, and problem-solving." [8] This globalization of projects has created the need for virtual teams who evolve their own cultures.

Virtual projects introduce virtual team cultures, which are product oriented. A virtual team culture is a grouping of multiple teammates, from varying socio-economic and/or cultural backgrounds, developing relationships and sharing experiences and beliefs, thus establishing a team culture. With this virtual team culture, the following components could conceivably be identified:

1. An Organization Operation Concept.
2. A Build Plan.
3. Architecture Definitions.
4. Process Definitions.
5. Virtual Communication Rules.
6. Site-Specific Infrastructure Information.

It is important that virtual cultures not disregard the positive aspects of local community cultures, but embrace the strengths and benefits of local cultures, while operating within a virtual structure. McMahon, P. states that, "Experience indicates that an effective virtual culture cannot be informal." [4] This statement expresses clearly that communication must be documented in a formal manner, and that team members must respect communication protocols.

#### Exploration of Distributed Project Management Tools

Organizations such as The Program Management Group state that "It's true there are many software tools out there that enable web-based team working by providing an environment that supports the access and sharing of information and limited communication facilities." [9] The challenge is to match the appropriate tools to the appropriate individuals, processes, tasks, environments, etc., for if not appropriately matched, the project will face increased challenges. There are a myriad of tools available today, which support a distributed, virtual, project undertaking. Both management systems and communication systems provide key support to project managers attempting to manage a virtual project.

Badir et al. summarized from Weiss & Thamhain 2001 that: “Managing GLSPs effectively differs from traditional large-scale projects because time, distance, and dependence on communication technologies in decision-making adds complexity to interactions between project members”. [10]

They also proceeded to suggest that a Global Large Scale Project (GLSP) is best managed via certain types of management systems. The first of these systems to be examined is the web-based project management system, which is capable of planning, monitoring, scheduling, and budgeting a project. A tool such as this would allow for the collection, storage, manipulation, and communication of information to the entire project team. The second element of this model is a data base management system (DBMS). This DBMS could be either a relational or an object-oriented data base environment, regardless, it would store information pertaining to the project, in addition to being programmed to reply to various queries. The third element of the model is a Work Flow Management System. “The workflow technology has been reported to be effective in specifying, executing, and coordinating the flow of tasks within a distributed environment” [10] stated Badir et al. A fourth element used in applying distributed project management is a relatively new technology that is an agent-based system. An intelligent agent object is an autonomous application, or part of an application, that has the ability to problem-solve without being prompted by the human element making up the entire system process. There are other implementations of this new and exciting technology. The intelligent agent most recognized is the Microsoft Paper Clip. In distributed project management, the application of agent-oriented technology according to Badir et al. is viewed as “a workflow agent knows all the



tasks that must be done, and the time of their execution. Therefore, agents search for subjects capable of executing the next task. When a subject is found, the agent moves directly to it and provides the data items needed so the subject can fulfill the task.” [10]

Nienaber & Cloete put forward the following example of the application of agent based technology. They suggest that an agent team could be used for the communication management aspects of a distributed project management undertaking. The agent team would address identified functions, by employing specialized software agents. Currently the following specialized software agents are available: [11]

- Messaging Agent: Responsible for inter-agent message transportation.
- Personal Assistant (PA) Agent: Responsible for assisting individuals accomplish tasks.
- Task Agent: Invoked by the PA Agent and monitored by the Monitoring Agent, this agent supports a specific task.
- Monitoring Agent: Responsible for monitoring and reporting schedule and rescheduling of task information.
- Client Agent: Interacts with other agents, but are for a specific task normally, such as information retrieval.
- Team Manager Agent: Responsible for managing the agent team activities.

There are a host of tools supporting the distributed project management efforts, electronic mail, bulletin boards, MSN type tools, teleconferencing, videoconferencing, and project management software tools of one-tier or two-tier infrastructure. All of which

increase the dynamics of a project's life cycle. Bernhard et al. stated: "Increases in the sophistication of communication technology, coupled with geographical dispersion of the organization's workforce, provide the impetus to move from traditional to distributed projects." [16]

In considering the key aspect of Distributed Project Management, as well as the cycles of Software Development Methodologies, it is suggested that Distributed Project Management practices will impact Software Development most dramatically in the preliminary stages of the software development activities. It is during these first phases where communication is the most critical during traditional (collocated) projects. Projects with everyone in a single location versus everyone distributed globally would allow one to hypothesize that these critical preliminary steps become severely at-risk in a distributed environment. Similarly, a distant student could be at greater risk of experiencing learning challenges, if compared to a campus-based student, during the initial learning cycles where critical patterns/information are communicated.

Illustrated below in Figure 3 are aspects of Distributed Project Management that are suggested as potentially imposing an influence, over a particular Software Development Methodology. As is suggested in Figure 3, some aspects have a high or moderate degree of impact, while some aspects are impact neutral.

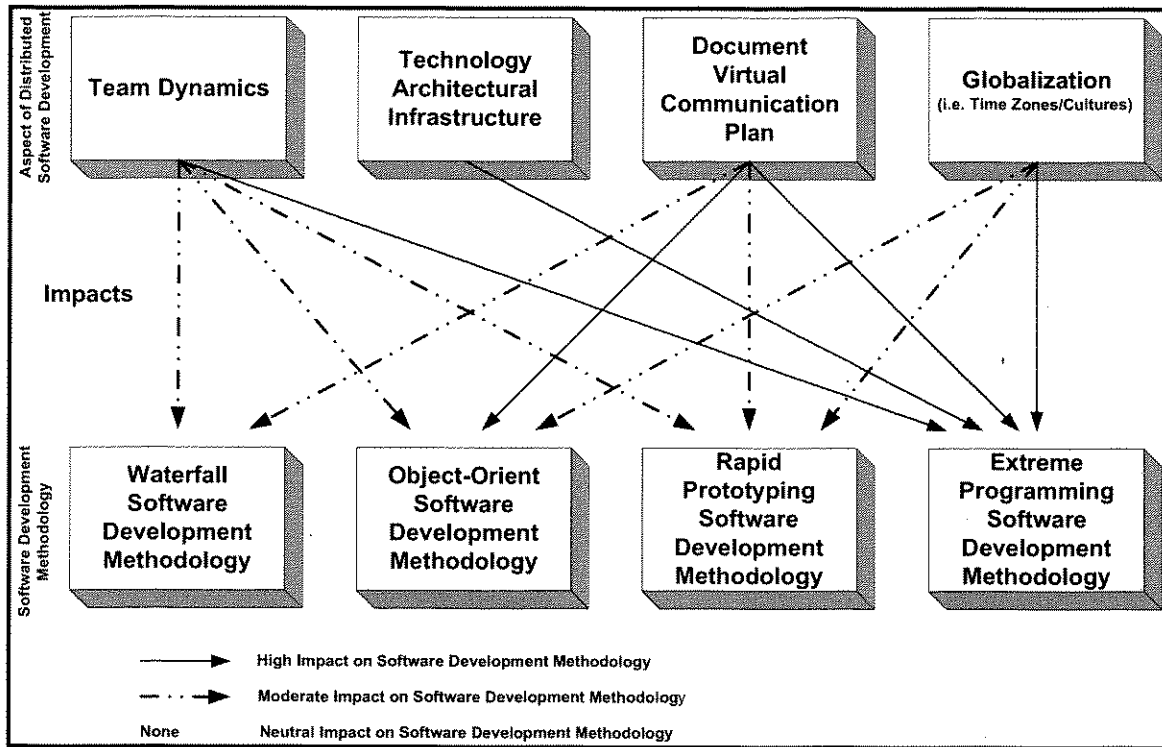


Figure 9 ~ Distributed Project Management Impacts over Software Development Methodology

According to Figure 3, (see above) the Extreme Programming Methodology is possibly the most impacted, by all aspects of Distributed Project Management, while the Waterfall Methodology is possibly the least impacted. Because the Waterfall Methodology is the oldest of the four reviewed methodologies, it has potentially the greatest ability to absorb aspects of Distributed Project Management without being negatively impacted. Conceivably this is primarily due to the collective experience of current IT and/or IS practitioners with the Waterfall Methodology and their ability to transfer their learned experiences from previous collocated projects to the Distributed Project Management venue.

## **CHAPTER IV**

### **INDUSTRY SURVEY**

#### Scanned Environment Review

A survey, of local industry, sites was administered. In an attempt to ascertain what the current regional (Atlantic Canada) exposure or experience to this paper's questions might be. The industries contacted are categorized into the following groupings:

- A. Computer Consulting Firms (7 sites)
- B. Corporations (8 sites)
- C. Government (3 sites)

With each industry, individuals possessing a variety of technical credentials completed the survey. The survey is designed to capture observations by these respective individuals on the current utilization and/or awareness of distributed project management on software development methodologies.

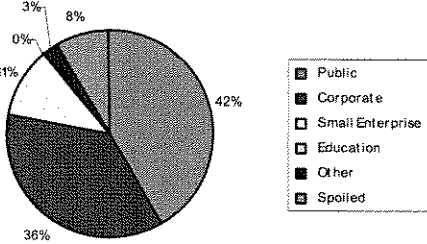
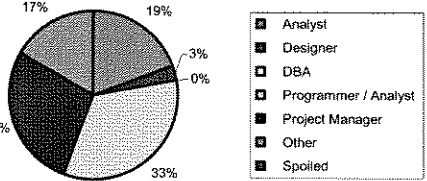
#### Data Collection Techniques

The quantitative data collection approach was applied in collecting data for the survey. A copy of the survey is located in Appendix A of this paper. Computer firms, corporations with information systems departments, and government information systems departments, sites contacted were deliberately gravitated toward, because private industry, it is assumed should not be as burdened by bureaucratic process. The survey circulated to key contacts within each site, requesting them to circulate throughout the respective information

systems department areas, to be completed and returned. It is assumed of the 18 sites contacted a possible 180 surveys could have been circulated and returned. Only 37 surveys were returned translating to a 20.5 % respondent rate, which is a low data base from which to draw solid supportive data for arguments put forward in this paper. However, what the survey does do is provide a perspective of the current state of affairs from an IS/IT practitioner group viewpoint.

### Data Analysis

Below are detailed reviews and observations of data obtained from the survey. It is important to note that this data should not be interpreted as statistically defensible, but only a scan that provides information on the current state of affairs within the sampling group. The intent of the survey is to examine the common practices pertaining to distributed project management and to ascertain what software methodologies are being applied within industry. Nonetheless, the results do show what and how these practitioners are applying within their respective software development activities. Therefore, the results can be interpreted as informal observations.

Section A ~ Experience / Environment																	
<p>The survey group is composed of more private sector contacts, however, according to the tabulated results more public sector survey participants (42 %) responded to the survey. Corporate respondents (36 %) make up the second largest contact grouping.</p>	<div data-bbox="889 449 1393 932"> <p><b>Employment Sector</b></p>  <table border="1"> <thead> <tr> <th>Employment Sector</th> <th>Percentage</th> </tr> </thead> <tbody> <tr> <td>Public</td> <td>42%</td> </tr> <tr> <td>Corporate</td> <td>36%</td> </tr> <tr> <td>Small Enterprise</td> <td>11%</td> </tr> <tr> <td>Education</td> <td>8%</td> </tr> <tr> <td>Other</td> <td>3%</td> </tr> <tr> <td>Spoiled</td> <td>0%</td> </tr> </tbody> </table> </div> <p>Figure 10</p>	Employment Sector	Percentage	Public	42%	Corporate	36%	Small Enterprise	11%	Education	8%	Other	3%	Spoiled	0%		
Employment Sector	Percentage																
Public	42%																
Corporate	36%																
Small Enterprise	11%																
Education	8%																
Other	3%																
Spoiled	0%																
<p>The largest grouping of respondents is the programmer / analyst group, followed closely by the project manager group. This was expected based on the survey administration directions. It is hypothesized that the ‘other’ sampling is composed of management personnel given its size (17 %).</p>	<div data-bbox="889 1044 1393 1527"> <p><b>Job Title/Duties</b></p>  <table border="1"> <thead> <tr> <th>Job Title/Duties</th> <th>Percentage</th> </tr> </thead> <tbody> <tr> <td>Analyst</td> <td>0%</td> </tr> <tr> <td>Designer</td> <td>19%</td> </tr> <tr> <td>DBA</td> <td>0%</td> </tr> <tr> <td>Programmer / Analyst</td> <td>33%</td> </tr> <tr> <td>Project Manager</td> <td>28%</td> </tr> <tr> <td>Other</td> <td>17%</td> </tr> <tr> <td>Spoiled</td> <td>0%</td> </tr> </tbody> </table> </div> <p>Figure 11</p>	Job Title/Duties	Percentage	Analyst	0%	Designer	19%	DBA	0%	Programmer / Analyst	33%	Project Manager	28%	Other	17%	Spoiled	0%
Job Title/Duties	Percentage																
Analyst	0%																
Designer	19%																
DBA	0%																
Programmer / Analyst	33%																
Project Manager	28%																
Other	17%																
Spoiled	0%																

It is highly probable that the sampling group is a senior group with (36 %) of respondents having 8 - 15 years experience, and (33 %) having over 15 years experience. Given the seniority of the group this raises concern, as they may not be open adopters of distributed project management models.

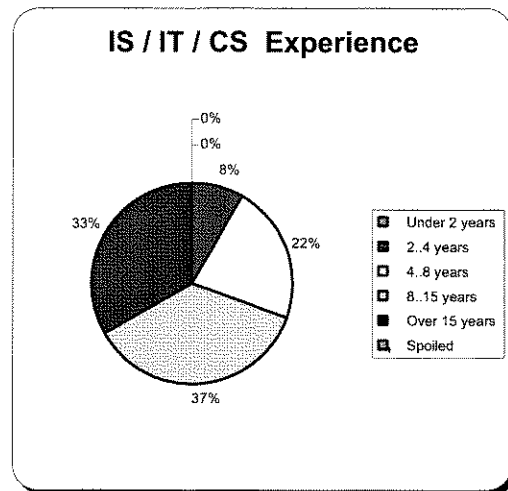


Figure 12

It appears that the sampling of this group is balanced with respondents possessing Diploma credentials at a rate of (44 %) of respondents, and Degree credentials (41 %).

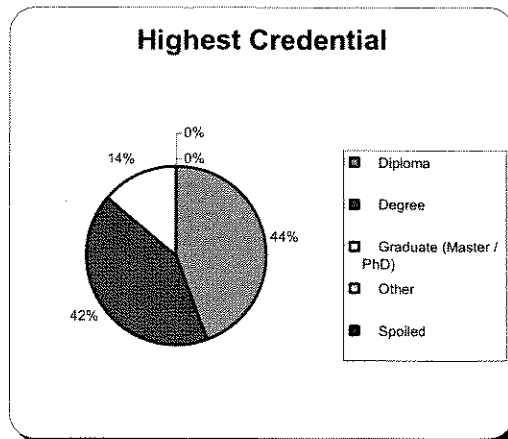


Figure 13

As anticipated, given that project work is such a large component of the IS / IT / CS professions, the majority of respondents had worked on team projects.

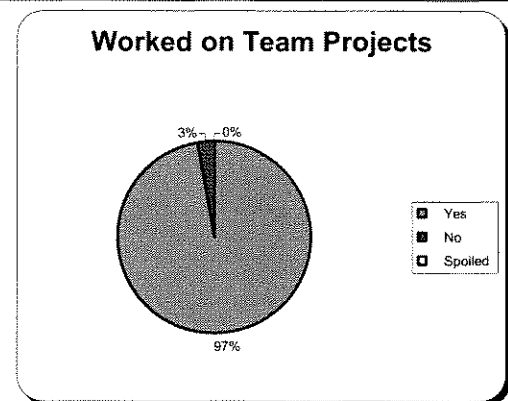


Figure 14

The results would indicate that the majority of respondents work, or have worked on geographically distributed projects, however, they do not indicate how often or for what duration.

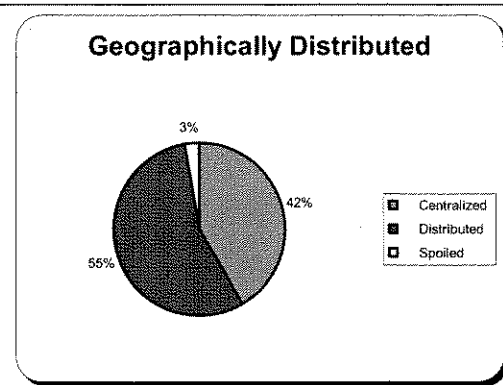


Figure 15

## Section B ~ Project Management / Software Development

Respondents indicate here that they are currently involved in the application of project management during their software development activities. They do not specifically report that they are in a distributed application, but conceivably, distributed or not, the respondents would be applying project management structures at their respective locations.

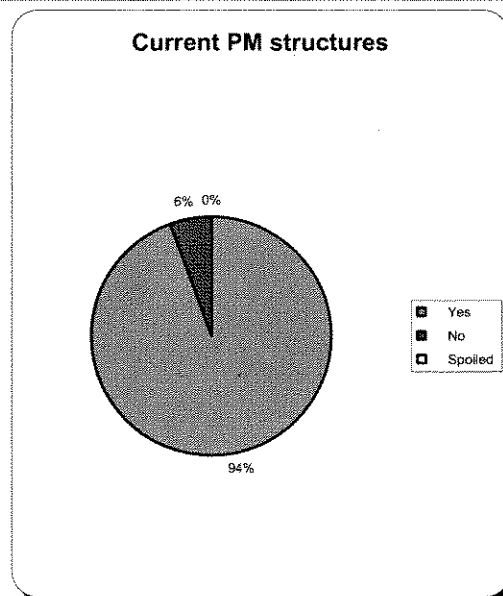


Figure 16



With approximately almost (60 %) of respondents holding 4 - 15 years experience it is logical that object-oriented (27 %) is the most applied methodology, with Waterfall being employed at a rate of (24 %). What is unanticipated is that (30 %) of respondents use other types of methodologies, than those listed within this survey.

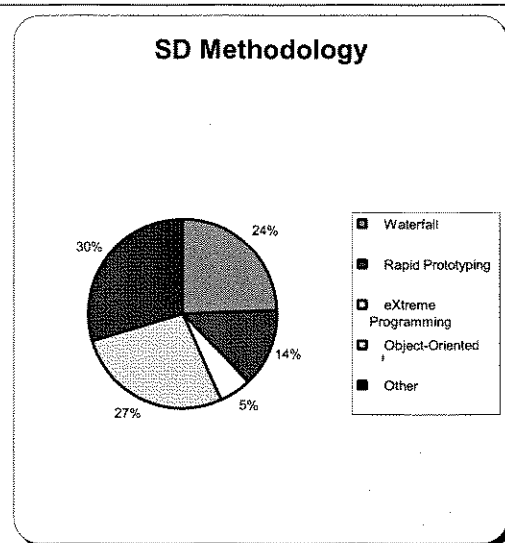


Figure 17

The findings that (97 %) of respondents apply project management as part of their software development process indicates that they are collaborating. Therefore, it is consistent that (89 %) of respondents report using collaborative tools.

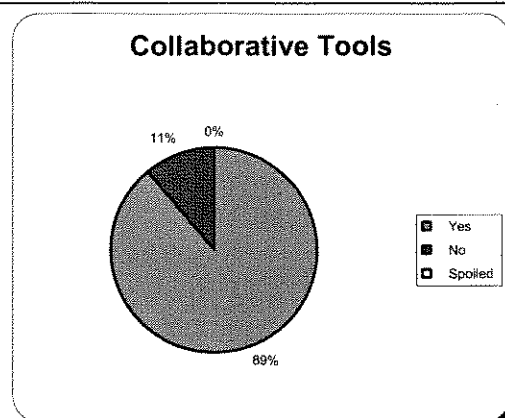


Figure 18

With respondents (97 %) applying project management it is consistent that e-mail (25 %) would be the highest collaborative tool used. Followed by PM Software (22 %), teleconference (18 %), and videoconference (9 %). Web Project Management tools at (3 %) has the lowest utilization statistics.

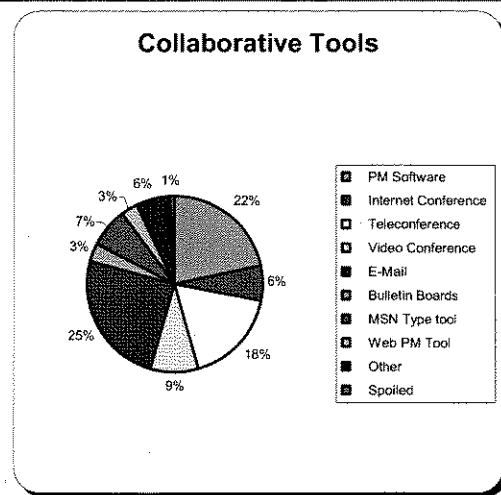


Figure 19

### Section C ~ Distributed Project Management Practices

(48 %) of respondents report that they are able to stay 'Mostly' within their adopted software development methodology. While (30 %) of respondents report 'Somewhat' and only (8 %) report 'Fully'. This may indicate that methodologies are impacted to varying degrees by distributed project management given that only (8 %) indicated they were able to stay fully within the software development methodology.

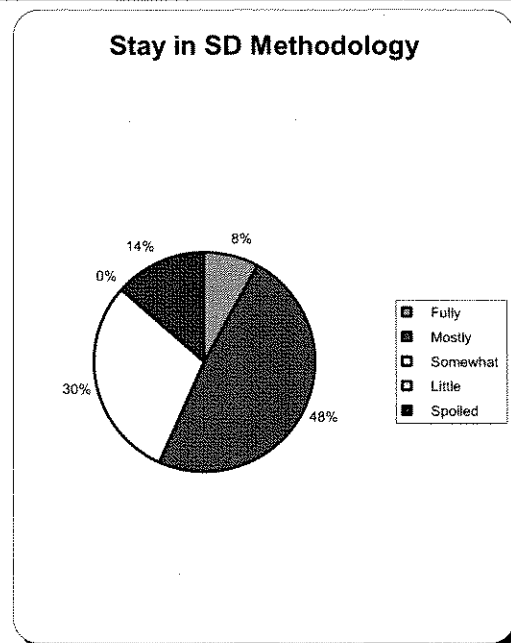
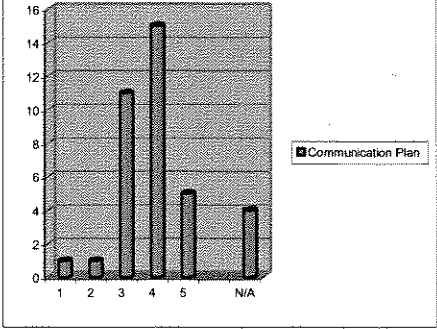
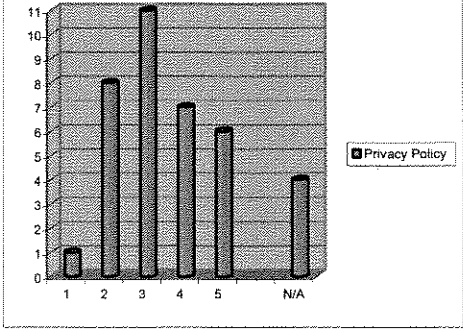


Figure 20

	<p>Number of respondents indicating the existence level of the identified criteria, while using Distributed Project Management</p> <p>Scale: Never ..... Always N/A</p> <p>1 2 3 4 5 n/a</p>
<p>Respondents indicate the following:</p> <p>6 % reporting 1 or 2</p> <p>30 % reporting 3</p> <p>55 % reporting 4 or 5</p> <p>9 % not applicable</p> <p>Scale: Never .....Always N/A</p> <p>1 2 3 4 5</p>	<p><b>Communication Plan</b></p>  <p>Figure 21</p>
<p>Respondents indicate the following:</p> <p>25 % reporting 1 or 2</p> <p>30 % reporting 3</p> <p>35 % reporting 4 or 5</p> <p>10 % not applicable</p> <p>Scale: Never .....Always N/A</p> <p>1 2 3 4 5</p>	<p><b>Privacy Policy</b></p>  <p>Figure 22</p>

Respondents indicate the following:

19 % reporting 1 or 2

19 % reporting 3

52 % reporting 4 or 5

10 % not applicable

Scale: Never .....Always N/A  
1 2 3 4 5

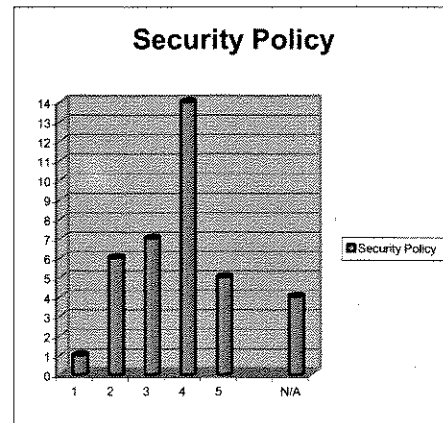


Figure 23

Respondents indicate the following:

17 % reporting 1 or 2

19 % reporting 3

35 % reporting 4 or 5

29 % not applicable

Scale: Never .....Always N/A  
1 2 3 4 5

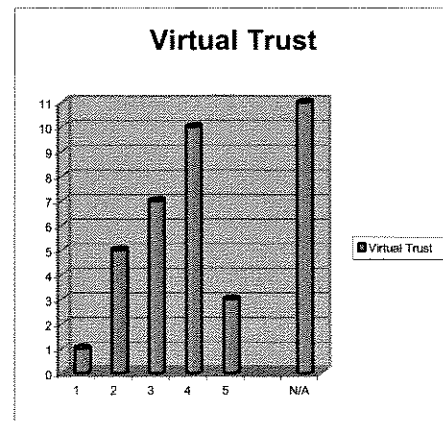


Figure 24

Respondents indicate the following:

11 % reporting 1 or 2

22 % reporting 3

56 % reporting 4 or 5

11 % not applicable

Scale: Never .....Always N/A  
1 2 3 4 5

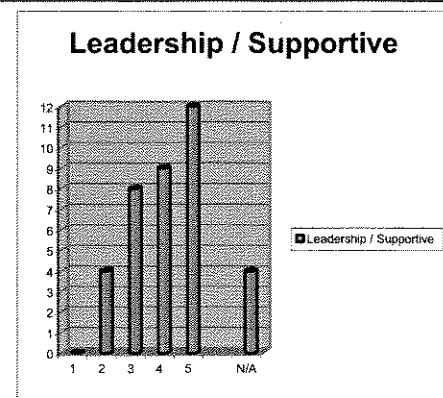


Figure 25

Respondents indicate the following:

13 % reporting 1 or 2

24 % reporting 3

49 % reporting 4 or 5

14 % not applicable

Scale: Never .....Always N/A  
1 2 3 4 5

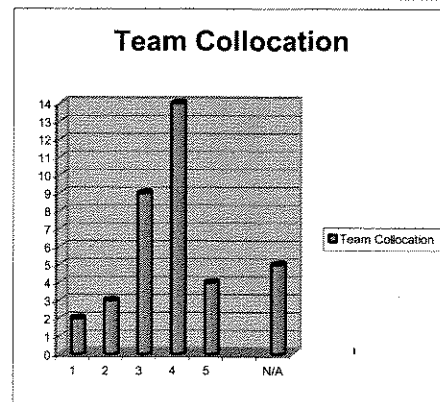


Figure 26

Respondents indicate the following:

8 % reporting 1 or 2

22 % reporting 3

60 % reporting 4 or 5

10 % not applicable

Scale: Never .....Always N/A  
1 2 3 4 5

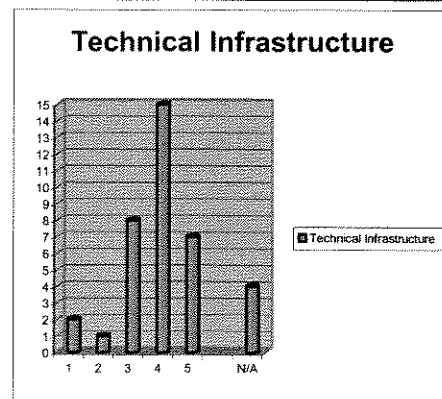


Figure 27

Respondents indicate the following:

17 % reporting 1 or 2

38 % reporting 3

30 % reporting 4 or 5

15 % not applicable

Scale: Never .....Always N/A  
1 2 3 4 5

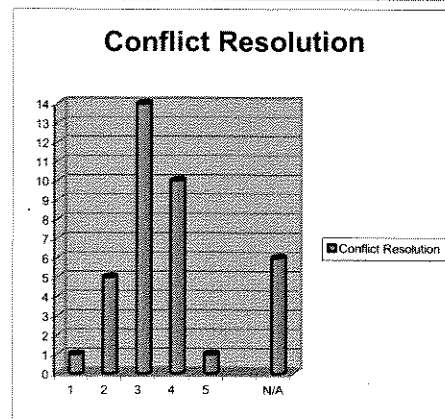


Figure 28

Respondents indicate the following:

38 % reporting 1 or 2

22 % reporting 3

25 % reporting 4 or 5

15 % not applicable

Scale: Never .....Always N/A  
1 2 3 4 5

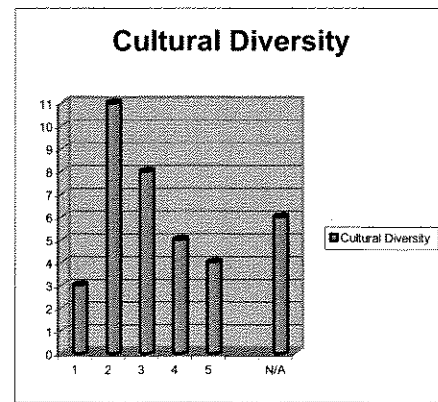


Figure 29

Respondents indicate the following:

17 % reporting 1 or 2

30 % reporting 3

44 % reporting 4 or 5

9 % not applicable

Scale: Never .....Always N/A  
1 2 3 4 5

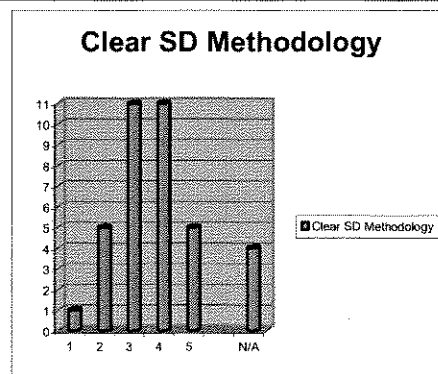


Figure 30

Respondents indicate the following:

27 % reporting 1 or 2

24 % reporting 3

38 % reporting 4 or 5

11 % not applicable

Scale: Never .....Always N/A  
1 2 3 4 5

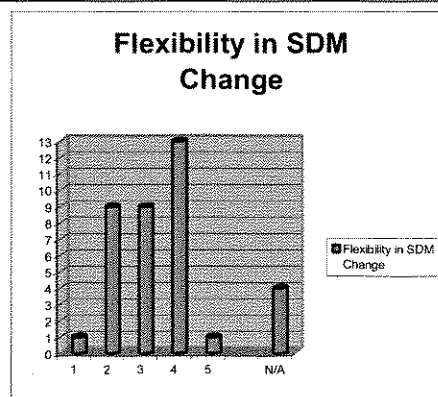
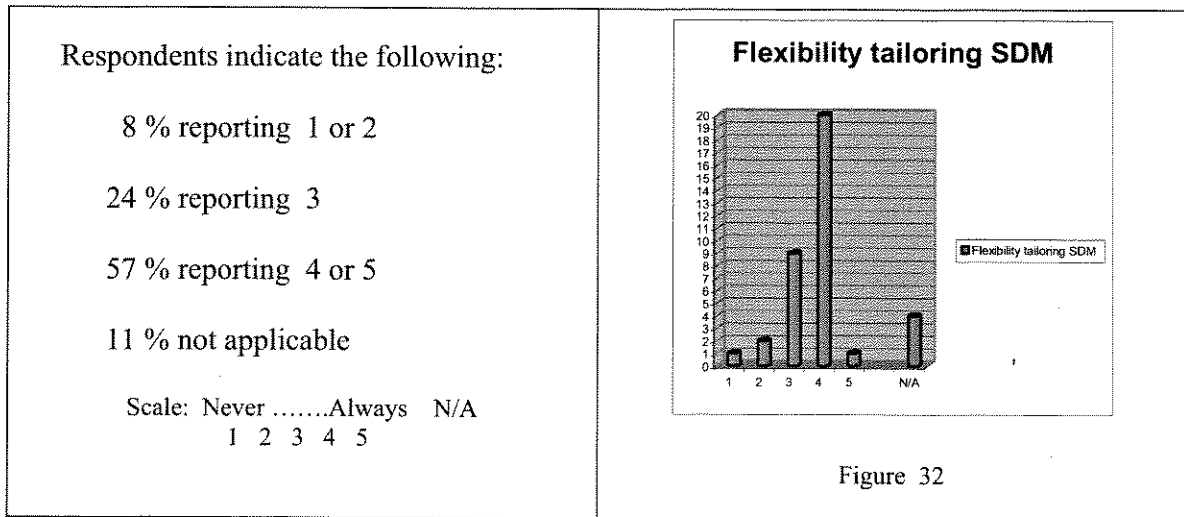


Figure 31



### Survey Findings Summarization

Based on the survey results, all respondents work, or have worked on team projects with half of the sampling group having experienced distributed projects. The group appears to be well credentialed, with the large majority possessing at minimum four years experience in the IS / IT / CS domains. Software development methodologies remain within the two most common methodologies, Waterfall and Object-Oriented. One third of the sampling group reported using other types of methodologies other than the four listed on the survey. A significant majority of the group reported having used collaborative tools; e-mail ranked the highest with regards to usage. Based on data from Section C of the survey it is determined that aspects of distributed project management were not present in the environment of over 15% of respondents. The aspects identified by the respondents as not normally being part of the distributed project management processes are the following: privacy policy, security policy, virtual trust, conflict resolution, cultural diversity, clear

software development methodology, and flexibility in changing software development methodology.

Further analysis of the survey results suggest that for respondents coming from government or larger corporations, budgetary considerations and resources may be less of an issue given the size of the organizations. This is interpreted from the statistics which indicate that 62% have a communication plan, 62% have a security plan, and 70% express the existence of a technical infrastructure. These three categories are indicative of a stable and structured organization. In analyzing respondent feedback pertaining to collaborative tools, indications are that 25% of respondents rely on e-mail as their primary collaborative tool. This high e-mail usage is a result of one or a combination of the following three possible scenarios: ease of availability, historical experience by practitioners, and/or low availability of Web Project Management tools, which appears to be supported by respondents indicating that only 3% use Web Project Management tools.

From a Distributed Project Management collocation effort, 63% of the respondents, responding with a four or five, indicating that they value occasional face-to-face collocation gatherings as an important element of Distributed Project Management. This is a strong indicator that our social and cultural experiences remain rooted in physical interaction, and that our evolutionary shift to a virtual interaction comfort zone is still evolving. A larger, more extensive sampling would be required to derive further defensible arguments from the survey results.



With 48 % of the respondents, responding with a four or five, able to remain true to the implemented Software Development Methodology. It would be premature in nature to claim a positive or a negative position regarding industry's ability to remain true to a methodology. What can be ascertained is that only 3 % of the respondents indicate that they use the Extreme Programming Methodology, which is the methodology data suggests has the highest probability of being impacted by Distributed Project Management practices. Two conceivable rationales for this low percentile use of Extreme Programming are the relative newness of the methodology, and/or industry's self-diagnosis that this methodology may be difficult to implement within a distributed environment.

A larger sampling would be necessary to derive solid and defensible data from a survey of this type. What can be derived from the survey results are soft indicators of what is transpiring within a small sector of the local technology industry.

## CHAPTER V

### DISTRIBUTED PROJECT MANAGEMENT IMPACT ON SOFTWARE DEVELOPMENT METHODOLOGIES

#### Impact on Software Development Methodology by Distributed Project Management

Software development can be described by software developers as a complex undertaking in ideal conditions; an undertaking that finds itself continuously at risk from external and/or internal factors which may impact on a project's successful completion. Even in the most traditional development approaches, i.e. all employees working from one location, project after project can be affected negatively for a multitude of reasons. According to the 1999 Standish Group Chaos report, projects incur negative impacts due to the following factors: lack of user input, incomplete requirements and/or specifications, changing requirements and/or specifications, lack of executive support, technology incompetence, lack of resources, unrealistic expectations, unclear objectives, unrealistic time frames, and the introduction of new technology. [1] These factors are listed in order of highest to lowest negative impact.

Current methodologies in software development provide a varied repertoire from which programmers may choose when tasked with matching system functionality with suitable development methodology. The industry survey reports the following breakdown of methodology usage within the five methodologies offered: Waterfall (24%), Object-Oriented (27%), Rapid Prototyping (14%), Extreme Programming (5%), and Other (30%). The small industry utilization of Rapid Prototyping and Extreme Programming complicates the ability

to draw specific impact indicators on the respective methodologies. The low industry application of these methodologies suggest that practitioners have self-determined that these methodologies would be more challenging to utilize in a distributed environment.

While Software Development Methodologies have a greater documented record of accomplishment, Distributed Project Management is a relatively new model, appearing in software development mainstream in the last ten years, therefore, still an evolving domain. With the domain still in a stage of infancy, to date, best practices have not been ascertained. This, coupled with the reality that computer science, information systems, and information technology domains are moving forward rapidly with regards to available technologies, making it increasingly daunting for practitioners to maintain currency with available software technologies and their respective applications.

In examining potential impacts on development methodologies overall, it is arguable that the following factors influence the entire process, therefore making it difficult to isolate what area is specifically impacted the most, the software development process or the distributed project management process. The four areas of influence discussed here are not intended to be a comprehensive list of areas influenced, but are put forward as key areas that could most strongly influence the methodology and/or project process.

The first area of influence put forward for review is communication. Communication is integral to a successful project outcome, and currently there are various technologies to allow both synchronous and asynchronous communications. It could be surmised that given

a project's particular stage, decisions to utilize one communication mode, synchronous and/or asynchronous, over another would be best or even required to better secure a positive project outcome. Regardless of the communication mode issues, language (translation requirements), technology speed and/or compatibility, to name but two will in all likelihood present themselves continuously, or at least intermittently during different tasks of a project. Impacts on methodology are not solely limited to technical or cultural issues. One could propose that in a distributed environment it is easier to 'not hear' the voice of a distant team member or virtual project manager. In this situation, no process or technology will rescue the development project; individuals must apply an overall sound professional work ethic toward their approach in working together. As well, they must actively listen to all team members and/or the project manager and assertively maintain clarification practices to ensure success in project efforts. It is critical that the entire team, whether physically or virtually present, be vigilant in communicating effectively.

With today's globalization of economies and work forces, we find an increase in the occurrence of projects that span multiple borders, countries, and continents; global projects spanning great distances, and encompassing multiple organizations, encounter the challenge of multiple time zones. Thus, the second area of influence is geographic distance. Although not a complicated issue onto itself, it does present logistical challenges that are complex to manage. An example of this complexity is a project scenario in which the time zones span more than six hours from one location of the project team spectrum to the other. This scenario would automatically dictate that somewhere within the global team, someone will be required to work very early in the morning or very late at night, possibly even in the

middle of the night, should the team want or need to meet virtually. In addition, if the time zones span less than six hours there may be a very narrow window of opportunity for everyone to conveniently attend a virtual meeting within their respective workdays.

The multi-organizational aspect of a distributed project is the third area of influence offered for discussion. In a large global organization, the occurrence of a distributed project is highly possible but in this context, it is contained within one organization. The term multi-organizational implies that a distributed project is resourced from a host of organizations, possibly spanning the globe, implying that a number of organizations would participate in a singular project effort. This aspect of distributed project may introduce possible issues pertaining to each respective organization such as: internal culture, internal standards, ethics, technological infrastructures, beliefs, and employment stability. Certainly, the reality of a multi-organizational model suggests the need to deal with many entities working toward a common objective. One could assume that there is a higher probability of issues surfacing as a result of the multi-organizational dynamic versus single-organizational model.

The multi-cultural aspects derived from the multi-organizational, multi-border dynamics of a distributed project is suggested as the fourth area of influence. There can be little argument that individuals possess diverse beliefs and value systems that are to some degree, influenced by the societal norms of the country in which they reside. Two individuals may well disseminate information and events differently, thus directly suggesting that a

multi-border level of complexity within a project, could result in influencing the process itself.

#### Review of Specific Software Development Methodologies Impact from DPM

The Waterfall methodology, as described previously, has been in use for approximately four decades and is deemed as a solid methodology. Although, today's software development tools are complete integrated development environments (IDE), that certainly incline toward a more spiral approach to software development; as a methodology Waterfall is highly procedural in structure and certainly lends itself well to project management models and practices. With regards to distributed project management it can be argued that Waterfall does fit this environment with its high documentation requirements and sequential processing approach to software development. The software methodology requirement of phase start and stop milestones, with all stops requiring client sign-off before proceeding to the next start, certainly allows for greater control over the software build at specific points in time by the entire distributed team. According to the survey data, of the respondents using the Waterfall methodology during distributed projects, they were able for the most part, to stay within the development methodology processes.

The fact that the Waterfall Methodology may be able to lend itself more conveniently to a distributed project management approach is not to suggest that issues could not present themselves as a result of being applied during a software development project via a distributed project management approach; and it is put forward that the following issues may

arise due to this particular pairing of processes. Project teams at different sites may apply the pure form of the Waterfall methodology differently, i.e. tailoring or creating a hybrid variation causing the actual methodology to yield uncertain outcomes. This would depend directly on the level of variation from one site to the other, as well as on what phase or key process of the methodology changes were made and to what degree. Uncertainty will invariably have an undesired impact on the methodology and/or project management processes.

General observations around the application of the Waterfall Software Development Methodology and Distributed Project Management are that it is well suited for a distributed application because of its highly structured sequential process approach to software development. Presently, Waterfall has a large base of experienced adopters, hence, as these adopters migrate into a distributed project versus a collocated project, they will carry their accumulated experience of working with the methodology into the distributed project application of the methodology. It is possible to argue that these experienced practitioners are best equipped to implement and manage a distributed software development project given their collective experience base.

The Object-Oriented approach has had a shorter existence in contrast to the Waterfall Methodology, but nonetheless, it has proven to be an accepted and reliable software development approach for the past two decades, based on a spiral approach to software development. As a result, it may be argued that it is more complex in its approach due to the multiple iterations taken to develop a solution to a problem. The increased process

complexity found within Object-Oriented, coupled with the increased dynamics complexity of distributed project management versus collocated project management, may yield a more overall complex environmental process to software development. According to the data acquired from the survey, adopters of Object-Oriented software development report that by and large they can stay within the process during a distributed project environment.

Issues that may present themselves as a result of utilizing the distributed project management on the Object-Oriented Methodology may well be: Variation of methodology, spiral cycle communication complexity, and virtual team members being absent from a cycle. In considering the first issue identified, we notice that this is similar to the issue found in the Waterfall Methodology. Virtual team members may all be adopters of Object-Oriented software development, but it is possible that one, or several, virtual participant(s) may modify the process within their respective environments. The second issue refers to the spiral nature of this approach. Adopters of object-oriented development view the spiral nature of the methodology as a strength, however, within a distributed context the absence of large start and stop milestones may result in negative impacts. These impacts may well result from multiple sites and multiple phase iterations not being 100 percent synchronized thus causing gaps within the development process. While this third impact is local in nature, it is nonetheless critical to Object-Oriented software development. If a virtual team member misses a spiral iteration, hence misses the related communication, the result could be a significant negative impact on their portion of the overall project outcomes.



General observations of the Object-Oriented software development application under a Distributed Project Management process are that, as this is a less mature process, its adopters are most likely earlier along in their professions thus more open to applying distributed approaches to the process. As well, Object-Oriented process itself promotes multiple iterations and parallelisms as it moves throughout its process. Another observation of Object-Oriented software development could conceivably be that it may be more challenging to manage within a distributed context.

Rapid Prototyping, as a methodology within a distributed development project context, may present an initially higher challenge, but generally lends itself to a distributed approach. As a methodology, once the initial prototype is created the virtual team will find themselves observing a process similar to the Waterfall Methodology. The creation of the prototype as the first task or phase of the methodology could be argued to be similar to running a 10-kilometer race on the first day of a new exercise regime. The virtual team is tasked at the commencement of the build to enter into the most complex dynamic, software coding, of any Software Development Methodology, at the same time as they work out the virtual nuances associated with working with new team membership. Clients view this approach, due to the functional prototype creation, as an excellent process to ensure their input (feedback). The research data collected from the survey does not provide adequate feedback pertaining to this methodology to allow for a formal interpretation and therefore are not referenced.

Issues and observations related to the Rapid Prototyping Methodology are centralized around the initial build complexity and challenge of the entire process. Once completed the process should become quite similar to the Waterfall process, hence, presenting similar issues as previously identified pertaining to the Waterfall Methodology.

Extreme Programming Methodology with its multiple build cycles is a radically different application within a collocated project context; logically it would be a more complex application in a distributed project context. The fact that the methodology is based on very short and tight cycles makes it very effective with regards to a build-as-you-go-forward, but from the perspective of distributed project management, it presents the process as a daunting undertaking. The analogy of creating flowerbeds as you move earth into the middle of the flowerbed, hence build as you go, certainly provides a visual to some of the challenges of the methodology. Building as you go, would be applicable in the areas of requirements, specifications, code design, and testing. All areas incurring multiple re-starts logically would present greater demands on all resources and both technical and human infrastructures; these demands could be more defined during a distributed approach due to the distributed associated challenges themselves. It may be difficult to stay true to the multiple-build approach of the XP methodology in a distributed approach given that most developers demonstrate a tendency to migrate towards a sequential approach such as the Waterfall Methodology. The reality of the short historical experience of Extreme Programming makes it challenging to draw sound observations as to how the methodology may be impacted.

Foreseen issues for Extreme Programming Methodology are suggested in the following statements. Defining requirements during the development process in a distributed approach could cause a state of 'lack of direction'. Pair programming would be more difficult as some team members may not be in the same physical location, therefore, losing various aspects of having both members working on one CRT/LCD. As well, the physical approach of Extreme Programming, all in one location, would present a challenge in numerous ways in a distributed approach; so too would the physical client participation requirement in the development area during an XP project. The Extreme Programming Methodology is somewhat radical in its approach; key factors for this are the physical collocation requirements of team members and clients, which are certainly huge challenges within a distributed context. The fact that the methodology requires people be in the same physical location places a parallel force against the fundamental aspects of the distributed approach.

Accepting that Extreme Programming is impacted to the greatest degree as a methodology due to the distributed application of software development, it is not surprising that possible accommodations have been, and continue to be explored. Recent studies by the University of Lincoln are exploring the question of: *what environment supports distributed extreme programming*. [19] This research will add to the body of knowledge being accumulated in regard to the evolution of classical extreme programming.

Kircher et al. suggest that distributed extreme programming is possible and quite attainable. They define distributed extreme programming as follows: "as Extreme Programming with certain relaxations on the requirements of close physical proximity of the team members."

[20] For this evolutionary model of extreme programming to work certain assumptions pertaining to the availability of tools must be accepted. These assumptions are: connectivity between team members already exists, electronic mail is available to all team members, adaptation of configuration management has occurred, application sharing is practiced, video conferencing is made available, and familiarity among the team members exists. [20]

Maurer and Martel defined the use of Minimally Invasive Long-term Organizational Support (MILOS) as a process-support environment developed to aid software developers functioning within a distributed extreme programming environment. For MILOS to support distributed extreme programming it must first provide an environment that coordinates both tasks and developers in a distributed project, second it must provide synchronous communication via tools such as electronic mail, third it must provide active information and information routing among team members, and finally it should support integration of process execution with knowledge management systems. [21]

Distributed Extreme Programming (DXP) is a recent adaptation of an infant methodology, which appears thus far to indicate that the adaptations are supporting distributed software development; although the relatively short historical experience of DXP software development makes it more challenging to concretely determine proven practices. [20]

#### Distributed Project Management Benefits / Impediments in Software Development

In Table 2 benefits and impediments of distribution as an approach are identified and explained.

<b>Benefits</b>
<p><u>Professional Base</u></p> <p>The large global area that one can draw from makes it viable to resource project tasks with highly experienced professionals who are specialists in their respective areas.</p>
<p><u>Issue Visibility</u></p> <p>In a distributed approach communication and documentation is highly critical; it could be argued that issues will be more defined given the micro-review aspect of the process.</p>
<p><u>Creativity</u></p> <p>The virtual aspect of a distributed approach in itself is a dynamically creative process; thinking outside the box is a valued skill, therefore, all participants may find themselves adopting a higher degree of a risk taking in their creativity processes.</p>
<p><u>Competitive Edge</u></p> <p>This approach may allow, due to the large employee base from which to draw for its team, for a competitive business edge versus a team physically located within a single organization.</p>
<p><u>Standards / Protocols</u></p> <p>Members of the virtual team may evolve to a higher level with regards to their respective documentation and coding standards from an exposure to a broader sector of professionals and their respective professional standards.</p>

<b>Impediments</b>
<p><u>Communication</u></p> <p>In a distributed process, communication elevates to higher degrees of importance. Both asynchronous and synchronous communication technologies support aspects that present challenges to communicating in a virtual context. Communication is said to be critical regardless of the process approach in a physical context, thus moving into a virtual context it is a logical assumption to reason that communication would be even more critical.</p>
<p><u>Security</u></p> <p>The aspects of security and privacy are at great risk in a distributed environment given that to communicate team members must do so via external channels versus into a single organization's internal security infrastructure.</p>
<p><u>Standards / Protocols</u></p> <p>From one organization, or from one team member to another, different standards or protocols in coding, communicating, documentation, may exist. These variables will cause issues unless they are recognized and regulated into a single standard and/or protocol.</p>
<p><u>Budgetary</u></p> <p>In distributed environments cost overruns can become a concern as it can be difficult to identify the exact overrun source. It is critical that the project manager have a varied repertoire of tools and skill sets to manage the financial complexity and diversity of a distributed project.</p>

Table 2 ~ Benefits / Drawbacks of Distributed Projects on Software Development Methodologies

### Distributed Project Management Impediment Elimination Strategies

*Communication* impediment can be partially resolved with the adoption of solid and reliable asynchronous tools such as e-mail or web discussion boards. The web discussion boards can be expanded to include specific web discussion rooms. The creation of a Q & A virtual area may assist virtual team members in being informed of challenges being experienced by others, as well as, possible solutions to these challenges. The web-enabled environment that accommodates for posting information is one that the project's client could take full advantage of as well. Web tools that would allow for joint, as well as an individualized, calendaring of tasks may present logistical management challenges on the communication of what tasks need to be completed by whom and by what deadline. The documentation of all communiqués is more essential in a virtual environment compared to a collocated environment structure because there is no convenient physical access to team members. Traditional synchronous communication tools, such as teleconferencing and video conferencing would provide the opportunity to have face-to-face discussion without the need of physical displacements. Recent technologies such as voice-over-IP and video-over-IP technologies permit the application of traditional communication approaches at a fraction of the cost.

*Security* impediments may be managed with the appropriate centralization of critical information, the adoption of appropriate firewall technology, and the documentation of virtual team member's rights and profile assignments. The implementation of a global, distributed Intranet would allow for more access control as well as the ability to impose standards and protocols. Basic data encryption will facilitate in providing appropriate

security levels. Finally, the creation of security agents whose task is to constantly search out any infractions of the standards and protocols would enhance security.

*Standards / Protocols* impediments may be minimized if the virtual team undertakes early in the process the establishment of a single standard and/or protocols for their project and the project's respective components. The adoption of standards and protocols alone is not sufficient; these standards and protocols must be communicated to the virtual team otherwise, it is of no worth to establish them. The communication of the standards can be accomplished via e-mail attachments, physical mail outs, or the establishment of a quality website that contains all documentation samples in PDF format. By establishing such a website these samples would be available for review upon demand by any virtual team member.

*Budgetary* impediments, from a management perspective should not be an issue assuming everyone follows the project plan and uses the identified resources. Human nature being what it is and software development having the historical background that it has, certainly one is required to face the harsh reality that projects usually go over budget. Two possible approaches in combating budget-overrun realities are, introduction of budget agents, who independently follow and approve what is transpiring as it occurs in the project and the creation of a project budget website. This website would allow virtual team members to view all line items and, when required, concentrate within any one of the main budgetary line items for a more in-depth budgetary explanation.



## Best Practices of Distributed Project Management on Software Development

As the domain of Distributed Project Management evolves it is highly predictable that a progression of best practices will evolve. In time, the Project Management Institute (PMI) will most certainly endeavor to establish such a list. It can also be assumed that as the IEEE Computer Society put forward the 1058 Software Project Plan Standard so too will a distributed standard be developed by the IEEE Computer Society to assist the increasing number of practitioners of this domain. Regardless of standards existing or not, Distributed Project Management does have the potential of positively or negatively impacting every aspect of software development and the respective methodology processes. At this point-in-time, one can but put forward only suggestions on what a best practices list might entail.

### ***Best Practice 1 ~ Communication***

Maintain open communication channels via asynchronous or synchronous technologies, to name the most apparent, computer mediated communications, instant messaging, web conferencing, voice-over-IP, video-over-IP, electronic mail, teleconferencing, and video-conferencing. It is important not to presume that no physical gatherings are required; strategically positioned physical team meetings or individual site visits may encourage the effectiveness of distributed communication channels used during the project.

### ***Best Practice 2 ~ Infrastructure***

Secure the appropriate technological infrastructure, both hardware and software, to support the distributed aspects of the project in question. Equally important, is the need to

secure the appropriate technology to facilitate the application of the particular Software Development Methodology being applied. Workflow management systems, Web project management systems, etc. will enhance the success factor of the software development project.

### ***Best Practice 3 ~ Development Methodology***

Select a Software Development Methodology that works best for the functional requirements of the software being developed. Identifying a methodology which allows the entire team to focus on a united process and resource your team with members who are experienced in the application of the selected Software Development Methodology. The environmental development tools available today promote a more spiral methodological approach versus a more structured methodology, such as the Waterfall Methodology.

### ***Best Practice 4 ~ Standards & Protocols***

Select documentation standards and protocols that will facilitate use of the selected methodology. The application of such standards and/or protocols throughout the development methodology will assist in minimizing needless confusion, which accounts for potential loss of time, productivity, and finances.

### ***Best Practice 5 ~ Collaboration***

Maintain regular contact with team members, whether it is the project manager contacting a virtual team member, or a virtual team member contacting other members or the project manager. It is critical to maintain regular contact, otherwise, a sense of lost

connectivity may permeate the virtual team. A virtual member can easily feel they are not being heard, therefore, it is important to regularly initiate such practices as: virtual member show-and-tell, member reviews of work accomplished, and occasional on-site visits. Keeping in mind that, in most cases, a member overcome with the feeling of total isolation will not function as effectively as a member who feels they are part of a group – even a virtual group.

#### ***Best Practice 6 ~ Code Controlling***

Select an appropriate source code controlling software tool that is specifically designed for virtual team application.

#### ***Best Practice 7 ~ Quality Assurance***

Establish quality assurance protocols regardless of the methodology being adopted for the software build, for in the end they will serve the process well by ensuring a software product that is as fault free as possible.

#### **Distributed Project Management Evaluation Criteria**

A step in the direction of quality assurance is to establish a mechanism by which the process can be evaluated objectively and in a timely fashion. The proposed assessment matrix in Appendix B is a tool that may assist in determining the level of impact that distributed processes could have on the software methodology used by the virtual team. As seen in the matrix, with a possible total score of 40, a total score greater than 35 may indicate no impact on the software methodology while a score of less than or equal to 35, but greater

than 25 may indicate a low negative impact on the software methodology. A score of less than, or equal to 25, but greater than 10, may indicate a moderate to significant negative impact on the software methodology. Finally a score less than or equal to 10 may experience a high negative impact on the software methodology.

Distributed Project Management, as defined, presents many opportunities as well as challenges. This section has attempted to demonstrate how Software Development Methodologies may be influenced by Distributed Project Management.

## CHAPTER VI

### CONCLUSIONS AND RECOMMENDATIONS

#### Conclusion

Software Development Methodologies are complex processes that are simultaneously resilient and fragile; distributed projects and their management have the potential of impacting Software Methodologies negatively hence potentially compromising the positive delivery of a sound software product. Methodologies themselves provide diverse approaches to resolving the requirements of a particular process. When looking at their respective theoretical models, although diverse, a common thread is woven among them. Whether it is a historically long-standing methodology such as the Waterfall approach or a new innovative methodology such as Extreme Programming, all have an identical goal in the end, to provide a solid, i.e. fault free, software product that meets the client's stated or determined requirements.

Distributed Project Management is a relatively new application of a proven approach, i.e. project management, but the fundamental aspects of project management are still being observed and evaluated throughout the project's life span. The key challenge of a distributed project, while no different than a geographically centered project, is one of communication. In a distributed project context, this challenge is elevated above all others. It is suggested that regardless of the strength of the adopted Software Development Methodology, if the project manager is not promoting strong communication practices, the entire process will be flawed.

In this report, results from the survey, although not meant to be statistically defensible but to be used only as a scan of the regional state of affairs, did indicate communication as the single most critical aspect that could influence software methodology. What the survey did not directly report, is how the four software methodologies, listed could be impacted by the distributed nature of the project.

This author suggests that Software Development Methodologies are process challenged by being applied over a virtual venue with the level of challenge being directly correlated to the Distributed Project Management application strengths and/or weaknesses.

Although no one methodology was singled out as one that would usually fail if applied over a distributed venue, the fundamentals of Distributed Project Management support that the methodology that would present the greatest challenge if adopted over a distributed venue is the Extreme Programming Methodology. This is due to the physical characteristics of the Extreme Programming Methodology that could be negatively impacted by being applied via a distributed approach, such as pair programming and client constant interaction with the entire development team. This is not to suggest that creative adaptations and/or accommodations could not be made to allow for the convenient use of this methodology over a distributed venue. This trend may well emerge with the evolution of Distributed Extreme Programming models.

### Further Research Suggestions

Future research exploration on the following aspects of the application of Software Development Methodologies and Distributed Project Management would provide enlightening data that could provide greater understanding of the application of Distributed Project Management and the impact introduced by such an application.

- ❑ Distributed Extreme Programming State of the Art.
- ❑ Distributed Project Management Life Cycle Versus Collocated Project Management Life Cycle Variation in Process.
- ❑ Software Development Methodology's Influence over Distributed Project Management Models.
- ❑ Application of Agent Technology on Distributed Project Management.

## REFERENCES

1. The Standish Group. 1995. The Standish Group Report ~ CHAOS. Web Reference: [http://www.projectsmart.co.uk/docs/chaos\\_report.pdf](http://www.projectsmart.co.uk/docs/chaos_report.pdf). Accessed: January 4, 2005.
2. Bonnal, P. et al. (2002, March). The Life Cycle of Technical Projects. Project Management Journal, pp 12-19. Web Reference: <http://search.epnet.com/login.aspx?direct=true&db=tfh&jid=1NR> . Accessed: January 12, 2005.
3. Schach, R. S. (2002). Object-Oriented and Classical Software Engineering, 5<sup>th</sup> Edition. New York, NY: McGraw-Hill Higher Education.
4. McMahon, P. (2001, November). Distributed Development: Insights, Challenges, and Solutions. The Journal of Defense Software Engineering. Web Reference: <http://www.stsc.hill.af.mil/crosstalk/2001/06/leishman.html> . Accessed: January 7, 2005.
5. Fjermestad, J. et al. (2004). Collaborative Project Management. A Special Issue of the Journal: International Journal of e-Collaboration. Web Reference: <http://web.njit.edu/~jerry/IJeC-Collaborative-Project-Management.htm> . Accessed: November 5, 2004.
6. Chaar, J. et al. (1996, May). Virtual Project Management for Software. IBM T.J. Watson Research Center. Web Reference: <http://lsdis.cs.uga.edu/activities/NSF-workflow/santanu.html> . Accessed: December 16, 2004.
7. Damian, D., & Zowghi, D. (2002). An insight into the interplay between culture, conflict and distance in globally distributed requirements negotiations. IEEE Computer Society. Web Reference: <http://csdl.computer.org/comp/proceedings/hicss/2003/1874/01/187410019c.pdf> . Accessed: January 30, 2005.
8. Beise, C. (2004, April). IT Project Mangement and Virtual Teams. ACM, pp 129-133. Web Reference: <https://portal.acm.org/> . Accessed: January 20, 2005.
9. The Program Management Group plc. (2004, December). Web Reference: [www.pm-group.com](http://www.pm-group.com). Accessed: January 5, 2005.
10. Badir, Y. et al. (2003, September). Management of Global Large-Scale Projects Through A Federation of Multiple Web-Based Workflow Management Systems. Project Management Journal, pp. 40 – 47. Web Reference: <http://search.epnet.com/login.aspx?direct=true&db=tfh&jid=1NR> . Accessed: January 12, 2005.
11. Nienaber, R. & Cloete, E. (2003). A Software Agent Framework for the Support of Software Project Mangement. Department of Computer Science and Information Systems, University of South Africa, pp. 16 – 23. Web Reference: <http://portal.acm.org/> . Accessed: December 27, 2004.
12. Schwalbe, K. (2002). Information Technology Project Management. Boston, MA: Thompson Learning Inc., pp. 4 – 27.



13. Jaafari, A. (2003, December). Project Management in the Age of Complexity and Change. Project Management Journal, pp 47 - 57. Web Reference: <http://search.epnet.com/login.aspx?direct=true&db=tfh&jid=1NR> . Accessed: January 12, 2005.
14. Collaborative Strategies. (2002). Distributed Project Management Update 2002 (Volume 1) Executive Summary. Web Reference: [http://collaborate.com/announcementsannounce\\_6.html](http://collaborate.com/announcementsannounce_6.html) . Accessed: December 29, 2004.
15. Hill, L. & Morgan, JP. (1995, October). Object Oriented Project Management. OOPSLA '95 Proceedings, pp. 89 – 93. Web Reference: <http://portal.acm.org/> . Accessed: December 30, 2004.
16. Katzy, B. et al. (1999, September). Knowledge Management in Virtual Projects: A Research Agenda. Web Reference: <http://portal.cetim.org/file/1/92/KatzyEvaristoZigurs.pdf> . Accessed: January 29, 2005.
17. Katayama, T. (2001). What is Object-Oriented Methodology?. Tokyo Institute of Technology. Web Reference: <http://www.soi.wide.ad.jp/class/20010030/slides/05/img/5.png> . Accessed: April 26, 2005.
18. Kanbay, J. (2002, January). Extreme Programming (XP). Serverworld Magazine. Web Reference: [http://www.serverworldmagazine.com/webpapers/2002/01\\_Kanbay.shtml](http://www.serverworldmagazine.com/webpapers/2002/01_Kanbay.shtml). Accessed on: April 25, 2005.
19. Adams, P. A Collaborative Environment to Support Distributed eXtreme Programming. University of Lincoln. Web Reference: <http://eprints.lincoln.ac.uk/39/01/abstract.pdf>. Accessed on: April 27, 2005.
20. Kircher, M. et al. Distributed eXtreme Programming. Corporate Technology. Munich, Germany. Web Reference: <http://www.cs.wustl.edu/~corsaro/papers/XP2001.pdf>. Accessed on: April 27, 2005.
21. Maurer, F. & Martel, S. Process Support for Distributed Extreme Programming Teams. University of Calgary, Computer Science Department. Calgary, Alberta, Canada. Web Reference: <http://sern.ucalgary.ca/~milos/papers/2002/maurerMartel2002a.pdf> Accessed on: April 27, 2005 .

## **APPENDIX A ~ Survey Tool**

**Athabasca University ~ Master of Science in Information Systems**

**COMP 696 Thesis Essay**

**Research Survey**

Researcher: Gerald L. Caissy BCS

**Topic:** *Impact of Distance / Distributed Project Management on Software Development Methodologies.*

**Section A:** Please check the one description that best reflects your situation:

<b>Employment sector:</b>  <input type="checkbox"/> Public <input type="checkbox"/> Corporate <input type="checkbox"/> Small Enterprise <input type="checkbox"/> Education <input type="checkbox"/> Other	<b>Current primary job title/duties:</b>  <input type="checkbox"/> Analyst <input type="checkbox"/> Designer <input type="checkbox"/> Data Base Administrator <input type="checkbox"/> Programmer/Analyst <input type="checkbox"/> Project Manager <input type="checkbox"/> Other
<b>IS or IT or CS professional experience:</b>  <input type="checkbox"/> Under 2 years <input type="checkbox"/> 2..4 years <input type="checkbox"/> 4..8 years <input type="checkbox"/> 8..15 years <input type="checkbox"/> Over 15 years	<b>Highest credential level obtained in any field:</b>  <input type="checkbox"/> Diploma <input type="checkbox"/> Degree <input type="checkbox"/> Graduate (Master or Doctorate) <input type="checkbox"/> Other
<b>Work (or have worked) on team based projects:</b>  <input type="checkbox"/> Yes <input type="checkbox"/> No	<b>Work teams are geographically:</b>  <input type="checkbox"/> Centralized <input type="checkbox"/> Distributed

**Section B:** Please select the answer that best reflects your situation:

Are you currently applying project management structures in your software development activities?

☐ Yes    ☐ No

What software development methodology would you classify as your primary methodology?

☐ Waterfall (Structured)                      ☐ Rapid Prototyping  
☐ eXtreme Programming                      ☐ Object-Oriented                      ☐ Other \_\_\_\_\_

Are you currently using collaborative tools to enhance your software development project management.

☐ Yes    ☐ No

If YES please indicate the tools you have used (*you may select many*):

<input type="checkbox"/> PM Software (ex. MS Projects)	<input type="checkbox"/> Internet Conference	<input type="checkbox"/> Teleconference
<input type="checkbox"/> Video Conference	<input type="checkbox"/> E-mail	<input type="checkbox"/> Bulletin Board
<input type="checkbox"/> MSN type environments	<input type="checkbox"/> Web PM tools	<input type="checkbox"/> _____
<input type="checkbox"/> _____	<input type="checkbox"/> _____	<input type="checkbox"/> _____

**OVER to Section C ...**

**Section C:** Please fill in the following as it best reflects your experiences:

Were you able to stay within the SD methodology given that you were using distributed project management?

- ☐ Fully (to the purest intent of the SD methodology)
- ☐ Mostly (used the SD methodology but experienced a few challenges)
- ☐ Somewhat (SD methodology itself caused challenges due to the distributed aspect of the PM)
- ☐ Little (most attempts to stay within the SD methodology did not work)

Based on your experience and development methodology identified, what impact(s) occurs on the development methodology as a result of working via distributed project management?

1	
2	
3	
4	

Based on your experience and development methodology identified, what possibility(ies) occurs towards the development methodology as a result of working via distributed project management?

1	
2	
3	
4	

How would you rate the existence of the following aspects, as they pertain to distributed project management: *(Circle the appropriate response that best reflects your experience)*

	Never	Always				
1. Communication plan / strategy	1	2	3	4	5	n/a
2. Privacy policies	1	2	3	4	5	n/a
3. Security Protocols	1	2	3	4	5	n/a
4. Virtual trust environment	1	2	3	4	5	n/a
5. Leadership / supportiveness	1	2	3	4	5	n/a
6. Occasional team collocation	1	2	3	4	5	n/a
7. Appropriate technical infrastructure	1	2	3	4	5	n/a
8. Conflict resolution processes	1	2	3	4	5	n/a
9. Cultural diversity awareness	1	2	3	4	5	n/a
10. Clear software development methodology adoption	1	2	3	4	5	n/a
11. Flexibility in changing development methodology	1	2	3	4	5	n/a
12. Flexibility in tailoring adopted development methodology	1	2	3	4	5	n/a

**Thank you for participating in my survey! Please return by January 28<sup>th</sup>, 2005 to:**

Gerald L. Caissy  
Holland College Charlottetown Centre  
140 Weymouth Street  
Charlottetown, PE, C1A 4Z1

Telephone: 902.566.9663  
Fax: 902.566.9335  
E-mail: [gcaissy@hollandc.pe.ca](mailto:gcaissy@hollandc.pe.ca)

## **Appendix B ~ Assessment Matrix**

## Distributed Project Management Impact on Software Development Methodology Assessment Matrix

**Directions:** Select the appropriate impact level, record the respective weight score in the score column.

Criteria	High Impact	Moderate Impact	Low Impact	No Impact	Score
Weight Score	1	2	3	4	
<b>Software Development Methodology</b>	Methodology has not been adopted.	Methodology has been identified, but each team member is applying their own interpretation (variation) of the methodology.	Methodology has been identified with standards on its application. Team members are adhering to it the best of their ability.	Methodology has been identified with standards on its application. Team members are complying 100% to the standards.	
<b>Communication</b>	No communication plan exist.	Communication plan exist but no one is adopting it as the standard. Team members are functioning more at an asynchronous mode.	<b>Communication plan exist and is adopted as the standard. A few team member are not practicing it 100%, with most functioning at a synchronous mode.</b>	Communication plan exist and is adopted by all team members 100%. Team is functioning in a blend of synchronous / asynchronous communication modes.	
<b>Security</b>	No security plan exist.	Security plan exist but only some have adopted it.	Security plan exist and most team members have adopted it.	Security plan exist and all team member have adopted it.	
<b>Culture</b>	Team membership composed of different socio, political, and economic beliefs.	Team membership composed of similar socio, political, and economic beliefs.	Team membership composed of common socio, political, and economic beliefs.	Team membership composed of identical socio, political, and economic beliefs.	
<b>Team Dynamics</b>	Members work in 100% isolation with no synchronous communication with other team members.	Members work together in asynchronous mode only.	Members work in over a multitude of asynchronous and synchronous modes.	Members work in a fully synchronized mode. Interact very freely with one another.	
<b>Infrastructure</b>	No technological infrastructure is defined, hence everyone is using their own interpretation.	Technology infrastructure is documented but only partially implemented.	Technology infrastructure is documented and implemented but not adopted by all members.	Technology infrastructure is documented, implemented and adopted by all members.	
<b>Language</b>	Everything requires translation.	Team members can read and write in one common language but verbal skills are mixed.	Team members can read and write in one common language and converse at a functional level.	Team members are fully fluent in one common language existing within the team.	
<b>Project Management CMM Level</b>	Functioning at level 1 of the Capability Maturity Model (CMM).	Functioning at level 2 or 3 of the Capability Maturity Model (CMM).	Functioning at level 3 or 4 of the Capability Maturity Model (CMM).	Functioning at level 4 or 5 of the Capability Maturity Model (CMM).	
<b>Client</b>	Client is not available.	Client is difficult to contact.	Client is available within a reasonable timeframe.	Client is part of the team.	
<b>Documentation</b>	No standards are defined, hence documentation is not being captured.	Standards exist but almost no one has adopted them.	Standards exist and they are being adopted by most members.	Standards exist and they are adopted by all.	
				<b>Total Score</b>	

**Scoring Results:** \_\_\_\_ / 40

Score <= 10 ..... High negative impact of software methodology  
 Score > 10 and <= 25 ..... Moderate negative impact of software methodology  
 Score > 25 and <= 35 ..... Low negative impact of software methodology  
 Score > 35 ..... No negative impact of software methodology