MANAGING GROUP DEVELOPMENT:
A PATTERN LANGUAGE FOR GROWING AND
MAINTAINING HIGHLY EFFECTIVE GROUPS

by

Judy N. Green

A thesis submitted in partial fulfillment of
the requirements for the degree of

Master of Computer Science

Carleton University

15 January, 2003

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 0-612-83374-7

Our file Notre référence

ISBN: 0-612-83374-7

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

Canada

**The undersigned recommend to the Faculty of Graduate Studies and Research
acceptance of the thesis**

**MANAGING GROUP DEVELOPMENT: A PATTERN LANGUAGE FOR
GROWING AND MAINTAINING HIGHLY EFFECTIVE GROUPS**

Submitted by JUDY N. GREEN, B.C.S.

In partial fulfillment of the requirements for

The degree of Master of Computer Science

Thesis Supervisor

Director, School of Computer Science

**January 17, 2003
Carleton University**

ABSTRACT

MANAGING GROUP DEVELOPMENT: A PATTERN LANGUAGE FOR GROWING AND MAINTAINING HIGHLY EFFECTIVE GROUPS

by Judy N. Green

Student # 274293

Chairperson of the Supervisory Committee: Dr. Dwight Deugo
School of Computer Science

A multidisciplinary thesis presenting a Pattern Language, of solutions from Social Science's group theory, designed to facilitate the growth and maintenance of highly effective groups, also known as teams.

Groups of Software developers must overcome many issues on their path to becoming highly effective groups. This thesis proposes that the vast number of solutions to these issues, in the domain of group theory, is a sub category of Organizational Patterns that the author introduces as People Patterns. These patterns address the specific issues that must be resolved for optimal performance of the group.

This thesis proposes the use of a state machine in the form of the Sequential Stage Theory of Group Development as described by Tuckman in 1977 to partition the People Patterns into groups identified by their forces. This partitioning is used to organize and present the solutions in the form of a Pattern Language of People Patterns.

ACKNOWLEDGMENTS

The author would like to thank all the people whose support was critical to the completion of this thesis:

- ◇ Dr. Larry Hughes, from Saint Mary's University, who convinced me that I might just have a knack for programming, and who has supported my academic endeavours throughout the years
- ◇ Dr. Ivan Tomek, from Acadia University, who set me firmly on the Object Oriented path
- ◇ Dr. Glynn Bissix of Acadia University, for introducing me to Group Theory and his Social Science students for their valuable feedback on the theories and solutions presented in this thesis
- ◇ Participants in the Java One 2002 Bird's of a Feather, "Growing and Maintaining Highly Effective Groups," for their validation of the ideas presented, their many helpful suggestions, and their continued support of this research
- ◇ Dr. Dwight Deugo for his patience and support throughout the past four years and two thesis topics
- ◇ Most of all my husband Allan and our son Jason, who picked up the slack at home and washed more dishes, laundry and floors than they truly cared to in order for this thesis to be completed

Thank you all

-Judy

TABLE OF CONTENTS

TABLE OF CONTENTS.....	V
LIST OF FIGURES	VIII
LIST OF TABLES.....	IX
I - INTRODUCTION	10
1 - CONTEXT.....	10
2 - PROBLEM.....	12
3 - GOALS.....	13
4 - EXPECTED CONTRIBUTION	13
5 - OUTLINE FOR THE REMAINDER OF THESIS	14
5.1 - Chapter 2 – Background.....	14
5.2 - Chapter 3 – Approach	14
5.3 - Chapter 4 – Results.....	16
5.4 - Chapter 5 – Conclusion.....	16
II - BACKGROUND.....	17
1 - INTRODUCTION.....	17
2 - GROUP DYNAMICS	18
2.1 - Introduction to Group Theory	18
2.2 - Theories of Group Development	25
2.3 - Summary of Introduction to Group Theory.....	28
3 - PATTERNS.....	30
3.1 - Introduction to Patterns	30
3.2 - Formats of Patterns.....	34
3.2.1 - Alexandrian Format	35
3.2.2 - Portland Format.....	36
3.2.3 - GOF Format	37
3.2.4 - Compact Format.....	38
3.2.5 - Canonical Format (a.k.a. Coplien format or AGCS format).....	39
3.2.6 - Cockburn PM Format	41
3.2.7 - Beck Format	42
3.2.8 - Fowler Format.....	42
3.2.9 - Formats of Patterns Summary.....	43
3.3 - Process Patterns vs. Organizational Patterns	45
3.4 - Problems with the Organization of Existing Process and Organizational Patterns	47
3.5 - Summary of Introduction to Patterns	48
4 - CURRENT RESEARCH ON ISSUES AFFECTING SOFTWARE DEVELOPMENT TEAMS.....	49
4.1 - Introduction	49
4.2 - Joining Together: Group Theory and Group Skills Sixth Edition	49
4.3 - The Team Handbook, Second Edition.....	50
4.4 - The Pragmatic Programmer: from Journeyman to Master.....	51
4.5 - The Mythical Man Month	52
4.6 - Managing Programming People	53
4.7 - The Manager Pool: Patterns for Radical Leadership.....	54

4.8 - The Peopleware Papers	56
4.9 - Software for Your Head: Core Protocols for Creating and Maintaining Shared Vision	57
4.10 - Summary of Current Research on Issues Affecting Software Development Teams	59
III - APPROACH.....	61
1 - INTRODUCTION.....	61
2 - IDENTIFYING PEOPLE PATTERNS	61
3 - ORGANIZATIONAL FRAMEWORK	64
3.1 - Divide and Conquer; the Sequential Stage Theory as a State Machine.....	64
3.2 - Context and Gating Issues of each State and Transition	66
4 - PATTERN LANGUAGE: GROWING AND MAINTAINING HIGHLY EFFECTIVE GROUPS	69
4.1 - Introduction	69
4.2 - Terminology	71
4.3 - The Framework and How to Use it.....	71
4.4 - The Patterns.....	74
Stage: Forming	74
Forming Meta-context:	74
Forming – Transition A: Growing from Forming to Storming	76
A-1 -- Getting to Know You	76
A-2 -- Go for the Goal.....	80
A-3 -- Our Team.....	84
Stage: Storming	87
Storming Meta-context:	87
Storming – Transition A: Growing from Storming to Norming	88
A-1 -- Value Your Neighbour.....	89
A-2 -- Resolving Conflict.....	91
A-3 -- Harnessing Competitive Energy	97
Stage: Norming.....	100
Norming Meta-context:.....	100
Norming – Transition A: Growing from Norming to Performing	102
A-1 -- Roles, Rights & Responsibilities	102
A-2 -- Only Productive Meetings.....	107
A-3 -- Creative Communication	112
Stage: Performing.....	117
Performing Meta-context:.....	117
Performing – Transition A: Maintaining Optimal Efficiency in Performing.....	118
Transitional context:	118
A-1 -- Celebrate Success	119
A-2 -- Oil & Lube.....	122
Performing – Transition B: Adjusting to Adjourning/Reforming.....	124
Transitional context:	125
B-1 -- Separation Issues	125
B-2 -- Post Mortem	128
5 - DISCUSSION AND DECISIONS	133
6 - SUMMARY OF APPROACH	136
IV - RESULTS	138
V - CONCLUSION	144
1 - SUMMARY	144

2 - FUTURE WORK.....	145
APPENDIX 1 - GLOSSARY.....	148
BIBLIOGRAPHY	150
WEB REFERENCES	152

LIST OF FIGURES

Figure 2.1 - Performance of Groups over Time	20
Figure 2.2 - Theory to Research to Practice	23
Figure 3.3 - Sequential Stage Theory as a State Transition Diagram	66
Figure 3.4 - Partitioning the Problem Domain	68
Figure 4.5 - State Transition Diagram	73

LIST OF TABLES

Table 1 - Logical Sequence of Stages over Time	26
Table 2 - Logical Stages of Group Membership Over Time	28
Table 3 - Guide for Communication	95
Table 4 - Rules for Fair Fighting	96
Table 5 - Communication Technology by Priority	116
Table 6 - Glossary of Group Development Terms	149

I - INTRODUCTION

1 - Context

In the realm of social science, there exists an area of study called Group Theory. This Group Theory is not to be confused with the area of Mathematics with the same name. Social Science Group Theory is the study of the interactions of people within small groups. This interaction is called Group Dynamics. Scientists from many disciplines have actively researched the dynamics within small groups since early in the twentieth century.

This area of knowledge is vast and unknown to most software developers and many managers. Johnson & Johnson (1997) state that, "it takes both knowledge and skills to build and maintain effective groups." Some software developers are lucky and have good instincts when it comes to interacting within a group. Most stumble along without the knowledge and skills that would help them become more effective as group members. Even worse, without this knowledge and the skills to become an effective group member, they may unwittingly hinder the effectiveness of all the members within their group. Even a software development group member that has some knowledge of group dynamics will encounter situations that are beyond their current level of knowledge and skill, at some point in their careers.

Johnson & Johnson and other social scientists have determined that an understanding of Group Dynamics is critical to the maintenance of psychological health. Knowledge of group dynamics is central to maintaining a viable family and to providing an effective education. An understanding of the interaction

within groups leads to more effective businesses and industries. This applies no less to the industry of software development.

The stereotype of a software developer, working through the night staring at his or her monitor with no interaction with other human beings, has never been farther from the truth. As the areas of software development and software engineering have matured, there has been an increased focus on improving the effectiveness of our software development teams through improved teamwork. Well known processes for software development, such as Extreme Programming and Agile Programming both contain facets related to how group members interact with one another.

In *The Mythical Man Month* (Brooks 1995), the author clearly make the case that more software development projects have failed due to problems within the dynamics of the group (specifically poor communication) than from any lack in technology. It is clear that an understanding of group dynamics is critical to the success of our software development groups. The first line Larry Constantine writes in his book *The Peopleware Papers* (Constantine 2001) states, “Good software does not come from CASE tools, visual programming, rapid prototyping, or object technology. Good software comes from people. So does bad software.” It is clear that many people have acknowledged the importance of people in software development, but it is also clear that the software development community needs to do something positive about it.

The large body of knowledge generated by the group theorists, often in the form of best-practices and tips, has been derived from the scientific process. These solutions to the problems humans encounter when they attempt to work in groups have been rigorously tested. They continue to be used within both clinical

settings and the field. These are solutions to problems within the context of groups. These solutions are in fact, what we in the world of software call patterns.

A Software Pattern is like a recipe used to recreate a solution to a problem within a context. Software Patterns and collections of related patterns, called Pattern Languages have been gaining favour since the mid 1980s, as a way to capture and disseminate the knowledge acquired by the most experienced and skilled software developers and architects. Two categories of Software Patterns called Process Patterns and Organizational Patterns have attempted to deal with some of the issues that affect software development groups and the software development process.

As we look closer at the group theory solutions documented by social scientists we quickly realize that they are repeatable solutions to problems within the context of groups. These solutions are patterns derived from the knowledge of those who study the field of group dynamics. In fact, we can think of them as candidates for a sub-category of Organizational Patterns that this thesis presents as People Patterns.

2 - Problem

The problem is that there is currently no clear format or tool that allows people to access a solution, to a problem that their group may be encountering, in a contextually relevant and timely manner.

There is a vast amount of complex information surrounding the solutions to the problems that groups experience as they mature. Solutions are typically scattered throughout hundreds of books and articles, intermixed with related information, which may be— but more likely is not— relevant to the specific situation encountered by the group member. Most people do not have the time to study group theory in depth and become discouraged when trying to find a solution to

a seemingly simple problem. They need a way to identify the problem and access potential solutions.

3 - Goals

The goal of this thesis is twofold. The first goal is to identify and present the solutions to the problems encountered by groups in their quest to mature into effective groups as a candidate pattern language of People Patterns. The second goal is to provide an organizational framework for this new pattern language that will allow us to partition the solutions to the problems encountered by software groups and present the solutions in a manner that will allow people to more readily identify their problem, and find a solution.

This thesis meets these goals by identifying a representative selection of People Patterns, within the group theory literature, as well as within the existing Process and Organizational Pattern Languages. The second goal is met through the organization of the identified People Patterns into smaller, well identified groupings based on the Sequential Stage Theory (Tuckman 1977) of group development. Each grouping is well identified with respect to the context within which the group members are functioning. This description of context helps to allow a user to readily choose which section of patterns is most relevant to their immediate needs. This in turn helps to speed up the process of finding a solution.

4 - Expected Contribution

This thesis will present thirteen candidate patterns in the form of a pattern language called Growing and Maintaining Highly Effective Groups. The patterns are presented in the organizational framework of a state machine where Tuckman's stages each become a state in the state machine, and the transitions are as described below.

e.g.

Transitions

Forming	→	Storming	- Growing from Forming to Storming
Storming	→	Norming	- Growing from Storming to Norming
Norming	→	Performing	- Growing from Norming to Performing
Performing	→	Performing	- Staying in the optimal Performing Stage
Performing	→	Adjourning	- Preparing for the group to disband

There will be a total of thirteen candidate patterns for the five transitions in Tuckman's Sequential Stage Theory (Tuckman 1977), which is used to partition the domain of solutions; three patterns for each of the first three transitions and two for each of the transitions from the Performing stage. This organizational framework will also provide a single inheritance mechanism that will allow the patterns at a specific stage to inherit context and forces from their encompassing stage.

5 - Outline for the Remainder of Thesis

5.1 - Chapter 2 – Background

The background Chapter introduces the reader to the concepts and research behind the group theory used in the approach. The reader is also introduced to the world of patterns, software patterns, pattern languages, Process Patterns and Organizational Patterns. A survey of related group theory and Organizational Patterns research is also presented.

5.2 - Chapter 3 – Approach

The approach Chapter presents and explains the techniques the author used to formulate the pattern language and organizational framework that constitutes the contributions of this thesis. The Chapter will explain how a multidisciplinary approach using divide & conquer, state machines, inheritance and patterns allows

us to use computer science techniques to partition and present the solutions to the problem groups face during their development, in an organized fashion. This Chapter will also describe the feedback provided by a class of Social Science students at Acadia University, and how it has enhanced the resulting pattern language.

This Chapter also presents this thesis' combined contribution of a pattern language of thirteen representative People Patterns mined from group theory research and existing pattern languages, and the organizational framework within which they are presented.

The People Patterns are presented within the organizational framework of a pattern language partitioned within a state machine (based on Tuckman's Sequential Stage Theory (Tuckman 1977)). Each stage in the Sequential Stage Theory (Tuckman 1977) maps to a state in the state machine. Each stage or state describes the contextual forces that influence all patterns within that state. There are contextual forces that relate to the stage of development as well as forces relating to the gating issues associated with a specific transition to another state. Each People Pattern in each state inherits the context of its state in addition to any contextual forces that are specific to the transition the People Pattern is associated with.

The resulting Pattern Language, within its organizational framework, facilitates access to specific solutions, to the problems what groups face, in a timely manner. By fully describing the context at each stage of group development, the framework educates the user, as well as partitions the solutions in a way that allows for easier identification of where the specific solution can be found. The solutions, in the form of People Patterns, are described in a manner that transfers

the knowledge from experts in group theory to those who can best benefit from that knowledge; group members.

5.3 - Chapter 4 – Results

The results Chapter discusses the validation of the patterns and organizational framework contributed by this thesis. It explains that the patterns and the theory, upon which the framework is based, have been independently verified before being used together in this multi-disciplinary approach.

5.4 - Chapter 5 – Conclusion

The conclusion describes how the contributions of this thesis have presented a solution to the problem described. A section on future work has also been included that describes how an expert system could be implemented to present the resulting pattern language in an easy to access format. This Chapter also describes how additional People Patterns can be mined and added to the pattern language.

II - BACKGROUND

1 - Introduction

This thesis uses core concepts from Computer Science such as divide-and-conquer, state machines and inheritance in order to create an organizational framework within which to present the solutions to the problems encountered by software development groups. These concepts are considered central to the study of computer science. As such, they will not be covered in depth within this thesis.

The contributions of this thesis include a pattern language of solutions to problems that groups face as they work toward becoming highly effective groups. In order to understand the solution and the approach presented in Chapter 3, the reader must have a reasonable understanding of patterns and group theory.

The introduction of this thesis, describes why knowledge of group theory is important, and why we should be striving to grow our groups into highly effective groups. Group theory is a vast area of study and provides the solutions, to the problems that groups face over time, which are presented as People Patterns in this thesis. This Chapter will give the reader an overview of this social science, introduce them to the solutions provided by social scientists and describe the specifics of the theories used to partition the domain of solutions.

This Chapter also introduces software patterns, and categories of patterns called process and Organizational Patterns. A short history of software patterns is presented, and an explanation as to why patterns are so useful. This section also describes several formats that are currently used to present patterns.

The survey of current research describes how patterns and Organizational Patterns are currently being utilized by groups within the software development industry. In the section describing current research we will look at the state of the art with respect to group theory as it is applied to the domain of software development groups.

2 - Group Dynamics

2.1 - Introduction to Group Theory

Most companies provide their developers with the best possible tools in order to develop highly complex computer systems. We spend hundreds of thousands, often millions of dollars to ensure that our developers have the best technical training and are equipped with the best development environments, compilers, hardware, and other tools. All of this to ensure that the software developed is correct, robust and delivered on time. Yet, time after time we experience projects that run horribly over budget, do not have clearly defined requirements, or do not meet the requirements that were set forth. Perhaps we are not paying enough attention to developing the most productive tool at our disposal.

“The truly committed cooperative group is probably the most productive tool humans have.” (pg 25, Johnson and Johnson 1997)

Johnson and Johnson point out that “creating and maintaining truly committed cooperative groups ... is far from easy. They tend to be rare.” It is no wonder that they are rare. “Does not play well with others,” may be a humorous quote on a software developer’s t-shirt, but it does not bode well for that individual’s effectiveness in a group. Human beings are not born knowing how to interact with one another. We are socialized as children and continue learn how to

interact with others as we grow. Once we reach adulthood, the focus on development tends to be directed toward technical or trades training, which is designed to prepare us for our chosen profession.

Even if we are so lucky as to be a member of an effective group we must always remember that groups change over time. Every member that joins and leaves affects the group interaction. When a group is just forming, more energy and effort is put into communicating and setting norms than in being productive. Once the housekeeping is out of the way the groups productivity increases.

Most groups only ever reach the level of performance shown as a Traditional group in Figure 2.1 - Performance of Groups over Time. Good productivity occurs when a group achieves the effective or co-operative level. The absolute best productivity possible occurs with the rarest of all groups: The Highly Effective Co-operative Group. This should be the goal of every group member.

Figure 2.1 - Performance of Groups over Time

People who have been so fortunate as to be members of a Highly Effective Co-operative Group have described it as something amazing. The productivity of the group far out reaches the expectations of anyone and they have fun doing it. These people often quest for a similar experience throughout their lives. At a Bird's of a Feather presented by the author of this thesis during JavaOne 2002, people in the audience described being a member of a highly effective group as something so positive and memorable that it was like an addiction. Something they quested for in all their subsequent group experiences, but something they were rarely blessed with again. It was the experience of this audience that, in most cases, the occurrence of a Highly Effective group was the accidental result of the combination of the specific individuals working within a specific project at a specific point in time. The audience members were excited to realize that there

are things that they could do to help their current groups evolve toward that elusive goal of becoming a Highly Effective Group.

Most of us have encountered both good and bad experiences when working within groups. Some people are blessed with good instincts but the vast majority cannot define what made the experience either good or bad, or suggest what could have improved it. There is a whole area of social science that specializes in just this problem. It is called Group Dynamics.

"Group Dynamics is the study of behavior in groups to advance our knowledge about the nature of groups, group development, and the interrelations between groups and individuals." (pg 36, Johnson and Johnson 1997)

Group Dynamics is a relatively young field of study and came into its present form in the early 20th century. Group Dynamics has roots in older sciences such as psychology, sociology, and even force field physics (Kurt Lewin). Lewin suggested that "...individuals locomote through different regions of their life-space, being either impelled by forces or drawn by valences that exist along power vectors." As you can see, Group Dynamics is inherently multi-disciplinary.

The key to understanding Group Dynamics is that it is very dynamic. What happens in a group is anything but static. The interactions between group members are influenced by forces such as communication, leadership, goals, and maturity. This complexity often discourages people from pursuing solutions within this domain. It is important to note that there is no silver bullet. Every group is in a constant state of change within a complex matrix of forces.

As a result, there are many theories that have been proposed pertaining to Group Dynamics. A theory identifies a characteristic or set of characteristics that have been identified as relating to a group. The scientists then perform research to validate (confirm), or refute the theory. Once a theory has been validated, it is implemented in practice. Therefore, the solutions to the problems that are identified within Group Dynamics research have been proven to work in a practical situation.

Figure 2.2 - Theory to Research to Practice

Out of Group Dynamics has come the distinction between a Group and a Team. Most people use the terms interchangeably. They are not the same at all. In fact, in order to have a Team you must first have a Group. A Team is simply one kind of small Group. Groups do not become Teams simply because that is what someone calls them. The definition of a team closely parallels the definition of a highly effective group. In this thesis, we will use the terms highly effective group and team interchangeably.

In Johnson and Johnson it says that a Team is a group of two or more individuals who:

- Have mutual goals
- Achieve positive interdependence while striving to achieve their mutual goals
- Interact with each other while working toward achieving their mutual goals
- Are aware of who is a team member and who is not
- Have specific roles (both explicit and implicit), functions and responsibilities that they have a duty to perform
- Have a right to expect other team members to perform their roles
- Have a limited time of membership

A Team is more than the sum of its parts. The team produces products of work that are a result of two or more individuals cooperating as well as individual products of work. Team members freely share information and resources for the common good of the group. They have succeeded in aligning their personal goals with the group goals. All personal agendas have been resolved to either align with the good of the group or neutralized so that they do not adversely affect the group's performance. Each Team member holds themselves and each of the others equally responsible for their individual actions and performance as well as the overall performance of the team. In order for the Team to have cohesion, there must be a specific, clearly defined purpose for the existence of the Team. That purpose must require that they accomplish something concrete.

There are many ways to classify a Team. One way, is to define a Team by the setting within which it functions. For example, a Work Team cooperates and communicates in order to produce a unit of work for the benefit of their mutual employer. An Athletic Team, works together to improve each member's physical

performance and works together to play the best game. A Learning Team has the goal of ensuring that each member learns and improves their skills.

Another way of classifying teams is by their function. For example, a Problem Solving Team obtains its members from various areas of the organization. They work together to produce a solution for a problem. Japanese Quality Circles are an example of this type of team. They often do not have the power to implement their solutions and as a result, they have fallen out of favour. Special Purpose Teams are temporarily formed, to work together to introduce a specific piece of technology or facilitate communication between labor and management. These teams have created a foundation for self-managing teams. Self Managing Teams consist of five to fifteen people who produce an entire product or service. All Team members take turns at all roles including management roles such as scheduling and hiring. It is believed that Self Managing Teams can increase productivity by 30% or more and substantially raise quality. Self Managing Teams are spoken of as the wave of the future.

2.2 - Theories of Group Development

It is a fact that groups change over time. One of the most respected theories, on developmental change within small groups, was introduced by Tuckman (Tuckman 1965, 1977). Tuckman reviewed over 50 studies of small groups, mostly involved in therapy and training activities. He deduced that groups go through five predictable stages. These stages have been labelled forming, storming, norming, performing, and adjourning (or reforming). At each stage, the group focuses on certain issues, as illustrated in Table 1, and this in turn influences the group members' behaviours.

Stage	Main Issue(s)
Forming	<ul style="list-style-type: none"> • Management of social interaction is the main-focus in this stage. • Become oriented towards each other. • Are concerned with being accepted • Strive to learn from other group members. • Are developing a dependency upon each other
Storming	<ul style="list-style-type: none"> • Management of conflict becomes the main issue as group members confront their differences. • May express dissatisfaction • May engage in competitive behavior • May fight. • Develop a counter dependency upon each other.
Norming	<ul style="list-style-type: none"> • The group develops a consensus as to the role structure and group norms. • Negotiate and agree on group norms. • Agree on explicit and implicit roles. • This increases group cohesion and harmony.
Performing	<ul style="list-style-type: none"> • The group is now working as a productive single unit. • Progress is being made towards achieving the group goals. The group becomes more achievement orientated. • We want to quickly develop into, and remain within this stage.
Adjourning (Reforming)	<ul style="list-style-type: none"> • The group disbands or starts the process over by reforming a new group. • Group members engage in activities associated with termination of duties. • Group members engage in actions that reduce their dependency on the group.

Logical Sequence of Stages derived from Johnson & Johnson, (1997) and Scholtes, Joiner & Striebel (2001)

Table 1 - Logical Sequence of Stages over Time

This process of group growth is an iterative one. If a member leaves the group or a new member joins, the process of growth will restart at an earlier stage, often throwing the team back into the first stage. If the group goals change then the

group may re-enter the process at the norming or storming stages. The time it takes to progress through the stages will vary with the maturity and compatibility of the group members. John Warfield says, "The great value in the Tuckman description is that it is right on target, and can be observed almost anywhere that people congregate. Another great value is that the Sequential Stage Theory shows us what we need to eliminate in order to grow an effective group. (Warfield 1976)" Warfield believes that the social overhead, represented by the first three phases of Tuckman's theory, can and should be eliminated in order to boost start the performance of a group. Many theorists, as well as the author, do not agree with Warfield. There is value in going through the process of learning. The key is successfully accelerating the group's growth through the first high-maintenance stages.

Recurring phase theories, such as the Recurring Phase Theory presented by Bales in 1965 are complementary to the Sequential Stage Theory introduced by Tuckman (Tuckman 1977). The recurring phase theories state that issues that affect a group recur throughout its development. This pattern language does not attempt to encompass the recurring phase theory; it may however, be helpful in identifying the issues affecting the group, and provide possible solutions when issues (forces) are revisited by a group. For instance, the partitions that this thesis applies using the Sequential Stage Theory (Tuckman 1977) are fuzzy partitions, in that a group may have to revisit a previous stage for a short period before continuing to advance, or the group may be functioning primarily within the stage before or after the boundary of forces identified as the gating issues.

Many forces apply themselves to the group at all stages of its development. Moreland and Levine (Moreland 1982, 1988) believe that group members go through predictable stages of membership that loosely parallel the stage within which their group is currently functioning. They labelled the stages of group

membership: prospective member, new member, full member, marginal member, and ex-member. These stages represent forces that influence the individual and as a result also influence the development of the group. They and are illustrated in Table 2. Individual goals often seem to conflict with group goals. A team leader will be more effective if they can identify individual goals and try to satisfy them directly or, whenever possible, by integrating them into the larger group goal.

Stage	Main Issue(s)
Prospective Member	Concerned with how the group perceives them.
New Member	The member attempts to change the group to meet their own needs. The group attempts to mould the member to meet the group's needs.
Full Member	The member engages in role negotiation while trying to find a niche for him or herself.
Marginal Member	The member starts to distance themselves from the group. The group starts to distance themselves from the member.
Ex-Member	The member's goals are no longer that of the group.

Table 2 - Logical Stages of Group Membership Over Time

2.3 - Summary of Introduction to Group Theory

- We have introduced the science of Group Theory and have described some of the different types of groups and theories that attempt to facilitate understanding of group dynamics.
- We have also explained the difference between a group and a team.

- We have seen that there are many forces at work upon a group. We have also been introduced to theories that naturally break up the process of group development, such as the Sequential Stage Theory (Tuckman 1977) and the Recurring Stage Theory (Bales 1965).
- We have seen how Highly Effective Groups and/or Teams are considered as very desirable, effective tools. We have discussed how it is hard to grow and maintain Highly Effective Groups and/or Teams.
- This introduction only skims the surface of the knowledge contained in the vast and complex realm of Group Dynamics. We have been encouraged by the wealth of existing solutions to problems encountered by groups that have been researched, tested, verified and continue to be used clinically and in the field. Given the current manner in which these solutions are presented, it is clear that it is nearly impossible to expect an average group member to understand the necessary concepts and then find and apply the correct solutions.

3 - Patterns

3.1 - Introduction to Patterns

Software Patterns were first brought into the spot light in the mid-to-late 1980's. The concept was taken directly from the work of a building architect named Christopher Alexander. His ground breaking works, *A Pattern Language* (Alexander 1977) and *The Timeless Way of Building* (Alexander 1979) introduced his concepts to the world. He wrote that each place was given its character by certain patterns of events that are repeated in that location. Alexander describes problems that occur over and over and the core solution to those problems is presented in a form he described as a Pattern.

For instance, there is a common architectural design described as The Triangle. This is actually an architectural pattern. Within a kitchen, people perform a certain set of events or actions over and over again; such as preparing a meal. In order for them to efficiently perform these actions, the kitchen would need to be structured in such a way as to facilitate their movement. The familiar triangle between the sink, stove and refrigerator is an example of a solution to facilitate the actions performed within a kitchen. Alexander's Patterns were like recipes that provided a common language for describing complex relationships, and describing solutions. Now, when someone mentions a triangle with respect to a kitchen most people understand what is being discussed.

As with different cooks using culinary recipes, Alexander's patterns could be used over and over, by different architects, without the implementation ever being exactly the same, but always achieving the same goals. This repeatability is referred to as being generative, meaning that the pattern can be depended upon to generate a solution. For example, when comparing several kitchens, which all use The Triangle pattern for positioning the sink, stove and refrigerator there will be a lot of variation in the details of the implementation. The goal of facilitating

the movement required to prepare a meal has been achieved in all cases through the implementation of the pattern.

There are as many definitions of a Software Pattern as there are writers of Patterns. The following definitions of Pattern and Pattern Language are generalized from several sources, and are not meant to be the definitive definitions. Alexander wrote "Each pattern is a three-part rule, which expresses a relation between a certain context, a problem, and a solution." For our purposes, a Pattern will be defined as a repeatable solution to a problem within a context. A Pattern Language is defined as a collection of related patterns.

Christopher Alexander also introduced the term "Quality without a Name," or QWAN. QWAN is often described as the "Ah Ha or Eureka moment", a moment when the reader of the pattern has an epiphany and the solution to the problem becomes obvious. Since patterns are well known repeatable solutions to problems, the best patterns are the ones where people say "That is nothing new ... it is obvious." The very fact that it is obvious lends itself to the QWAN affect.

Software developers realized that many of the solutions they were designing were solutions to problems that they or others had already solved. This was obviously inefficient and some people started looking for ways to deal with this problem. Since the details of each implementation, and incompatibilities between the programming languages used to implement the solutions, made it impossible to share or re-use the written code, people looked for other ways to deal with this problem efficiently without reinventing the wheel.

People in the software industry also realized that the number of talented software architects currently available were not sufficient to support the growing industry. There had to be a way to re-use their years of experience and knowledge, or at least a way to communicate it to others with less experience. They thought that if

there was a way to describe the solution to a well known problem, then others would be able to solve similar problems using their known, and tested solutions. A Pattern has a structure that was and is ideal for this purpose.

Object Oriented (OO) Software practices and languages were also gaining favour during this period. The OO proponents were looking at ways to develop software in a more efficient manner. They were the first to look at Christopher Alexander's Patterns and make the connection to the world of software development.

In 1987, several leading minds in the area of software development discovered Alexander's work and started applying it to the design decisions made when developing software. It was Ward Cunningham and Kent Beck who first developed a design pattern language with the goal of documenting the design, and best practices, used to develop user interfaces in the Smalltalk programming language.

In 1994, a landmark book was published by a group affectionately called The Gang of Four (GOF). *Design Patterns, Elements of Reusable Object-Oriented Software* (Gamma 1994) was written by Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides. In it the GOF presented a collection of related Software Patterns in three collections of Pattern Languages: Creational Patterns, Structural Patterns, and Behavioural Patterns. With this work they gave developers the tools they needed to express a problem and its solutions as well as the organizational framework into which other related mined patterns could be placed.

Currently, there are annual conferences that are dedicated to refining new patterns and presenting them to the community. The Pattern Languages of Programming (PLoP) conferences have been running annually since the first historic conference hosted by the Hillside Group in 1994. These PLoP

conferences are now found world-wide, with Euro PLoP, Koala PLoP and Chili PLoP. Patterns are now commonly seen at many other industry conferences, such as Object-Oriented Programming, Systems, Languages and Applications (OOPSLA) and Java One. There are games designed to help developers learn patterns. "GOF Pursuit" was presented at OOPSLA in 1998, by Charles Weir and James Noble from the Microsoft Research Institute. It was a Trivial Pursuit™ type game that encouraged teams to learn the patterns presented in the GOF Patterns book. There is even a pattern language for writing patterns available on the Internet [MES 2001] and in the PLoP 3 book (Martin 1998).

The act of finding and identifying patterns is called pattern mining. When a pattern is mined it is identified and written down in one of the currently acceptable formats. One of the criteria is that the pattern must include examples of where it has been used successfully. Patterns are not new solutions to old problems. They are well known, tested solutions to old problems.

Patterns that are mined become candidate patterns. Generally the process starts when a pattern author puts together a first draft of a candidate pattern. When they approach the community through a PLoP conference or through a mentor they will be assigned a pattern shepherd. A pattern shepherd is a person who has previously authored a pattern, been through the process themselves and agreed to help newcomers. Pattern shepherds volunteer to shepherd new authors through the steps required to have their candidate pattern(s) developed and eventually approved as a true pattern. The pattern will go through several iterations of review and revising, between author and shepherd, before being presented to a group of peers, usually at a pattern conference. The pattern is thoroughly discussed by the group and is completed as a group process. Only when a pattern has been through the entire process is it acknowledged as a pattern and is no longer a pattern candidate. As you may have guessed, the patterns in the thesis

are for the most part, candidates. There are some officially recognized patterns, by authors such as Coplien, McCarthy and Auer etc, that are presented and these will be identified as such.

Many people have gone on to mine patterns in various areas of software development. While those in the area of Object Oriented Development have led the pack in this respect, many people interested in the process of software development have also made significant contributions to patterns community. Patterns started as an inherently cross disciplinary concept. Starting as a concept identified within the realm of building architecture it was applied to software architecture and then to the process used to develop software.

A very interesting paper was presented at the first PLoP conference in 1994 by Jim Coplien. His Pattern Language titled, *A Development Process Generative Pattern Language* (Coplien 1994), was the first example of a design pattern applied to the process of developing software. Coplien has continued to mine and present patterns that provide solutions to problems encountered in the process of developing software. Currently his patterns page lists 42 Process Patterns that he has defined and presented to the community. Some of his patterns are People Patterns and can be seen in the approach Chapter of this thesis.

A book called *Anti-Patterns* (Brown, et al. 1998) presented the idea of a Pattern presenting a solution that sounds good, and is often seen, but simply does not work as expected. An Anti-Pattern results in the opposite of the desired affect. Patterns have given the software community a common language for discussing design issues and their solutions, as well as making solutions widely available.

3.2 - Formats of Patterns

As patterns have gained favour, several different formats for presenting the patterns have been adopted to varying degrees. There are many common formats

including the Alexandrian, Portland, GOF, Compact, AGCS and the Canonical formats. There are also several formats that have been named for the people who have been very active in the world of patterns: Cockburn PM, Beck, and Fowler. The Canonical format is also known as the Coplien format for Jim Coplien's significant contribution to patterns and pattern languages. As you read the following sections you will quickly see that most formats have the key sections in common.

3.2.1 - Alexandrian Format

The Alexandrian form is very flowing and verbose. It has a Zen quality to it when read in its entirety. It quickly fell out of favour to the other often more concise formats.

Alexander (Alexander 1977) proposed the following sections:

- *A picture, which shows an archetypical example of the pattern*
- *An introductory paragraph, which sets the context for the pattern*
- *Three delimiting diamonds*
- *A headline in bold type that gives the essence of the problem in one or two sentences*
- *The body, the longest section: background, motivation, variations*
- *The solution, in bold type: how to solve the problem*
- *A diagram, that shows the solution as a labelled picture*
- *Another three diamonds to terminate the main body*
- *A paragraph that ties the pattern to all the smaller related patterns that round out this one*

This version of the Alexandrian format was almost immediately modified as described in the Portland Pattern Repository (PPR-AF 2002). The new version of the Alexandrian format more closely resembled the following:

- *Title-The name of the pattern. Alexander's names can name the thing created by the pattern, the process of creating it, or some attribute of the solution.*

- *Prolog*-One sentence per pattern that can be expected to precede this one.
- *Problem statement*-One or two sentences that summarize the problem solved by the pattern.
- *Discussion*-Anywhere from 4 to 40 paragraphs that illuminate the system of forces resolved by the pattern.
- *Solution*-One or two sentences that tell you what to do to solve the problem.
- *Diagram*-A picture or two, hand sketched or photographed, that illustrate the pattern (and sometimes the lack of the pattern).
- *Epilogue*-One sentence per pattern that can be expected to follow this one.

3.2.2 - Portland Format

The Portland form was designed to be highly simplified and is often used when presenting patterns on overhead slides. It is the preferred format for patterns submitted to the Portland Pattern Repository, which currently lives and grows on the WikiWiki Web. The WikiWiki Web is a site that is created and maintained by its readers. It is a great source of information on patterns and many other software related topics.

This example is taken directly from the description of the Portland Form on the Portland Pattern Repository (PPR-PF 2001).

Having done thus and such or so and so..." or "At the point in time where this one thing has ended but before this other thing has started" or "Sometimes" or "frequently"... You have some problem. [Context and a brief description of the problem]

"The reason this happens is..." or "this is caused by..." or "Usually this is because of...". "You will want to..." or "It is important that..." or "The primary goal is...". "But..." or "You will also want..." or "Another important issue is...". [Cause of the problem followed by the forces that must be resolved in order, roughly, from strongest to weakest and with conflicts between the forces highlighted]

Therefore:

Take some particular kind of action; or, build something that looks like this; or select this sort of option over that sort. [Describe a solution that resolves the strongest forces in this context.]

"Now you can..." or "Next you will want to..." or "This means that you will have to..." or "You will still need to...". [Describe the resulting context: what has been resolved, what needs to be addressed next, what new possibilities are available at this point, what new problems have arisen, what possibilities are no longer available etc...]

"This is part of the same problem that is addressed by..." or "You may also want to consider..." or "Use this when you've already done thus and such and now you want to move in the direction of doing this other thing..." or "For a fuller description of the larger set of problems for which this is a partial solution see:..." [Summary section. Discuss the greater context in which this pattern belongs, related patterns, and the specific relationships between those patterns and this one.]

3.2.3 - GOF Format

In the GOF "Design Patterns" they indicate that a pattern has "four essential elements:" The pattern name is a catchy name that succinctly describes the

“problem, its solutions and consequences in a word or two.” The problem describes “when to apply the pattern. It explains the problem and its context.” The solution “describes the elements that make up the design (solution).” It does not describe a concrete solution. The solution section describes an abstract description of a problem and the general arrangement of elements and concepts that will lead to its resolution. The final section in the GOF format is the consequences. The consequences are the results, both positive and negative, that will result from using this solution to the problem. This section is extremely important when making decisions regarding which pattern to apply, since no two situations are ever identical.

The Portland Pattern Repository summarizes the GOF format as including the following sections (PPR-GF 2001):

- *Name*
- *Classification*
- *Also Known As (optional)*
- *Motivation*
- *Applicability*
- *Structure*
- *Participants*
- *Collaborations*
- *Consequences*
- *Implementation*
- *Sample Code*
- *Known Uses*
- *Related Patterns*

3.2.4 - Compact Format

The Compact form was suggested by Kent Beck. It is not surprising that the father of Class Responsibility and Collaboration (CRC) cards would suggest a

pattern format that would easily fit on index cards. The idea was to present the pattern in a quick easy to read and understand format. The following is a list of sections proposed for this format:

- *Context*
- *Problem*
- *Forces*
- *Solution*
- *Resulting Context*

3.2.5 - Canonical Format (a.k.a. Coplien format or AGCS format)

The Canonical form is also called the Coplien form after Jim Coplien who was one of the early contributors to software patterns [PPR-CoplF 2001]. The canonical form is also virtually identical to the AGCS form as described in the AGCS Pattern Template (Rising 1998) on page 85. The canonical form is currently to most favoured format and is the format that will be used in the resulting patterns of this thesis. The sections of the Canonical form are as follows [PPR-ConoF 2002]:

- *Name*
- *Alias (multiple or optional)*
- *Problem*
- *Context*
- *Forces*
- *Solution*
- *Example (included at the end of the AGCS form)*
- *Resulting Context*
- *Rationale (optional)*
- *Known Uses*
- *Related Patterns*

The following sections are described in the AGCS form but not listed in most descriptions of the Canonical form.

- *Sketch*
- *Author(s)*
- *Date*
- *References*
- *Keywords*

A very interesting combination of the Canonical form and the GOF form is described in the Portland Repository by Brad Appleton [PPR-ConoF 2002]. He proposed using the Canonical form for the major headings and placing the more verbose GOF sections optionally in the following locations, and using them only where additional information is required.

- *Pattern Name*
- *Aliases (Also Known As)*
- *Intent*
- *Problem*
 - *Motivation*
- *Context*
 - *Applicability*
- *Forces*
- *Solution*
 - *Participants*
 - *Structure*
 - *Collaborations*
- *Resulting Context (also "Resolution of Forces")*
 - *Benefits*
 - *Consequences*
- *Rationale [if needed]*
- *Examples*
 - *Implementation*
 - *Sample Code*
- *Known Uses (sometimes this is part of Examples, or vice-versa)*
- *Related Patterns*

3.2.6 - Cockburn PM Format

Alistair Cockburn has made considerable contributions to many areas of Software Development and the processes used to create software. He used the following format for the process and Organizational Patterns that he presented as described in the Portland Pattern Repository [PPR-CockF 2001].

- Title: *<Author>*
- Thumbnail: *<some difficulty>, so... <some action>.*
- Indications:
 - *<something you notice, hinting that this pattern may be applicable>*
- Contraindications:
 - *<something that would indicate that this pattern would not be applicable>*
- Forces:
 - *<things you want>*
But...
 - *<things that would prevent the obvious solution you would choose from working>*
- Do this:
 - *<Your suggestion>*
- Side Effects:
 - *<new problems that you might expect to crop up upon applying the solution, or new issues that come to the fore. Should not be empty.>*
- Overdose Effect:
 - *<what undesirable thing happens if you keep applying the suggested action over and over and over and over>*
- Related Patterns:
 - *<other patterns that relate, either super ordinate, subordinate, competitor, or neighbouring patterns, with references to where they can be found>*
- Principles:

- *<whatever theoretical grounding you can uncover that would lead you to believe this suggestion should work. This is not so much intended as a section to convince the reader, as it is intended to provide raw material for later pattern researchers to work from, when we get around to investigating principles underlying several dozen or several hundred patterns.>*
- *Examples:*
 - *<the traditional three examples for patterns>*
- *Reading:*
 - *<what to read to understand more about the principles or the pattern>*

Cockburn uses this format in many of his early patterns, but the format did not catch on widely within the community.

3.2.7 - Beck Format

The Beck form was used by Kent Beck in his book *Smalltalk Best Practice Patterns*. It has similarities to the Portland form and the Compact form. The Beck form contains the following sections as described in the Portland Pattern Repository [PPR-BeckF 2002]:

- *Title*
- *Context (optional. May include forward or back references)*
- *Problem -- always phrased in the form of a question the reader might have to ask themselves*
- *Forces*
- *Solution (The solution should include the name of the pattern in some form)*
- *Resulting Context (may include forward or back references)*

3.2.8 - Fowler Format

The Fowler form is named after Martin Fowler who was an early contributor to the patterns movement. It is a very simple format consisting of the following sections taken from the Portland Pattern Repository [PPR-FowlF 2002]:

- *Title*
- *Summary of the pattern (or do this so bad stuff won't happen to you)*
- *Discussion of the bad stuff you avoid by doing this pattern, and how this pattern helps you avoid the bad stuff.*

3.2.9 - Formats of Patterns Summary

As you have seen, most formats for patterns contain many of the same sections. Some formats are better suited to one category of pattern over another. The essential criteria that a pattern must contain are summed up by the authors of Anti Patterns, on their website. (BRO 1998).

Essential Criteria

The following criteria are essential requirements for all software patterns. The ordering of the essential criteria is not important, since they are all mandatory requirements.

1. RULE OF THREE: *A software pattern documents a recurring solution. The pattern solution abstracts three or more experiences. The solution is something which is regularly applied or practiced by some community(s) of sophisticated developers and architects. The logical basis for the rule of three is: the first occurrence shows the design can work, the second occurrence is interesting, and the third occurrence suggests that the design might be worthy of being a pattern because it appears to have a wider applicability. Non-normative comment: The informal concept behind the rule-of-three is: the first occurrence is an event, the second occurrence is a coincidence, and the third occurrence may be a pattern.*

2. PATTERN NAME: *A software pattern has a name corresponding to the documented solution. Within a pattern catalogue, this name is unique.*

3. DOCUMENTATION: *A software pattern is a literary form which contains a documented solution to a software problem and associated design forces.*

4. TEACHING: *A software pattern contains both solution and teaching elements. The teaching elements address the "why?" questions, including the justification and rational explanation of the pattern. The teaching elements include a discussion of context and forces or an elaboration of benefits & consequences in the resulting context. (See TEMPLATE below).*

5. REVIEW: *A software pattern has been reviewed and accepted by qualified peers. Based upon the review, the pattern is revised, as appropriate, by the pattern author(s) in response to peer commentary. It is also highly desirable for the pattern to be reviewed by larger groups, including public review on the Internet (See PRACTICAL FEEDBACK).*

The authors of Anti Patterns go on to describe the desirable criteria that a pattern should contain, and the conditions that warrant that criteria. (Brown 1998).

Desirable Criteria

The following criteria are desirable attributes of software patterns, ordered by priority. The priority corresponds to the relative importance to software pattern quality. The most important criteria are listed first.

1. SOLID SOLUTION: *The solution answers the "what?" and "how?" questions conclusively. A solid solution comprises substantial and valuable technical content. It provides substantial value to new software practitioners or experts not practicing this solution. A formal pattern presentation defines the solution from both abstract and concrete perspectives. In other words, there is an abstract description as well as a specific example. If appropriate, it is desirable that the solution explicitly define a structure, and that the structure be portrayed in a diagram using UML.*

2. *TEMPLATE*: It is highly desirable for a software pattern to be organized in multiple sections (called a template) that address different aspects of the name, problem, and solution. A templated software pattern is documented using a pattern template, containing at a minimum: Name, Problem, and Solution (or equivalent) sections. The purpose of these sections is to address all of the key questions/aspects about a pattern. These template sections are subdivisions of the categories: Name, Problem, and Solution. For example: Gamma and Buschmann patterns both have 14 sections (except different), a CORBA Design Pattern has 12 sections, and a formal Anti-Pattern has 17 sections. In addition to the Name/Problem/Solution sections, the following sections are highly desirable: context, forces, and resulting context (AKA: benefits and consequences).

3. *PRACTICAL FEEDBACK*: Practical feedback is the validation of the software pattern's effectiveness subsequent to its dissemination. Practical Feedback includes review and feedback from expert practitioners who have applied the pattern (See REVIEW). This criteria is in addition to the rule-of-three. In the context of a pattern catalogue or language, a software pattern should contribute to successful software practice. Ideally, practical feedback is validated through software development experience of independent readers. Software patterns should contribute to the creation of architectures, software designs, or IT systems which deliver quality (or tangible value to the software stakeholders).

4. *POPULAR ACCEPTANCE*: It is desirable for all software patterns to be published and publicly available. The popular acceptance and awareness of a software pattern and its pattern catalogue is measured by objective criteria, such as publication circulation and classroom surveys.

3.3 - Process Patterns vs. Organizational Patterns

There are many different categories of software patterns. Among the most well-known are analysis patterns, design patterns, architectural patterns, Organizational Patterns, and Process Patterns. Analysis patterns, design patterns and architectural patterns all relate to the experiences in analysing problems and designing software. The results of implementing these types of patterns are the

architecture and designs, which result in computer source code. We will not be dealing with these types of patterns in this thesis.

Many factors go into successfully developing software. The most highly successful and productive organizations are those that exhibit the same patterns of behaviour, process and organization. Process and Organizational Patterns were derived from the processes and interactions that have been successfully used to organize and schedule projects and improve the positive interdependence within software development teams.

Process patterns are mined from the experiences pertaining to the process of developing software. In 1994 at the PLoP '94 conference, Jim Coplien introduced a key first process pattern language title *A Generative Development Process Pattern Language* (Coplien 1994). His pattern language was published, along with other ground breaking patterns, in *Pattern Languages of Programming Design* in 1995. This pattern language was also published in *The Patterns Handbook* edited by Linda Rising (Rising 1998). Many people and organizations have continued to actively mine patterns that improve the process of software development, in the hopes that other organizations can learn from the experiences of those who have had success in this area.

The very nature of Process Patterns takes into account all the interactions that go into developing software. It is common to see Process Patterns that deal with scheduling, process, and organizational structuring.. Scott Ambler defines a Process Pattern as "... a pattern which describes a proven, successful approach and/or series of actions for developing software. [AMB 2002]" He goes on to state that Process Patterns and Organizational Patterns go hand in hand and should be used together.

Organizational patterns and Process patterns have always had a very close relationship. Organizational patterns deal directly with the management and organization of the people involved in the software development process. Process patterns deal with the structure and organization of the process that people use to develop software. This is a very fine distinction and in practice there is often an overlap between the two categories of process and Organizational Patterns.

3.4 - Problems with the Organization of Existing Process and Organizational Patterns

The focus of process and Organizational Patterns is always on the larger goal of effectively developing robust correct software. When discussing the process of software development, it would be very hard to do without addressing the issues that people face within the process. Unfortunately the People Patterns are often lost in the larger picture. The People Patterns that have been identified within process and Organizational Patterns are often geared toward implementation by a manager or someone with the power to implement process or organizational change. As a result, it is difficult for an individual member of the group to find a solution that they can implement by themselves. This is why People patterns are scattered throughout existing process and Organizational Patterns. They are a step towards the larger goal.

What is the goal of the pattern? This is a fundamental key to identifying People patterns. The reason that People Patterns exist within both Organizational Patterns and Process Patterns is that the focus has always been on accomplishing the larger task or goal of developing software. This thesis proposes that we alter our focus incrementally, to a goal of improving the interaction between group members, in order to grow the group into a highly effective team. Accomplishing this incremental goal would become a step towards the larger goal of developing

software as efficiently as possible. This new goal is the one to keep in mind when mining for People Patterns.

3.5 - Summary of Introduction to Patterns

- Patterns are an easily understood recipe for presenting a repeatable or generative solution to a problem within a context.
- Patterns provide a common language with which to communicate complex solutions to common problems.
- Patterns provide a mechanism to disseminate the knowledge held by the talented, highly skilled and experienced architects, to other less experienced individuals.
- Patterns provide an easily understood format, excellent for organizing our Group Theory solutions.
- Pattern Languages present a format for organizing and disseminating a collection of solutions.
- By adding a new goal, of growing and maintaining highly effective groups, as a stepping stone toward the larger goal of managing and organizing the people who are involved in the process of developing software, we provide a new category of Organizational Patterns called People Patterns..

4 - Current Research on Issues Affecting Software Development Teams

4.1 - Introduction

This survey of current research will describe how patterns and Organizational Patterns are currently being utilized within the software development industry. It will also describe research directly concerned with presenting solutions to the problems that software developer's encounter when working together within a group.

4.2 - Joining Together: Group Theory and Group Skills Sixth Edition

David W. Johnson and Frank P. Johnson (1997)

Johnson and Johnson present a comprehensive introduction to the study of Group Theory. They introduce the history and personalities of this social science. There are several techniques presented including a concept known as experiential learning. One of the great benefits of experiential learning is that the knowledge gained is retained better than with most other methods of learning. The experiential learning occurs by role playing and participating in the many quizzes and exercises. The quizzes and exercises are designed to help the readers understand how they personally deal with situations such as conflicts and what their leadership style is, among many other topics.

Their work is full of exercises that illustrate topics such as communication, leadership, and the value of teamwork. These exercises can and should be executed by people who are interested in exploring the dynamics within their group. The result is a greater understanding of the individual group members and how they interact with each other. The exercises make excellent icebreakers and team building activities for groups wanting to improve their dynamics.

Johnson and Johnson also introduce several theories on group development. It was while studying the book that the author realized that there were solutions to

many of the problems that had been personally encountered while previously working within groups. The idea for the strategy proposed in this thesis is directly derived from exposure to this research.

The main drawback to this research is the manner in which the information is presented. It is comprehensive and contains vast amounts of information that cannot be considered solutions to specific problems. The information in each Chapter is meant for focused study of a particular aspect of group dynamics. The research is presented in-depth on each topic such as leadership style, conflict resolution, communication styles, roles, etc. It does not make the solutions available in a just-in-time fashion. As a result, many people would not take the time to read the valuable information contained in this body of work.

4.3 - The Team Handbook, Second Edition

Peter R. Scholtes, Brian L. Joiner, Barbara J. Streibel (2001)

The Team Handbook (Scholtes 2001) is an excellent workbook containing many People Patterns. Several of the candidate patterns presented in later Chapters come from this work. The authors work hard to sell the Joiner 7 Step Method™. This method uses a scientific approach to analysing the problem and creating a potential solution for future use. It uses the concepts like holding a post mortem of the project, looking at both the things that went well as well as the areas that need improvement. It is a very interesting concept and presented in detail. There are also a number of tools and techniques presented that would be beneficial to anyone creating, managing or working in a group.

The authors have written this book primarily for managers and leaders of groups. Individual group members can use a lot of the material presented, but do not

have the authority to implement some of the ideas. That said, group members would benefit from reading this book and should always be encouraged to suggest some of the ideas presented.

4.4 - *The Pragmatic Programmer: from Journeyman to Master*

Andrew Hunt and David Thomas (2000)

These solutions are exactly as Ward Cunningham states in the forward, "It is a pattern language in sheep's clothing." *The Pragmatic Programmer* (Hunt 2000) is an excellent collection of best practices and tips that should be mandatory reading for all software developers. The explanation of urban decay and how it relates to software development is particularly interesting. It is called *No Broken Windows*. In short, a study on what caused areas of inner cities to become abandoned determined that it started with a single broken window that went un-repaired. Soon there was graffiti on the walls and more broken windows, all because of the perception that nobody cared because no body took responsibility for maintaining the property. We can see this analogy in both software and group development.

Another idea presented is called *Boiled Frog*. This one describes the situation when we are complacent and do not realize that we are in hot water until it is too late. This too, is an analogy pertinent to group as well as software development.

The format of this work is conducive to just-in-time learning. It makes an excellent reference. In fact, a tear out reference card is supplied with the work. The drawback is that, like most works on this topic, its solutions are focused on software development in general and only some of the solutions proposed can be

considered or adopted as People Patterns. It takes time to extract the People Patterns and then it is not obvious as to when they can be applied.

4.5 - The Mythical Man Month

Frederick P. Brooks, JR (1995)

Frederick Brooks' classic collection of essays on the art of software development is full of issues related to group dynamics. His ideas were forward thinking. This is evident in the fact that the twentieth anniversary edition, published in 1995, is still relevant and even more popular than the original. It also presents some very interesting points in the realm of group theory.

Brooks talks about the importance of group goals, communication and roles within the team. He clearly states the need for roles. He presents the concept of a Surgical Team, first attributed to Mills (pg 32). The surgical team is lead by a surgeon and their co-pilot. The surgeon is the boss and maps out the direction the project will take. The Surgeon is also the main point of contact with other teams. This way many small teams can work together without having to manage routes of communication between each member in the group.

In the Chapter titled "Why Did the Tower of Babel Fall?" Brooks describes the failure of the biblical project to build the Tower of Babel and uses it as an analogy for the communication problems that plague software development teams. His essays on scaling teams and the ideal size of a team are directly related to group theory.

Brooks' work was ground breaking and shows that the issues arising from group interactions are as important as other aspects of managing a software

development team. Brooks does not focus on people issues, but mentions them in the larger context of a software project, as does much of the literature in this area. This has the positive affect of bringing these issues into focus, but also muddies the solution by making it too complex.

4.6 - Managing Programming People

Philip W. Metzger (1987)

Metzger has a delightful presentation style as he discusses what it takes to successfully manage programming people. His many colourful anecdotes result in the information he is trying to impart actually staying with the reader. Many of the tips he provides could be written as Organizational Patterns. He presents many solutions to common problems such time management, leading by example, how to praise and how to discipline. He also talks about the roles on the development team and what to look for in potential team members, with respect to interpersonal as well as technical skills. Metzger also touches on the relationships between the analysts and the customers. For example, how important it is to communicate clearly and confirm what has been communicated in writing. One of the concepts that the author works hard at showing its importance is the fact that all members of the team should be treated as members of the team. For instance, documentation and testing are as important as the architect, designers and programmers. He makes it clear that there should never be a hierarchy of perceived importance within a well functioning development team.

As one might guess from the title, Metzger's audience is primarily managers of programming people. His work is interesting in that it contains lots of information that is useful or all team members to know and it translates very well

into organizational and Process Patterns. As with many organizational and Process Patterns, the average team member does not have the influence to implement most of the suggestions. This work is also organized in a fashion that follows the software development process and presents the information based on how a manager should interact with team members in specific roles. This results in it being difficult for the average team member to find solutions to specific problems pertaining to interpersonal issues between team members.

4.7 - The Manager Pool: Patterns for Radical Leadership

Don Sherwood Olson, Carol L. Stimmel (2002)

The Management Pool is the first work of its kind that uses patterns overtly to present solutions to interpersonal issues that software developers encounter while working within a group. The premise upon which this work is organized is by asking the question “what if —instead of a programmer pool— there was a management pool, from which self-selected software developers chose the leader for their project?” The authors ask the question “would you be chosen?”

All of the patterns presented in this work, are aimed at understanding the relationship between software developers and those who lead them. They are targeted at managers, but it is important that development staff be aware of the issues that govern interaction with management, themselves and other team members.

In the pattern named Forty Whacks the authors caution against acting toward your development team as if it was your family. What is significant about this pattern is that it talks about a person’s natural inclination or tendency to feel and act a certain way. It goes on to explain why it is important to not act that way and

gives alternatives. It is almost written like an anti-pattern. The references and anecdotes are reminiscent of the literature written on group theory.

Another pattern presented is called Social Jester. It talks about the problems that occur when a group member is promoted to a leadership position. Many people have felt a chasm widen between them and a colleague when either they or the colleague has been promoted. This pattern explains some of the reasons that this occurs and points out that this distancing undermines the trust that is necessary between managers and their team members. It is important to understand the team members' perceptions of their manager and how the manager can modify their actions and interactions in order to positively alter those perceptions. This pattern advocates the use of humour to reconnect with team members. It is definitely easier to see a manager as 'human' when they have just made a self depreciating joke or acted silly at a team function.

Sixty one patterns are presented organized into five categories: psychological and retentive patterns, behavioural and expulsive patterns, strategic patterns, tactical patterns and environmental patterns. It is full of valuable information that can often be applied to team member-to-team member interactions as well as the targeted manager-to-team member interaction.

The authors use their own format in presenting their patterns. It is organized as free flowing prose including sections called name, picture (an anecdotal context which sets the tone), problem, discussion of the forces and context, and the solution. The format may be unfamiliar to some and down right contentious to die-hard patterns gurus, but it does get the points across in an easy to read manner. The authors admit that they have not always followed The Rule of Three, in that they do not present three specific uses of the pattern. They refer to these patterns as dreams and hope that they will become infectious.

The authors have presented a large volume of patterns and have organized them in a reasonable manner. Even so, it is not always obvious which category your solution lies within. The patterns do not follow a well known format. As with many organizational and process pattern these patterns target managers and not group members. It is a step in the right direction but has not solved the problem that this thesis has identified.

4.8 - The Peopleware Papers

Larry L. Constantine (2001)

In *The Peopleware Papers*, Larry Constantine has attempted to merge the world of group theory and software development. He presents seventy seven essays on many of the challenges faced by people developing software. He does cover the full spectrum of software development and it is clear why this author has been so prolific. His style of presentation is humorous and on target. His essays deal with the outer edges of software development. How software people interact with each other and how they interact with customers. He draws from group theory and uses much of the same terminology as was presented in the introduction to group theory earlier in this thesis.

Many of the essays could easily be re-written as a People Patterns. For instance, his Office Protocol deals with communication within the office. Any developer will tell you that they often work on a problem that takes them several levels deep in logic and a considerable amount of time to get there. Interrupting a developer at this stage, even if well meant, can set result in a lot of time lost and even more frustration. Constantine advocates a vocabulary of interrupts. Taking from software development terminology he advocates IRC (pronounced 'irk') meaning an interrupt request. To which an acceptable response is either ACK

(pronounced 'ak') to acknowledge the request and encourage the speaker to go ahead or NAK (pronounced 'nak' or 'en-nak') to indicate not right now, I'll get back to you later. The benefit of this approach is that you can respond almost without conscious thought and are much less likely to break your concentration.

There are many articles that are more in the realm of process or Organizational Patterns. His target audience is anyone in the field of software development. There is a large amount of information organized in a way that allows the reader to assimilate small portions of it at a time. The organization of the work does not lend itself to just-in-time learning and once read it would be hard to use the work for future reference.

4.9 - Software for Your Head: Core Protocols for Creating and Maintaining Shared Vision
Jim & Michele McCarthy (2002)

The McCarthys successfully showcase a collection of patterns and anti-patterns that address some of the issues from our introduction to group theory. This work targets the group members and makes it clear that every member of the group is responsible for the success or failure of that group. In fact, the basis of the Core Protocol is the equation "Team == Software." The quality of the software is a result of the quality of the team. The Core Protocol evolved over time, through the feedback of workshop attendees and the authors' personal on-the-job experiences. It is a comprehensive volume of work.

For the most part, the patterns apply to all team members. The authors do speak from a management point of view when discussing the importance of responsibility and how managers should always ensure that they are leading by

example and not expecting behaviour from their development groups that they themselves are not already modelling.

The authors have developed many patterns that deal with areas seldom discussed in software development books and courses. They present patterns for understanding and dealing with your feelings. They deal with the importance of common group goals. The McCarthy's discuss understanding personal goals and how to align personal goals with the goals of the group. They present patterns on engaging with other developers and many patterns on improving communication.

A pattern that is particularly interesting is entitled Turf. It deals with how developers will retreat onto their turf and guard it zealously, especially when things are not going well. People are most comfortable on our home turf. They feel safe and in control. The danger with allowing this behaviour on a development team is that ideas are no longer being shared, and the solutions that are implemented may not be the best that was available. The authors do make the point that everyone needs some turf of their own. Ownership of code is very important to the project and the well being of the developers. The danger arises when developers stop sharing ideas and entrench on their turf. As with most things in life, the key is finding the correct balance for your team.

The work is a very large collection of very valuable information. Most, if not all the patterns, presented in this work, are People Patterns. The patterns are arranged into categories that are not immediately obvious to the new reader. Part titles such as Check In, Decider, Aligning, and Shared Vision have more meaning to readers familiar with the Core Protocol. There are quite a few catch phrases and new terminology that is specific to the Core Protocol. As a result, the new reader must learn about these concepts before knowing which part and Chapter will contain a potential solution to their problem.

The McCarthy's use their own format for presenting their patterns. They present a problem, a solution and propose a protocol that governs behaviour during implementation of the solution. Within the protocol, they allow for variations and other conditions that modify the solution. Although it contains some of the same sections (name, problem, and solution), it is not at all like the formats familiar to the world of software patterns. All of this results in a lot of new terminology and organization that the reader has to get past in order to access the excellent information available.

4.10 - Summary of Current Research on Issues Affecting Software Development Teams

The problem of working well within a group has been well identified and understood. As a result there have been many different approaches to presenting solutions to the problems that groups face during their development. The research into group dynamics provides us with a wealth of potential solutions to problems relating to interpersonal interaction. Many authors of books on software development include some solutions to group issues along with descriptions of their process or organizational advancements.

Many researchers present the solutions to an audience of managers, such as in *Managing Programming People* (Metzger 1987), assuming that the reader will have the power or influence to make the group or group member change. Others have developed a full protocol, such as *Software for your Head: Core Protocols for Creating and Maintaining Shared Vision* (McCarthy 2002), that includes solutions to the problems groups face, but requires that the groups adopt their process in order to fully understand and use the solutions. Other research, such as the *Mythical Man Month*, includes really good solutions hidden amongst a collection of anecdotal essays on experiences. Other research, such as *The Pragmatic Programmer* (Hunt 2000) has great solutions, but does not present them in a manner in which people

can easily decide what group of solutions they should be looking for. Most research is not presented in a known and well understood format such as a pattern. Some writers do present the solutions in the form of a pattern, but they have devised their own pattern formats, rather than using a known and accepted format.

This thesis will use many of the solutions and patterns presented in the proceeding research. These solutions will be rewritten into the Canonical format and placed within the organizational framework of a pattern language within a state machine, using the Sequential Stage Theory (Tuckman 1977) to partition the solutions. The patterns chosen will be solutions that any group member can implement. The patterns can be used within any process currently being used by the group. The patterns will be grouped within stage/states that are described in a manner that will allow for clear identification of the stage the team is currently in. This framework will also allow for just-in-time learning, since the user need not read the entire pattern language in order to find and use a single solution.

III - APPROACH

1 - Introduction

There is currently no clear format or tool that allows people to access a solution, to a problem that their group may be encountering, in a contextually relevant and timely manner. This Chapter will describe the approach used to present the solutions to these problems and the evolution of the organizational framework within which the solutions are presented.

The solutions to the problems that software developers face when working together in groups are presented in the form of People Patterns. These People Patterns are solutions mined from many sources, including group theory and existing organizational and Process Patterns. The People Patterns are presented as a pattern language, within an organizational framework.

This framework is in the form of a state machine modelled on the Sequential Stage Theory (Tuckman 1977). Each People Pattern inherits some context from the stage in which the problem it solves is found. Each People Pattern is located in the framework within the stage described by its context, and associated with the transition that will lead the group members to the next optimal stage in their development.

The pattern language of People Patterns organized within a framework that supports the inheritance of context solves the problems identified by this thesis.

2 - Identifying People Patterns

What is a People Pattern? What makes a People Pattern different from an Organizational Pattern or a Process Pattern? This thesis proposes that People

Patterns are currently sub-components of both Process Patterns and Organizational Patterns, due to the simple fact that you cannot resolve organizational or process issues without dealing with the people issues. Human interaction is an integral part of software development. This is why many People Patterns can currently be identified within both Process and Organizational Pattern Languages.

Each type of pattern has its own focus within the realm of Software Development. Process Patterns focus on providing the rules, structure, or game plan for developing software. This includes setting up the project, getting funding, getting buy-in from management, determining which software development paradigm to use etc. In other words, all the structural management that goes into developing software.

Organizational Patterns focus on the management of the people involved in the software development process. This focus has been on the management of the people. This implies that the individual using the pattern has the power and influence of a manager. Unfortunately, this is where the current research stops.

A complexity arises when the issues addressing interpersonal interaction are embedded within the solutions for problems related to the processes used or the organization of the resources required to develop software. By pulling out the solutions to the interpersonal issues (People Patterns) and making them available, independently of the Process and Organizational solutions we can more clearly define the domains of each type of pattern. This in turn will help make it easier for a user to know where to look for solutions.

People Patterns are concerned with the interactions between individual group members, with the goal of growing and maintaining a Highly Effective Group. A People Pattern is concerned with how software development team members

interact with each other. It is concerned with how to improve that interaction in order for the group to grow into a more efficient tool within the software development process. All group members can apply these patterns. This empowers group members and also makes the health of the group their collective responsibility, not the sole responsibility of management. Once the team is working efficiently the complexity surrounding human interaction within Organizational and Process Patterns is reduced to a more manageable level.

In order to differentiate a People Pattern from a Process or Organizational Pattern we must look at the goals of each type of pattern. The reason that People Patterns exist within both Organizational Patterns and Process Patterns is that the focus has always been on accomplishing the larger task or goal of developing software. This goal is shared between Process, Organizational and People Patterns. Process Patterns achieve this goal by providing solutions to problems arising from the steps people take or processes they follow when developing software. Organizational Patterns address the same goal by focusing on the management of the people and resources that are required to develop software. We can think of this as the shared or ultimate goal of People, Process and Organization Patterns.

If we alter our focus incrementally, to a micro-goal of improving the interaction between group members, in order to grow the group into a highly effective team, we build the most powerful tool our software development teams can have. Accomplishing this micro-goal becomes a step towards accomplishing the larger goal of developing software as efficiently and effectively as possible.

3 - Organizational Framework

3.1 - *Divide and Conquer; the Sequential Stage Theory as a State Machine*

Given a collection of People Patterns, without an organizational framework, we would encounter the same problem seen in the current research. The amount of information would be overwhelming, resulting in the information being hard to access. In order to simplify the identification of the current stage of development of the group in question and provide access to the solutions, we use Tuckman's Sequential Stage Theory (Tuckman 1977) to partition the domain of solutions written as People Patterns.

Tuckman defined the stages of group development as Forming, Norming, Storming, Performing and Adjourning/Reforming. We map each of these stages to a state in our State Machine. All groups start as individuals or aggregates. We have added the starting state of Aggregate to the diagrams in order to indicate the transition from non-group to group. This thesis concentrates on the growth from Forming through to Adjourning/Reforming.

As you can see in Figure 3.3, there are valid transitions in addition to the ideal transitions sequentially through the stages in the order prescribed by Tuckman (Forming → Storming → Norming → Performing → Adjourning/Reforming). At any stage in a groups development changes may occur that will cause the group to revert to an earlier stage of growth. Often the addition or loss of a group member will result in this occurring. These are valid transitions, but since they are not optimal transitions and traveling them is the result of factors outside the control of the group, they will not be covered in the current version of this pattern language.

Which previous stage the group reverts to and how long it takes them to move through the stages back to where they started will differ with the maturity level of

the group members and the environment within which they function. In some instances a group that is functioning at the Performing stage will revert all the way back to the Forming stage when a new member is added to the team. When this happens it is imperative that the team takes the time to work through the stages as rapidly as possible. In other instances a group in the Performing stage may only revert to the Storming or Norming stage, for a short while, before they re-enter the Performing stage. What is important is that each group member be aware of the stage the group is currently functioning in and recognize when a change has occurred that requires additional work to get back to the desired stage. Each group member is responsible for the health of the group. This includes educating themselves and each other on these issues.

Also notice in Figure 3.3, that the Performing stage has a self referential transition (labelled 2.5). This indicates that there are issues that must be addressed in order for the group to stay in the Performing stage rather than regressing. Often we see teams that work very well together fall apart after a few months. This is often due to neglect. They are performing so well that it becomes expected and the positive feedback, that they had been receiving, is no longer offered. The group members may also become complacent and this too will cause the group to revert to an earlier less efficient stage of development.

Figure 3.3 - Sequential Stage Theory as a State Transition Diagram

Transitions between stages/states

2.2 - Forming	→	Storming	- Growing from Forming to Storming
2.3 - Storming	→	Norming	- Growing from Storming to Norming
2.4 - Norming	→	Performing	- Growing from Norming to Performing
2.5 - Performing	→	Performing	- Maintaining Optimal Efficiency in Performing
2.6 - Performing	→	Adjourning	- Adjusting to Adjourning/Reforming

3.2 - Context and Gating Issues of each State and Transition

In a state machine there are guards on each transition from one state to another, which must be satisfied in order for the transition to be traversed. In the Sequential Stage Theory (Tuckman 1977) there are gating issues that groups must address at each stage before moving to the next stage of development. These

gating issues are the problems defined in the People Patterns at that stage and associated with a specific transition.

In Figure 3.4, we can see solutions arising from the domain of Group Theory. We partition these solutions by assigning them to state where they are most likely to appear as defined by the Sequential Stage Theory (Tuckman 1977). Each stage/state has a context and each transition has a number of gating issues that must be addressed before the group can move to the next stage.

The gates are on the entrance to the transition, not on the state itself. For instance, the Performing stage has two optimal transitions. One leads back into Performing for continued efficient performance of the group. The second leads to Adjourning/Reforming, which is also a desired transition, such as when the group is disbanded to start new projects with new groups. This means that for the Performing Stage there are two collections of gating issues. These collections of solutions will share a large amount of the context of the Performing stage (meta-context), but there will be additional contextual factors (transitional-context) that will be specific to each transition. The two collections of solutions will share some aspects of the stage's context but the goals of each set of patterns and their associated transitions will be very different from each other.

Figure 3.4 - Partitioning the Problem Domain

Gating issues associated with each stage are as follows:

- 2.2 - Forming → Storming - Growing from Forming to Storming
- 2.3 - Storming → Norming - Growing from Storming to Norming
- 2.4 - Norming → Performing - Growing from Norming to Performing
- 2.5 - Performing → Performing - Staying in the optimal Performing Stage
- 2.6 - Performing → Adjourning - Preparing for the group to disband

The patterns will be organized within each state as relating to a specific transition. The Canonical format will be used and each pattern will inherit the contextual state from the category within which it appears (meta-context). For instance, *In Forming, related to Growing from Forming to Storming* is a category that defines a meta-context. All patterns within this section of the pattern language will inherit context from this meta-context. They may optionally add additional context in the definition of the pattern. The problem stated in each pattern is a single gating

issue known to occur at that stage and that must be resolved in order for the group to grow to the next stage through the associated transition.

Each People Pattern will address a gating issue described as the problem. This thesis will provide a total of thirteen candidate patterns; three People Patterns for gating issues associated with each of the first three described transitions and two for each of the transitions within the Performing stage. There are many more People Patterns that will fit into this framework in the future.

4 - Pattern Language: Growing and Maintaining Highly Effective Groups

4.1 - Introduction

The issues surrounding working in teams permeate our lives. Often, in our technologically focussed world, these issues are ignored or avoided. The skills, required to work well within a group are not explicitly taught, in the Computer Science or Electrical Engineering curriculum of most schools. It is a very interesting fact that many more software development projects fail due to human issues than technological ones. This fact has been identified and re-identified by many different people, ever since people started developing software in teams. How you get individuals to work well together as a team is a well known problem, to which there has not been (nor ever will be) a definitive solution. As Frederick Brooks stated in *The Mythical Man Month*, "There are no silver bullets" (Brooks 1995).

Much research, in the social sciences, has been focused on group dynamics. The goal of this pattern language is to provide a guide for managers, team leaders, facilitators, and group members to identify the stage at which their group is currently functioning and facilitate the group's progress towards becoming an effective group. By identifying the context within which a group is functioning,

and the forces that are currently at work, the process of identifying a solution to the problem(s) the group is facing will be simplified. This in turn will help facilitate the process of accelerating a group's development, to a state in which it is the most productive (an effective group), and keeping it there.

Working well in a team situation is not a problem that is limited to software development. People in other walks of life have the same issues, but seem to address them differently. In *The Peopleware Papers*, Larry Constantine writes, "Many programmers view group development and team building with a certain scepticism, They don't cotton to cheerleading or chants or group games that seem only to support a superficial team spirit. Such touchy-feely stuff is not for them." Constantine goes on to say that he started his book with a Chapter on group theory "...Because understanding the language of groups and group processes may be every bit as important as understanding a programming language. (Constantine 2001)"

Building a team and improving interdependence between team members is more like caring for an infant, than designing a software system. You start with a list of things that may be wrong, prioritize them and address each one in turn. Is the baby in pain, does it need its diaper changed, is it hungry etc. Eventually all the immediate needs of the child have been addressed and hopefully the child is now content. If not, you investigate what else might be wrong. As the child grows and enters the next stage of its development the list of its needs change. When building a team, as when raising a child, it helps to know what the immediate needs of the group are at that particular stage of development. This gives you a list from which to try solutions.

This pattern language provides the organizational structure required to partition the solutions that people face when working together within a group. Software

development groups develop and mature, just as a child does. As they develop through the stages of their growth their needs change. At each stage their needs must be met before they can advance into the next stage of development. As a result the patterns in this language are grouped according to the stage of the group's development. Within each stage, the patterns are associated with the transition that is required.

4.2 - Terminology

Within the science of Group Theory, the terms group and team mean different things. Popular use of the term team as equivalent to group can be confusing because the differences are subtle. Many of the patterns in this pattern language are mined from literature that uses group and team interchangeably. Group theorists prefer the term group, or highly effective group, while popular texts and manuals prefer the term team. In order to reduce the chance of confusion, within this pattern language, the terms group and team will be used interchangeably. For our purposes they are to be considered the same.

Within this pattern language and its organizational state machine, the terms state (referring to the state in the state machine) is synonymous with stage (referring to the stage of growth in the Sequential Stage Theory (Tuckman 1977)). There is a one-to-one mapping between the states in the state machine and the stages in the Sequential Stage Theory (Tuckman 1977).

4.3 - The Framework and How to Use it

The reader is not expected to be a student of group dynamics. Definitions of the commonly used group theory terms are provided in Appendix 1 - Glossary. The reader need not be the group leader. Each group member has a responsibility to contribute to the health and growth of the group and many of the solutions can be implemented by a group member.

The People Patterns are organized into states in a state machine (Figure 4.5 - State Transition Diagram on page 73). Each state maps to a stage in the Sequential Stage Theory (Tuckman 1977). In the pattern language, each state in the state machine describes a meta-context for the corresponding stage in the Sequential Stage Theory (Tuckman 1977). This is the current situation, needs and behaviours typical of people whose groups are at that stage of development. For each stage there is also a list of feelings and behaviours that are common among group members at that stage of the group's development. The meta-context with the list of feelings and behaviours is a diagnostic tool designed to help a user identify which stage their group is currently operating within.

The numbering of the patterns is designed to help the reader remember the context of the pattern. The patterns are numbered using a notation similar to software versions. The first number indicates the stage that the group is currently in. The second number refers to the transition. The third number is the pattern associated with that transition.

e.g.

The notation 4.2.1 refers to:

<Current stage><Transition><Pattern>

<Performing>.<adjusting to adjourning or reforming>.<separation issues>

State Transition Diagram for the Sequential Stage Theory

Legend	Patterns
Normal transition path.	2.1 From Aggregate to Forming
Premature return to the forming stage.	2.2 From Forming to Storming
Stage of the Sequential Stage Theory	2.3 From Storming to Norming
	2.4 From Norming to Performing
	2.5 From Performing to Performing
	2.6 From Performing to Adjourning (or Reforming)

Figure 4.5 - State Transition Diagram

The reader should scan the meta-context, feelings and behaviours sections of the patterns in order to identify the situation in which the reader finds him or herself. Once you have identified the stage/state your group is currently in, you can choose a transition from within that state. A transition is the path that the group

will optimally travel to get to the next stage of development, and state in the state machine. In some cases there are several transitions associated with a particular state.

Gaining an understanding of the group dynamics involved is often enough for the group members to be able to identify the real problem. Once the problem is identified, group members may then be able to resolve the problem, using skills and techniques with which they are already familiar. If existing skills and techniques are not sufficient, the reader can apply some or all of the suggested solutions that are presented as People Patterns. Choosing a potential solution in the form of a People Patterns is accomplished by reading the pattern names and descriptions, associated with the state and transition that you have previously identified. Choose the potential solution that most closely addresses your current problem. As the pattern language grows, more solutions will become available.

Once the gating issues for a specific stage have been resolved, the group situation will begin to resemble the descriptions in the meta-context of the next stage as the group moves into the next stage of group development.

4.4 - The Patterns

Stage: Forming

Forming Meta-context:

In the forming stage, group members are on their best behaviour while they get to know one another. This stage is often nicknamed the Honeymoon stage. Each group member is interested in finding their place within the group. To do this they are interacting with one another to find out each others interests and skills. The group members must learn to trust each other and to build positive interdependence. They need to learn to value each person's contribution to the

goal of the group. The goal of the group should be clearly defined and understood by all group members. This is a time of transition and testing. Group members will test the leader's guidance, both formally and informally. There is so much going on that distracts group members at this stage that it is normal that not much progress is made toward any concrete goals.

Feelings that group members often feel at the forming stage include:

- Excitement, anticipation and optimism
- Pride in being chosen for the team
- Initial tentative attachment to the team
- Suspicion, fear and anxiety about the job ahead

Behaviours often exhibited at the forming stage:

- Attempts to define the tasks and decide how it will be accomplished
- Attempts to determine the acceptable team behaviour and how to deal with team problems
- Decisions on what information is needed
- Lofty, abstract discussions of concepts and issues; or, for some members, impatience with these discussions
- Discussion of symptoms or problems not relevant to the task; difficulty in identifying relevant problems
- Complaints about the organization and barriers to the task

Feelings and Behaviours are from *The Team Handbook* (Scholtes et.al.2001)

FORMING – TRANSITION A: GROWING FROM FORMING TO STORMING

(Transition 2.2 in Figure 4.5 - State Transition Diagram on page 73)

This section contains the patterns that are solutions to some of the problems or gating issues that groups face when they are in the Forming stage and trying to grow into the next stage of group development, Storming.

A-1 -- Getting to Know You

Problem:

A newly formed group consists of many people with different backgrounds, who do not necessarily know or trust each other. The group members do not know enough about each other to determine where each of them will fit into the group or how their skills will benefit the group.

Context:

The group has just been formed. They are a collection of individuals placed together and instructed to perform as a group. A group of individuals, when initially brought together, is not a group. They are a collection of individuals. All completely new groups start out this way. The people are simply in the same place at the same time. Each group member is excited about being chosen for the group and they all want to perform well, but they do not know enough about each other, or trust each other enough to start the process.

Forces:

The members all want to perform well. The members all want to get along with the other group members. The members all have different personalities. These differences make it difficult for some people to initiate a conversation or interaction that would allow them to get to know each other. Some people with an extroverted nature can initially intimidate those who are more introverted. The members all have strengths and weaknesses, both technical and social. The members all have something that they can learn from the other group members. Differing strengths in social skills can impede the sharing of the technical knowledge that is critical for the group to succeed. The members want to be seen as trustworthy, and in turn want to be able to trust their fellow group members. Initiating trust gaining activities is often scary. There is a very real risk of rejection or ridicule when someone shares knowledge about themselves.

Solution:

The group members need to learn each other's strengths and weaknesses in order to find where their skills will best benefit the group. They need to learn to trust each other. The group should actively engage in organized activities that will maximize group trust and cohesiveness, such as ice-breaker exercises. By using organized activities, everyone is expected to participate. This eliminates the need for one person to shoulder the risk by being the first to share personal information. By sharing this information the group benefits by learning to trust each other as well as learning about each other. This knowledge will help in the subsequent stages of the group's development when they are each filling roles within the group.

Example:

A common ice breaker involves group members pairing off and interviewing each other. After a brief time, roughly 10 to 15 minutes, the group members take turns introducing their partners to the group.

Another activity commonly engaged in by new groups is to do something social together, like a team lunch. Some companies have a kick off party for new projects. These both provide the social atmosphere that is more conducive to the sharing of personal information.

Another organized exercise involves a little self learning as well. Organize your team mates to take the Myers Briggs or Kiersey Personality Sorter. This is a list of questions that people answer about themselves. The answers are evaluated to determine which of the main personality types they fall into. It is interesting to see what personality type you are and how the different personality types are known to interact with each other. While this is a controversial test with respect to absolute results, it provides a great way to open the channels of communication. Once everyone in the group has taken the test, they can start discussing the results. It is often fun when people do not agree with the results of the test, it opens the door for communication on a level that is hard to open otherwise. When the results are correct it can help to understand how better to communicate with your team mates.

Resulting Context:

Once this exercise is completed, it is easier for group members to risk expressing their ideas and thoughts on a technical level. They have risked their personal information and the response was positive. This makes it easier to take the risk again. Group members also know a little more about each other. This makes it

easier to identify with other group members, since there is more likely something that they now know that they have in common.

Each group member now knows more about the other group members. This knowledge helps the individual know where they will fit within the groups and what roles each person is most likely to fill.

Rationale:

By telling someone about yourself you are showing trust in that person. By listening and accurately repeating the information told them (without ridicule), they are behaving in a trustworthy manner. This sharing of trust is an important element in building trust between group members. In addition, the information shared helps open the way to new conversations, since people will see that they have something in common. This information will also help in determining what roles people will fill within the group, and how their skills will benefit the group.

The different personalities and experiences can be leveraged as starting points for conversation rather than dividing the group based on differing personalities.

A team lunch or kick-off party are also good choices because people are more apt to talk about themselves and get to know each other in a social atmosphere.

An organized activity will accelerate the process. A series of social gatherings will eventually accomplish the same thing, but it will take a little longer. An organized activity takes away the risk of being the first to initiate the exchange. This accelerates the interactions and sharing of personal and professional information within the group.

Known Uses:

The example ice breaker is derived from the Chapter, "Group Goals and Social Interdependence," in *Joining Together: Group Theory and Group Skills* (Johnson 1997).

Ice breaker exercises are a common practice in most classroom situations at all grade levels.

Team lunches and social gatherings are also common practice in many companies.

A-2 -- Go for the Goal

Problem:

The group is newly formed and has no shared goal or vision. Personal goals and agendas are working against the best interests of the group.

Context:

The group has come together and each member comes with their personal goals and agendas. Everyone expects to get something different from this venture. The group is too new to have developed positive interdependence between the members. Often their management team will have a goal in mind, but it is not disseminated to the rest of the group, or the goals change too often or are unreasonable.

Forces:

The different goals and agendas that each group member carries with them can work against the group goal. Unless the group goal is clearly defined, understood and adopted by the group members, their personal goals will overshadow the

group goal. Each group member wants the group to succeed. If they do not have a goal to measure the group's success against, they will focus on succeeding at their personal goals, some of which may be in direct opposition to the best interests of the group. Some group members may be disillusioned and may not expect the group to succeed, even while they yearn for that success. This can turn into a self-fulfilling prophecy if they are not provided a shared group goal on which to focus. Without knowing exactly what the group is supposed to do, resources can be wasted on irrelevant tasks.

Solution:

Prepare a charter or mission statement to clearly define the team goals. This can be in addition to any company mission statement. The group's goal should be specific to the current task. Clarify the expectations as to resources available to your group. Define the boundaries and conditions on your group. If a group goal is defined, is reasonable and is seen to be attainable, most group members will align their personal goals with that of the group.

This group goal or mission statement can be supplied to the group or be created from within the group. Either will work as long as the goals are reasonable and clearly defined. In *The Team Handbook* (Scholtes et.al.2001), on page C-27, the authors have provided an exercise that the team can work through in order to explore the team's charter or mission statement in depth.

Example:

Clearly define the goal of the project. The group goal or mission statement should state the purpose and scope of the project. It should explain the problem or opportunity that is being addressed, why it is important and how much progress or improvement is expected by the group.

The following example is derived from the CDT project currently being implemented as part of the Eclipse tools project. Eclipse is an open source framework written in Java. It ships with a Java Development Toolset (JDT) and the Plugin Development Environment (PDE). The author is involved in the development of the corresponding C/C++ Development Toolset (CDT).

The following is an example of a mission statement appropriate for the Open Source development of the CDT within Eclipse:

The Eclipse CDT team is responsible for designing and implementing a fully functional, extensible, C/C++ development environment within the Eclipse framework. The design should be fully extensible and provide open APIs that Independent Software Vendors (ISV) can extend in order to implement tools that work with the core C/C++ development environment. A reference implementation of all APIs should also be provided that will provide default functionality for all extensible IDE functionality. The reference implementations should use open source tools such as GNU GDB and Make.

Resulting Context:

The group members now know that there is a shared goal to focus on and work toward. They want to succeed and will now attempt to align their personal goals with that of the group in order for the group and them to achieve success. Those personal goals and agendas that may be in opposition to the goals of the group can now be re-evaluated in light of the group goal. Those group members that may have been sceptical of success will gain confidence when the group has a well defined goal. Most people will realign their goals or otherwise find a way of making their personal success dependent upon the success of their group.

Rationale:

People inherently want to succeed and are willing to work toward that success. They need to know what they are working toward. We have often heard of groups that “floundered.” They had no direction and therefore never went anywhere or achieved anything of note. In all the literature that was reviewed for the thesis a resounding point kept being made again and again. Groups need to know what their goals are. This simple act of providing a goal has kick started many groups on their way to great success.

The group members will now know why they have been brought together. All future actions can be measured against the fulfillment of the group goal or mission statement. If a task does not fulfill the mission statement, it can be argued that it is not a relevant task and the group should not waste time on it.

Known Uses:

At the Birds of a Feather held at JavaOne 2002, on the topic of this thesis, the participants overwhelmingly agreed with this pattern. It was astounding to see so many people testifying that a simple thing like a goal had made such a difference to their groups. It was the shared consensus that far too many teams flounder without knowing what their goal is.

Teams need clearly defined guidelines from within which to work. They need to know the importance of their task within the larger organization. They need to know what resources are available; such as what the timeline is, what resources are available to them. They need to know what decisions they are empowered to make and what other teams or people they need to interact with in order to accomplish their task. (Derived from *The Team Handbook* (Scholtes et.al.2001))

In *Joining Together: Group Theory and Group Skills* (Johnson 1997) the authors summarize the section on Group Goals and Social Interdependence by stating

“Groups exist for a reason. People join groups to achieve goals that they are unable to achieve by themselves. The personal goals of individual group members are linked together by positive interdependence. Group goals result. Group goals direct, channel, motivate, coordinate, energize and guide the behaviour of group members. To be useful, however, group goals have to be clear and operational.”

A-3 -- Our Team

Problem:

The group members do not yet feel that they are part of a whole. They have not yet taken ownership of, or responsibility for, the health and success of the group. They have not developed a sense of positive interdependence. Problems go unaddressed and small problems start to grow into large ones due to entropy.

Context:

The group members are individuals and do not yet feel that they are part of the team. Group members do not know their roles within the group. They are still in the honeymoon stage and avoid conflict.

Forces:

The group members are eager to contribute to the group. They do not necessarily know how to contribute. Problems start to arise, but no one takes responsibility for solving them because they do not feel that it is their place. They may assume that a second group member should fill a certain role and the second group member may feel the same about the first. Misunderstandings occur, but are overlooked because everyone is on their best behaviour.

Solution:

Build a shared feeling of responsibility, by declaring a policy of shared responsibility. All group members are responsible for all aspects of the group's success or failure. All problems, no matter how small must be addressed immediately.

Example:

In *The Pragmatic Programmer* (Hunt 2000) the authors describe a pattern called "The Cat Ate My Source Code." In it they state "One of the cornerstones of the pragmatic philosophy is the idea of taking responsibility for yourself and your actions in terms of your career advancement, your project, and your day-to-day work." They go on to state the fact that no matter how well organized a project is the unexpected will happen. Often it is bad. When this happens and we are responsible we must take responsibility and do what we can to rectify the situation. "We can be proud of our abilities, but we must be honest about our shortcomings—our ignorance as well as our mistakes." Often the thing that goes wrong is not within the realm of our influence, but we are still responsible for doing everything in our power to fulfill the responsibility that we took on by accepting the task.

Also in *The Pragmatic Programmer* (Hunt 2000) the authors present a pattern called "Software Entropy". In this pattern they present the "Broken Window Theory." This theory was first written about with respect to crime in inner cities. Researchers wanted to know why some inner cities are well kept and safe places to live, while others are run down and havens for crime. Their research led them to one surprising cause; one broken window. That is all it took to start the steep slide downward. "One broken window, left un-repaired for any substantial length of time, instils in the inhabitants of the building a sense of abandonment—a sense that the powers that be don't care about the building. So another window

gets broken. People start littering. Graffiti appears. Serious structural damage begins. In a relatively short space of time, the building becomes damaged beyond the owner's desire to fix it, and the sense of abandonment becomes reality."

Resulting Context:

Problems get handled as soon as they are identified. No time is wasted or bad feelings generated by placing blame. Each team members begins to feel that they are an integral part of the team. By making shared responsibility the policy of the team, it circumvents the tendency to place blame. It also allows people to feel that they are contributing to the overall well being of the group. It also empowers the group members. They realize that they have some control that comes along with the responsibility. This helps build positive interdependence.

Rationale:

Problems pertaining to the group's development are critical to address immediately, as this will flow over into all other aspects of the groups productivity and health. Each problem is every team members' problem and therefore their responsibility to do what they can to resolve the problem.

Software development issues must also be addressed immediately. If the full solution is not immediately available, enter comments in the source code or an error message for the user to indicate that the problem is known and is being worked on. The sense of responsibility must encompass all things that are related to the group. By sharing the responsibility for the success or failure of the group, the group members will build positive interdependence and resolve problems quickly.

Known Uses:

The New York City Police Department adheres to the “No Broken Windows” policy. Since the research into inner city decay, they have made every effort to take care of the small issues—that if left alone—would lead to larger problems.

The authors of the book *The Pragmatic Programmer* (Hunt 2000) have identified the two solutions that were presented as solutions to this problem.

Stage: Storming

Storming Meta-context:

In the storming stage group members realize the full extent of the task they have come together to accomplish. Often this results in anxiety and frustration as they do not yet feel that they can accomplish the task. Group members often argue during this stage. They must learn to manage this conflict. Fair fighting rules must be in place. The group members must learn how to deal with dissatisfaction. They must overcome competitive behaviour and determine each member's individual roles. Clearly defined roles and responsibilities need to be identified and adopted by each group member. Each group member must confront their differences and find a way to accommodate each other. Group members are questioning, they need answers. Now is the time to reaffirm the group goal. The group members need to adopt the group goal as something important to focus on and strive towards.

Feelings that group members often feel at the storming stage include:

- Resistance to tasks and methods of work different from what each individual member is comfortable using
- Sharp fluctuations in attitude about the team's chance of success

Behaviours often exhibited at the storming stage:

- Arguing among members even when they agree on the real issue
- Defensiveness and competition; factions and “choosing sides”
- Questioning the wisdom of those who selected this project and appointed the other members of the team
- Establish unrealistic goals; concern about excessive work
- Creation of a perceived “pecking order”; creating disunity, increased tension and jealousy

Feelings and Behaviours are from *The Team Handbook* (Scholtes 2001)

STORMING – TRANSITION A: GROWING FROM STORMING TO NORMING

(Transition 2.3 in Figure 4.5 - State Transition Diagram on page 73)

This section contains the patterns that are solutions to some of the problems or gating issues that groups face when they are in the Storming stage and trying to grow into the next stage of group development, Norming.

A-1 -- Value Your Neighbour

Problem:

Group members are questioning the wisdom of those who choose the members for this team. They are unsure that the team contains the skills and personalities required to accomplish the task assigned to them. Factions may form within the group. These factions or individuals may gang up on one or more team members that are having a harder time to fitting in.

Context:

The team has moved into the storming stage. The honeymoon is definitely over. Team members are questioning almost everything about the project. Team members with stronger personalities are blossoming and may be over shadowing those quieter members of the team. People are starting to jockey for position within the team. Roles are starting to be assigned or assumed.

Forces:

Each team member may be questioning their and their team mates' value within the group. It is common to see group members bragging about their skills and trying to minimize other group members' skills. Sometime this is done because they do not feel that they are being appreciated for their skills. Other times a group member may have valid concerns about another group member's contribution to the group. The skills of some introverted group members may not be obvious or have not yet been revealed.

Solution:

Present each team members skills and abilities in a positive light. Make it clear that each unique skill is required for the team to succeed. Create areas of

expertise, or roles, and assign each group member to a role that showcases their particular talents.

Example:

From his philosophical pattern titled *Values Individuals* Ken Auer states “Figure out his strengths and weaknesses. Find roles, whenever possible, that take advantage of his strengths and publicly recognize how his unique strengths applied to that role are very important to your project.[AUE 1998]”

Joining Together: Group Theory and Group Skills (Johnson 1997) on page 517, uses an analogy of the Plains Indians Medicine Wheel regarding building productive teams. The authors state “The Plains Indians ... follow the Medicine Wheel Way.” The Medicine Wheel represents all things in the universe. All things are equal within the Medicine Wheel. “All things ... know of their harmony with every other thing, except for humans. Finding harmony involves recognizing the uniqueness of each person and his or her unity within the Medicine Wheel.” The Medicine Wheel way says that all humans are unique except that all are equal in their loneliness. The only way to overcome that loneliness is by touching others. “We touch others by joining a team and seeing the unity of all members. We are all separate and unique individuals. We are all part of a larger harmony working together.” They go on to state that each person cannot stand alone, that they need a team. “Each person needs to be:

1. A unique and valued individual success in his or her own right.
2. Part of a successful (and great) team effort.” (Johnson & Johnson 1998)

Resulting Context:

Each group member has a clear understanding of the value of each of their team mates. In turn they are reassured that their worth has been identified and

recognised by the group. Factions will dissolve as group members start to refocus on the goals of the group. They are no longer distracted by their speculations regarding the abilities of their fellow group members.

Rationale:

Bringing the issues into the open and discussing the contributions and expectations of each group member, discourages counter productive attitudes. Assigning roles with clearly defined responsibilities that map to each team members skills, help group members define their boundaries. When we make this information available to all group members, the ones who are questioning fellow group members' abilities are given the information that they have been speculating about. When roles are assigned there is also the opportunity to discuss who would be best suited to that role and why. This also helps to alleviate concerns about the abilities of fellow group members. As well as providing a forum for discussion as to what is expected from each role.

Known Uses:

Ken Auer uses these concepts in his organizational pattern *Values Individuals* [AUE 1998].

A-2 -- Resolving Conflict

Problem:

Two or more group members are feuding and the conflict has grown to the point that someone needs to intervene.

Context:

The group members are starting to assert themselves. This often leads to conflict. Group harmony and communication are the responsibility of each group

member. Group members do not have to be in a defined leadership position in order to intervene.

Forces:

The emotions are high and it is uncomfortable for all group members. Those not directly involved in the conflict are avoiding those who are. No one wants to take sides. Everyone is hoping that the combatants will find a way to smooth over their differences. The team members involved in the conflict may feel that they have to win in order to save face within the group. The need to win the conflict often leads to irrational behaviour. Both team members involved in the conflict ultimately want what is best for the group, but they are not willing to give in if it is at their expense. The group members involved in the conflict may resort to name-calling and other counter productive behaviour.

Solution:

Anticipate and prevent problems. If possible avoid conflict by ensuring that team members do not have existing issues when they are selected for the group. Be on guard for potential problems and deal with them before they grow out of proportion to the original conflict. Set in place rules that state that the group should use the guidelines presented in the Guide for Communication (Table 3) and Rules of Fair Fighting (Table 4).

When intervening keep your reactions in step with the situation; neither overreact nor under react to anything that is said or done. Use non-combative language that is designed to calm the situation and help the combatants adhere to the format presented in the Guide for Communication (Table 3).

The following are the main options that are available to group members when presented with a conflict:

- Do nothing, this could clear up on their own or they could get worse
- Off-line conversation, talk to one or both of the combatants in private, preferably on their turf.
- Impersonal group time, engage the group in a conversation regarding communication and conflict. Keep the conversation impersonal and lay no blame.
- Off-line confrontation, speak directly to all combatants that are perpetuating the conflict or making it worse. Keep this conversation private and on your turf.
- In-group confrontation, speak directly to all combatants that are perpetuating the conflict. This conversation is with the entire group.
- Expulsion from the group, radical option if no other tactics resolve the situation

Remind the group members of Table 3 - Guide for Communication and Table 4 - Rules for Fair Fighting

Example:

In *The Team Handbook* (Scholtes et.al.2001) the authors state that prevention is the best medicine. Do not select members for the team who are known to have issues between them. A "Self Selecting Team" [COP2] will help avoid this problem. Jim Coplien created the Organizational Pattern that addresses this issue. A Self Selecting Team has input into the selection process when hiring or assigning members to the team.

The guide for communication is derived from *The Team Handbook* (Scholtes et.al.2001). The same guidelines have also been presented in both Parenting and Marriage workshops that the author has participated in.

Guide for Communication

Sequence	Explanation
"When You ..."	Start with a "When you ..." statement that describes the behaviour without judgement, exaggeration, labelling, attribution, or motives. Just state the facts as specifically as possible
"I feel ..."	Tell how the behaviour affects you. If you need more than a word or two to describe the feeling, it is probably just some variation of joy, sorrow, anger, or fear.
"Because I ..."	Now say why you are affected that way. Describe the connection between the facts you observed and the feelings they provoke in you.
Pause for discussion	Let the other person respond
"I would like ..."	Describe the change you want the other person to consider...
"Because ..."	...and why you think the change will alleviate the problem.
"What do you think?"	Listen to the other person's response. Be prepared to discuss options and compromise on a solution.

How the feedback will work:

When you [do this], I feel [this way], because [such and such]. (Pause.)
What I would like you to consider is [doing X], because I think it will accomplish [Y]. What do you think?

Example:

“When you are late for meetings, I get angry because I think it is wasting the time of all the other team members and we are not able to get through our agenda items. (Pause.) I would like you to consider finding some way of planning your schedule that lets you get to these meetings on time, That way we can be more productive at the meetings and we can all keep to our tight schedules.”

(Derived from The Team Handbook Scholtes et. al 2001)

Table 3 - Guide for Communication

The rules of fair fighting may seem like common sense, but we often forget them in the heat of conflict. By clearly stating what the rules are and making all group members aware of them, many conflicts will resolve themselves quickly.

Rules of Fair Fighting

Talk about your feelings and say what you want to be different. Do not lay blame.
Do not overreact to small things and turn them into major conflict.
No hitting below the belt. (i.e. Saying things to deliberately hurt)
Don't try to make the other person feel bad about them selves or feel guilty.
Don't use silence or withdrawal of contact as a battle tactic.

Don't behave in a cold and distant manner.
Avoid personal attacks on the other person.
Avoid interpreting the other person's motivation or behavior.
Make your fight specific.
If tempers are too hot, agree to discuss the issue at a later time.
Talk calmly, slowly and stick to the point or the other person is likely to tune you out.
Do not bring up old fights.
Do not become addicted to being right all the time.
No one can win a fight. A fair fight is one where you each feel that you are emotionally okay with the outcome.
No warnings about resorting to violence.
No violence!

Table 4 - Rules for Fair Fighting

Resulting Context:

The immediate conflict has been resolved with both participants feeling that their side of the argument was heard. Groundwork has been laid down for future conflicts with the presentation of the Rules of Fair Fighting and the Guide to Communication.

Rationale:

Many issues may be at the root of any conflict. No matter what the reason for the conflict is, all group members must behave appropriately when confronted with a frustrating situation. Have clearly defined rules helps guide group members when

without the rules their emotions might win out over common sense and group harmony.

By involving all group members in the conflict resolution, they are reassured that the situation is or can be resolved. By taking a no-blame approach to defining the rules of communication no one is pressured to take sides, and the combatants are given a way out that will allow them to save face with their group members. By having the group address the conflict as a group issue, the individuals who were behaving inappropriately get the message that the behaviour will not be tolerated. Doing this without confrontation works best because the group members are never put in a situation of having to win or take sides.

Known Uses:

Parenting courses and marriage courses use both techniques quite extensively. A quick search on the internet for “Rules of Fair Fighting” will present over 100 relevant references.

The Guide to Communication is presented in *The Team Handbook* (Scholtes et.al.2001). The ideas that it presents can be found in many resources; such as *The Nonviolent Communication Model* [ROS], published by Marshall Rosenberg with excerpts available from the web.

A-3 -- Harnessing Competitive Energy

Problem:

Group members are competing with each other for resources and ownership of knowledge. One or more group members are hoarding knowledge. They may be refusing to write their design documents, or to discuss the design and technology that is required for their part of the project. They take on every problem that

arises within their domain, even when they do not have the time to complete the job successfully.

Context:

Each group member was chosen based on the skills and knowledge that they can bring to the group. It is essential that these skills are taught and the knowledge shared, in order for the group to function as a highly effective group. The hoarding of knowledge is counter productive to the goal of mutual interdependence.

Forces:

People like being known as the expert on a subject. Some people feel that by sharing their knowledge, they lose some of the power they otherwise would have as the expert on that topic. It is essential that knowledge be shared in order to achieve positive interdependence and for the group to achieve their goals. In addition, by taking on all tasks associated with their domain they may be over extending themselves when other group members have extra time that would be better spent on that task.

Solution:

Convince the team member that is hoarding knowledge that it is in the best interest of the team for them to share their knowledge. In turn they will learn from the other team members. A formalized program of lunch-time learning is a valuable approach. Several half hour blocks of time are set aside each month. People are assigned a time when they are to present to the rest of the group a topic with which they are most familiar. Everyone takes a turn presenting and attendance is mandatory.

Example:

The pattern titled “Overbearing Participant” from *The Team Handbook* (Scholtes 2001), suggests harnessing the skills of the group member who is hoarding knowledge. They suggest that the group reinforce that there is no area of expertise that is untouchable. The project is owned by everyone on the team. The most successful way of turning around this hoarding behaviour is to celebrate the expertise of each group member. Acknowledge the individuals expertise and enlist their aid in educating the other team members on the topic with which they are an expert.

Resulting Context:

The knowledge is now being shared among all group members. Group members are learning from each other and all members are better able to fulfill all tasks required of the group. This shares the knowledge, responsibility and uses the resources within the team more efficiently.

Rationale:

The group members must share their knowledge in order for all group members to be able to contribute fully to the group’s goals. This is critical with software development. If the design of one component is not shared then other group members cannot provide information that would improve the overall design and ensure that all components interact with each other as they should. In addition, each member of the team has an obligation to mentor the other group members, just as they each have a right to be mentored.

People who like being considered an expert are usually flattered when asked to present on the topic. By setting up a situation where everyone is expected to share knowledge, the knowledge hoarder does not feel singled out. By sharing the

knowledge, the responsibility is also shared. This makes for a better distribution of responsibility and more efficient use of technical resources.

Known Uses:

The pattern titled “Overbearing Participant” from *The Team Handbook* (Scholtes 2001), suggests harnessing the hoarding team members pride in their knowledge by getting them to formally train other team members.

The concept of lunch-time learning or brown bag lunches is common in the field of high tech. Presentations are scheduled to take place over the lunch hour. Participants bring their lunches and eat during the presentation. These are an informal and valuable low-cost tool for disseminating information.

Stage: Norming

Norming Meta-context:

During the norming stage group members start reconciling their differences. Competitive behaviour falls away as each group member accepts their coworkers and group cooperation and positive interdependence grows. Members start to align personal goals with the group goal. Groups members must form a consensus as to group norms; the rules of acceptable behaviour. They must agree on group rules and define a group process for dealing with conflict. At the end of this stage each group member will have a good understanding of their role within the group. Now is the time to engage in activities that increase group cohesion and harmony. Group members start being productive in this stage.

Feelings that group members often feel at the norming stage include:

- A sense of team cohesion, a common spirit and goals

- Acceptance of membership in the team
- Relief that it seems everything is going to work out

Behaviours often exhibited at the norming stage:

- An attempt to achieve harmony by avoiding (or managing) conflict
- More friendliness, confiding in each other, and sharing of personal problems; discussing the team's dynamics
- A new ability to express criticism constructively
- Establishing and maintaining team ground rules and boundaries (the "norms")

Feelings and Behaviours are from *The Team Handbook* (Scholtes et.al.2001)

NORMING – TRANSITION A: GROWING FROM NORMING TO PERFORMING

(Transition 2.4 in Figure 4.5 - State Transition Diagram on page 73)

This section contains the patterns that are solutions to some of the problems or gating issues that groups face when they are in the Norming stage and trying to grow into the next and optimal stage of group development, Performing.

A-1 -- Roles, Rights & Responsibilities

Problem:

Group member's roles may not be fully defined or not fully understood. Some group members are not living up to their roles or their group member's expectations.

Context:

The group members have all been brought together as a complementary skill set. This includes their ability to fulfill the crucial implicit roles within the group as well as their technical skills that have gained them their explicit roles. Each member of the group has different expectations regarding the behaviour of themselves and their fellow group members. Some group members have higher expectations than others. The group has gotten to know each other quite well at this point and have learned how to communicate with each other. The group members are past the point of engaging in fights, but the annoying little habits and behaviours are taking their toll on the serenity of the group.

Forces:

Some roles have not been fully defined or were not well defined, which has left the group members unsure of what is expected of them. The people filling some

roles may be unaware that they are not fulfilling some of their responsibilities. The group members want to succeed in their rolls, but they may be unsure of what that fully entails.

Solution:

The roles (both explicit and implicit) that individuals are filling must be fully defined and agreed upon. The fact that one role's right is another role's obligation must be fully understood by every member of the group. People need to know what is expected of them beyond what their job description says. Group members must actively agree to their roles and responsibilities in order to ensure that all group members are accountable. Implicit roles are often not assigned, but group members fall into them as they gain confidence within the group.

Example:

In *The Mythical Man Month* (Brooks 1995) Brooks introduces the explicit roles he considers critical for a software development team. Described as "The Surgical Team" each person on the team has well defined explicit roles as well as implicit roles that they will pick up depending on their interests and proficiencies. He contends that an efficient team should consist of ten or fewer people. Brooks presents the roles defined by Harlan Mills:

- The surgeon or chief programmer. All decisions flow through him and he has final say on the architecture and design of the system being built. Mills recommends that the surgeon have ten or more years of experience.
- The co-pilot. He is the surgeon's alter ego. He is able to fulfill any part of the surgeon's job, but is less experienced. His main role is to provide feedback to the surgeon regarding design and architecture.

- The administrator. While the surgeon has the last word on raises, space, personnel etc. He must spend virtually none of his time on these matters. If the project is large enough this should be a full time position otherwise one person can fill this role for two teams.
- The editor. The surgeon is responsible for generating the documentation regarding the project. The editor takes the first draft and reworks it into its final form.
- Two secretaries. One each for the administrator and the editor,
- The program clerk. He is responsible for maintaining all record both human and machine produced. It is his responsibility to present all programs and data as team property, rather than belonging to the developer who created them.
- The toolsmith. The toolsmith is responsible for installing and maintaining all tools required by the development team. This may include a centralized CVS repository, integrated development environments and modelling tools.
- The tester. Is responsible for providing test cases and suites of regressions an integration tests for all the surgeons code.
- Language lawyer. One language lawyer can service two or three surgeons. He is responsible for finding the most efficient and elegant way of implementing what the surgeon has designed. He is an expert in the language that the project is being written in and is a valuable asset that is consulted by all individuals who are writing code.

In *The Team Handbook* (Scholtes et.al.2001) the authors present both implicit roles that they feel are crucial to every team, and often overlooked. The following implicit roles are adapted from *The Team Handbook* (Scholtes et.al.2001).

Task Oriented

- Information giver - finds and supplies pertinent information for the group.
- Initiator - states pertinent beliefs about the task and others' suggestions, designed to initiate discussion.
- Elaborator - builds on suggestions made by others and considers the implications of those decisions.
- Clarifier - gives relevant examples, offers rationales, and probes for meaning by asking questions or restating problems.
- Tester - raises questions to test out whether group is ready to make a final decision.
- Summarizer - reviews or tries to pull together the discussion content so that it is clear and easily understood.

Maintenance Oriented

- Harmoniser - mediates differences of opinions between group members.
- Encourager - praises and supports others in their contributions to the group.

- Compromiser - imagines options that satisfy mutual needs, recognizes differences between group members.
- Group Clown - draws off negative feelings with humour.
- Gate keeper - keeps communications open and creates opportunities for others to participate in all communication.
- Den Mother - tends to the emotional growth and group development.

Resulting Context:

The group members are all aware of what their roles are within the team, as well as what each of their team mates expect from them in that role. Frustration has been alleviated by clearly defining the rights and obligations that go along with the roles. Group members feel that their worth to the group has been identified and appreciated, as they have been offered implicit roles that suite their personalities.

Rationale:

In *Joining Together: Group Theory and Group Skills* (Johnson 1997), Johnson and Johnson clearly state the importance of the group knowing that the obligation of one role is the right of another. The importance of clearly defined roles is repeated throughout the literature on group theory and software development as the references in the examples indicate.

It is vital that people know what is expected of them. Often job descriptions are supplied, but they tend to focus on the explicit or technical skills. It is as

important to define the implicit roles that group members will fill. It is important for group members to encourage each other in taking on the implicit roles within the group.

Known Uses:

Most companies provide a job description, which is essentially a list of explicit roles.

The Mythical Man Month states that roles are vital to the success of a software development team.

The Team Handbook (Scholtes 2001) and *Joining Together: Group Theory and Group Skills* (Johnson 1997) both spend considerable time explaining the importance of well defined roles within the group.

A-2 -- Only Productive Meetings

Problem:

Group members are beginning to dread meetings as a colossal waste of time. They may have had bad experiences in the past or this group's meetings may be very unproductive and boring. Group members may be skipping meetings or coming in late and generally making the situation worse. By missing status meetings they are missing crucial information and mistakes are being made. These mistakes are slowing down the progress of the group and creating tension among group members.

Context:

Within a group, often the only time to catch up on what every group member is working on and catch potential problems or conflicts is to discuss the group's

status within a weekly or short daily meeting. The group members are starting to be productive and they like it. Any interruption in their productivity is met with resistance. This is especially true of developer's as they get into the depths of their craft they resent any interruptions.

Forces:

The group members are productive and feeling good about contributing to the goals of the group. Any interruption is frustrating for them and they may not see the benefit of a weekly meeting. Tension is increasing and some group members are protesting by not attending or chronically arriving late to meetings.

Solution:

Agree on a set of rules for running meetings.

Example:

Some processes such as Extreme Programming advocate very short daily meetings that are held standing up.

Ensure that every group member is fully engaged in meetings. Check-in is a commitment that the group member makes. Each group member who checks-in consciously focuses their attention on the meeting and connects with the other attendees. Discuss and adopt the concept of "Check In" as discussed in *Software for your Head: Core Protocols for Creating and Maintaining Shared Vision* (McCarthy 2002).

The Team Handbook (Scholtes 2001) advocates defining a set of rules for all team meetings. If all group members have input into and agree on the rules, the problems will be resolved.

The Team Handbook (Scholtes 2001) provides a list of rules for meetings. The list includes:

- Use an agenda to keep the meeting on track. Agendas, organize and focus everyone on the goals.
- Send the agenda out to all attendees, prior to the meeting so that everyone knows what to expect and can be prepared.
- Respect everyone's time. Everyone should treat the meeting as if they had traveled 100 miles to attend. Give the attendees your full attention for the duration of the meeting. Turn off the cell phones and any other gadget that might interrupt the meeting.
- Assign action items that make it clear that the task is assigned to a specific person and that they are responsible for having it completed within an agreed upon time frame.

Resulting Context:

All group members understand why the meetings are important. They feel that their need to spend time on only productive tasks is understood and being taken into consideration. Group members are attending the short productive meetings and progress continues to be made toward the group goals.

Rationale:

Keeping meetings constructive may seem like common sense. People like to converse and will often digress from the topic of the meeting. By publishing an agenda and then sticking to it these digressions are kept to a minimum. Everyone's time is important and it is seen as disrespectful to allow interruptions in a meeting, such as cell phone or a late arrival.

When there is a clear purpose and direction for meeting, people will be more likely to attend and contribute. When everyone is communicating, potential problems will be avoided and everyone will have a clearer understanding of how the group is progressing towards their goals.

Known Uses:

The Team Handbook (Scholtes 2001) uses rules for meetings and goes a step farther by advocating a sample agenda for a first meeting and subsequent meetings.

Software for your Head: Core Protocols for Creating and Maintaining Shared Vision (McCarthy 2002) on page 41, discusses the problems arising from ineffectual meetings and proposes that participants speak up in order to get the format changed so that the meetings become productive. They advocate something they call "Check in at Meetings." Check In is a concept where you consciously focus on the task at hand. The authors provide the following list of feelings and behaviours that are typical of teams that have this focus and are checked-in.

- You laugh a lot.
- You ask team mates for help the moment you suspect you might be stuck.
- You finish team mate's sentences. (Sentence finishing is a way of saying that you can anticipate where your team mate is going with the idea and you are headed there with them.)
- You need only a look to communicate a complete idea. Much of the time you are aware of what other team mates are thinking.
- You can solve difficult problems in real time.

- You can make big changes with minimal discomfort.
- You are focused only on results.
- You are not afraid to let team mates feel things. You encourage one another to feel.
- You cry with one another.
- You feel like your work is an integral part of your life.
- You feel that you can solve any problem that is presented.
- You tackle the biggest problems that you can imagine and solve them.
- You are willing to be patient and wait for a big idea if something seems like it will require too much effort.
- You only hire someone for the team who will push you to be greater.
- You don't solve problems by automatically requiring more time, money or people.
- You have a constant flow of good ideas that you share with your team mates.
- You are always willing to drop your idea for a better one.

The McCarthy's go on to state "The process of developing high-tech products relies on team presence. This relationship is particularly crucial if you are aiming for great products. If you aren't present, you can't possibly be great."

A-3 -- Creative Communication

Problem:

The group members all have different communication styles, which come into conflict at times. The group members misconstrue the communication styles of the other group members. Or group members are using inappropriate means of communication in certain circumstances and interrupting their group members at inconvenient times. This is leading to increasing tensions, reduced productivity and potential conflict.

Context:

The group members need to communicate to discuss issues that arise every day. Group mates need to communicate on many levels and priorities. The group is starting to be productive. People are not taking enough time as to ensure that they are understood and that no misunderstandings occur. Some people are quite terse when they are deep in thought and busy. They may resent the intrusion or they may just be trying to maintain their train of thought.

Forces:

People may be appearing at a group member's desk with their questions. If the group member is deep in thought with a complex algorithm and interruption at this point in time could put them back several hours. Many different types of electronic and traditional communication are available to group members. The group has not defined rules for how to use these various forms of communication.

Solution:

Define rules for communication within the group. What you say, how you say it and when to say it are all equally important. The Chapter “Communicate!” in *The Pragmatic Programmer* (Hunt 2000) provides a list of reasonable rules.

Agree upon what methods of communication will be used in which circumstances. There are many communication methods available to groups. The group needs to define when and how to use these methods of communication in order to be most productive.

Example:

The Pragmatic Programmer (Hunt 2000) presents a Chapter called “Communicate! (pg 18)” In it they provide a list of key points that will help the team members to communicate more clearly.

- Know what you want to say. It may be beneficial to first produce an outline in order to organize your thought before presenting your ideas to anyone.
- Know your audience. In order to communicate there must be a shared frame of reference in order to convey information. Talk in terms of how your information impacts your audience.
- Choose your moment. Try to understand your audience’s priorities. Someone who is racing to meet a deadline or who is deep in complex thought does not need to be interrupted with a request that can wait.
- Choose a style. The person you are talking to may want all the facts or an executive summary. They may only want to know what you need

them to do or they may want to know the context within which the question is relevant.

- Make it look good. “Too many developers (and their managers) concentrate solely on content when producing written documents. ... Any chef will tell you that you can slave in the kitchen for hours only to ruin your efforts with poor presentation”
- Involve your audience. In software development often the process we go through to produce a document is more important than the resulting document. Design documents fall into this category. In this case, be sure to involve you audience early to gain maximum benefit and feedback.
- Be a listener. There is one technique that you need to use if you want people to listen to you ... listen to them. Even if you already know all the information, if you do not listen they will not listen to you when it is your turn. When possible encourage people to ask questions. Turn the presentation into a dialog.
- Get back to people. Be sure to respond to all types of communication. A question asked by email or instant messenger is just as valid a question as one asked to your face. It deserves as much care in the answering of it.

In *The Peopleware Papers* Larry Constantine presents a section called “Office Protocol (pg 29).” In it he talks about the usual ways of communicating being cumbersome and unwieldy in a collaborative development environment. By the time someone has excused themselves and asked if they are interrupting ... they already have and you have forgotten what line of code you were just studying.

“Working groups need a vocabulary of interruptions that is short, sweet and simple. What works for hardware seems to work for people, so in our offices we IRQ, we ACK, and we NAK.”

- IRQ is short for interrupt request. (Pronounced ‘irk’)
- ACK is short for a positive acknowledgement or “OK go ahead.”. (Pronounced ‘ack’)
- NAK is short for negative acknowledgement or “Not Now!” (Pronounced ‘en-ack’ or ‘nack’)

The shortness of the protocol makes it possible to make yourself known without interrupting a train of thought. It becomes an almost instinctive response requiring little thought.

Another option is to ensure that all developers are using the communication technology that is available to them. The following is a list of communication methods and their associated priorities. These priorities will be ordered based on the communication style and preferences of your organization or group. If something is of lower priority send the request via the method that is least intrusive to the receiver.

Communication Technology	Level of Intrusiveness
In Person	Highest
Telephone	High
Instant Message	Medium
Email	Low

News Groups	Lowest
-------------	--------

Table 5 - Communication Technology by Priority

Techniques that have been used successfully to indicate that a group member does not want to be disturbed include: wearing headphones, or displaying a sign on the cubical opening. One of my colleagues belonged to a group that used a small bright orange plastic pumpkin, left over from Halloween. It could not be missed when sitting atop a developer's monitor. Whatever indication is chosen, all group members must be aware of it in order for it to be effective.

Resulting Context:

All group members know what is expected of them in terms of communication technique and style. By communicating while deciding on the rules and techniques, that their group will use, all group members learn a little more about each other's communication style and have input into the group's style. Now when someone needs to ask a question they can choose the most appropriate form of communicating without the risk of interrupting. Tensions are reduced and productivity is once again on the rise.

Rationale:

Some development groups who are sitting in an open concept of multiple cubes find it very difficult to indicate that they are busy. These techniques have to be agreed upon and understood by all group members.

Whatever rules are decided upon should be a group decision with all group members having input. By involving the group members in the decision process they are more likely to adopt the new rules.

Once rules are in place the group members will only be interrupted when it is an extremely important matter, otherwise the question will be waiting for them when they have the time to answer it. This will lead to more productive use of developer time and lessen the frustration that the interruptions may have been causing.

Known Uses:

The Pragmatic Programmer (Hunt 2000) and *The Peopleware Papers* (Constantine 2001) both advocate rules for interrupting to communicate. Communication as a topic is extensive and its rules for clear communication are addressed in *The Team Handbook* (Scholtes 2001) and *Joining Together: Group Theory and Group Skills* (Johnson 1997).

Stage: Performing

Performing Meta-context:

By the time a group is in the performing stage they have formed a productive single unit. They are fully performing their duties; diagnosing and solving problems, choosing and implementing changes. The group is getting a lot of work done. Each group member has discovered and accepted each others strengths and weaknesses. The group norms have been fully defined and accepted. Roles within the group have been resolved.

Feelings that group members often feel at the performing stage include:

- Members having insights into personal and group processes, and better understanding of each other's strengths and weaknesses

- Satisfaction at the team's progress
- Close attachment to the team

Behaviours often exhibited at the performing stage:

- Constructive self-change
- Ability to prevent or work through group problems

Feelings and Behaviours are from *The Team Handbook* (Scholtes et.al.2001)

PERFORMING – TRANSITION A: MAINTAINING OPTIMAL EFFICIENCY IN PERFORMING

(Self-referential Transition 2.4 in Figure 4.5 - State Transition Diagram on page 73)

This section contains the patterns that are solutions to some of the problems or gating issues that groups face when they are in the Performing stage and trying to stay in this, the most productive stage of group development. Without a conscious effort to stay in this stage the group will eventually regress to less productive stages of group development. When this happens they will have to work through the gating issues in order to return to the Performing stage.

TRANSITIONAL CONTEXT:

When a group is actively in the performing stage there is a natural inclination to feel like they have made it. They worked hard to get to this point, now all they have to do is get their work done. Growing teams is hard ongoing work. It is

important that the group not risk missing issues that might need immediate attention. The group members need to acknowledge the progress being made toward group goals. They must be encouraged to continue to become more achievement-oriented. It is important to continually assess personal and group performance. Highly Effective groups are distinguished by the emotional attachment that they feel for each other. There is a sense of an extended family. In order to create an atmosphere for a Highly Effective Group to form it is important to actively engage the group in activities that bring them together outside of the work environment. Encourage strong friendships to form. Never forget to celebrate successes.

A-1 -- Celebrate Success

Problem:

There is now a tendency to take the group's current success for granted and to neglect celebrating the successful completion of milestones. This leads to feelings of neglect and is having a negative affect on the group's morale. If these feelings are allowed to continue the lowered morale will inevitably lead to a reduction in productivity.

Context:

The group has been performing very well and is being very productive. Tasks are being completed on time and are of good quality. The group no longer requires the attentive touch it did while it was in its formative stages.

Forces:

The group is performing well and all milestones are being met. The management team is confident in the group and have focused their attention on other pressing

matters. The group members are enjoying this hard won autonomy, but are concerned that management will not notice their future success.

Solution:

The group members must celebrate the milestones they have achieved. They must take the time to mark the successes that the group has worked so hard to achieve. Celebrate personal successes and special occasions in addition to professional ones.

Example:

The Team Handbook (Scholtes 2001) advocates celebrating little successes. These can be marked by going bowling or go-carting as a group. Often something silly, that the group members would not participate in on their own, serves as a team building exercise as well as a celebration.

Organize a team accomplishment lunch. While everyone is enjoying their meal have them write on a sticky note one or more accomplishments that they achieved since the last accomplishment lunch. When everyone has finished eating, group members should take turns standing up and describing what they accomplished. The sticky notes can be collected to be displayed back at the office. Sometimes it is a particularly tricky algorithm that a developer finally figured out or an extremely elusive bug that has finally been tracked down and exterminated. It is important to have fun with the process. Sometimes the sillier the accomplishments people offer to the group the better.

Get t-shirts made with the team's "Brand" on it. Define an identity for the group or the product that they are working on. Often a code name for a project can be used. At AmikaNow! Corporation, we were developing products that were to be released in three stages. We code named the three stages, Charmander®, Charmeleon® and Charizard® after three Pokemon® monsters who represented

an evolutionary sequence. Pokemon® was a popular video game in the late 1990's. As each product was released we had team T-shirts made with the appropriate creature emblazoned on the front. It did not take long for Pokemon(tm) paraphernalia to start appearing on the desktops of group members.

Celebrate personal milestones as well as group milestones. Sign cards on each group members Birthday and share a birthday cake at lunch. Acknowledge a group member's anniversary with a card or a team lunch.

Resulting Context:

The group members gain the feeling that their accomplishments and they themselves matter even though the spot light is no longer focussed on their group. The team morale has improved and group members look forward to celebrating their next success.

Rationale:

People need to feel appreciated. When they do not get the appreciation that they feel they deserve they feel neglected. This leads to lowered morale and productivity. By doing simple things within the group such as acknowledging personal and professional milestones, people get the message that they matter. It is very important to maintain that feeling throughout a person's membership within the group.

Known Uses:

The Team Handbook (Scholtes 2001) advocates celebrating successes as does *Joining Together: Group Theory and Group Skills* (Johnson 1997).

This includes the personal experience of the author and her colleagues, with respect to creating a brand for the team and celebrating successes at two local companies (AmikaNow! Corporation and Mitel Networks).

A-2 -- Oil & Lube

Problem:

The group has been performing so well that everyone has forgotten about ongoing issues with group dynamics. Some communication problems may have cropped up or other small problems may be reasserting themselves.

Context:

Groups go through normal cycles of high and lows, even within each stage of group development. This is most obvious within the Performing stage. Everything is going along so well and all group members are working in concert with each other. Problems go unnoticed until they are big problems. When small problems are identified, they often seem larger and more important than they really are.

Forces:

The group members are working well together. They may be feeling so comfortable with each other that they are not as courteous and professional as they should be. Group members are content in their roles, but have problems when those roles inevitably change.

Solution:

It is time for an oil change and a lube job. Groups, like vehicles, require regular maintenance. Group members must continually assess personal and group performance. They must make adjustments to accommodate changing requirements and schedules.

The group must schedule regular cycles of review of both their group dynamics and processes. Everyone in the group must provide input as to the current state of the group, both the good and the bad. Problems that are identified might arise from changes in the group goals. If this is the case, a new mission statement should be created. If there are problems within the group dynamics they should be addressed with the appropriate solution. In this way the group creates an improvement plan. This can be as simple as choosing the highest priority two or three issues to address before the next review, and planning how to fix these problems. Or, it can be as complex as the Joiner™ 7 Step Method advocated by the authors of *The Team Handbook* (Scholtes 2001).

Example:

In *The Team Handbook* (Scholtes 2001) the authors emphasise the importance of continuing to communicate! It takes ongoing effort to keep team on track. Group members must continually offer support and engage in activities that boost moral. Conflict resolution is an ongoing task. When these issues have been neglected direct action is required.

Schedule a review meeting, use a process to review and improve the group's situation. Remember we do not want any un-repaired broken windows, which would lead to the downfall of the group. In *The Team Handbook* (Scholtes 2001) the authors suggest implementing an Improvement Plan." This is a seven step process that covers defining the problem, evaluating the current situation, performing a cause analysis, brainstorm potential solutions, compile the results and implement changes that will help ensure that the same problem is not encountered again. The seven step plan is a way of not just fixing a problem, but really finding out what the cause was and how it can be prevented in the future.

Resulting Context:

The group dynamics are back on track and the group is communicating and interacting well again. The issues have been resolved and the group members have been reassured that their group matters and will get help when it needs it.

Rationale:

When everything is going well, we as human beings, tend to relax. When we loosen our vigilance regarding group interactions, problems will soon re-appear. Building a group is like building any other relationship between human beings. It takes continuous work and vigilance for the relationships to remain healthy. By evaluating what went wrong to allow a negative situation to occur, we can help prevent that from happening again in the future.

Known Uses:

The Team Handbook (Scholtes 2001) has developed a seven step method around the concept of an "Improvement Plan." Throughout their research they clearly state that creating and maintaining teams takes ongoing work. It is important for the team to have a plan as to how to deal with group maintenance.

In *Joining Together: Group Theory and Group Skills* (Johnson 1997) the authors also take the stand that maintaining group relationships takes work. They propose a number of quizzes and group activities that help group members get to know themselves and their group mates better. By providing an ongoing process for group development problems can be anticipated and addressed before they become issues for the group.

PERFORMING – TRANSITION B: ADJUSTING TO ADJOURNING/REFORMING

(Transition 2.6 in Figure 4.5 - State Transition Diagram on page 73)

This section contains the patterns that are solutions to some of the problems or gating issues that groups face when they are in the Performing stage and must prepare to enter into the next stage of group development, Adjourning or Reforming. This occurs when one or more group members leave the group.

TRANSITIONAL CONTEXT:

When a group has been functioning at the performing stage for a while, they have built strong bonds between the group members. In Highly Effective Groups, this emotional bond is the distinguishing factor. When a group is going through the transition to adjourning or reforming, issues arising from these emotional bonds must be addressed. Group members must engage in activities to reduce their emotional dependencies on fellow group members. They must engage in activities terminating their role and technical responsibilities. Subgroups may need to prepare to reform into a new group. The group as a whole must find Closure and prepare to move on. Group members may mourn for the loss of their friends and the bond that they have shared. The group may need to deal with separation issues. In preparation for the separation, arguments may break out; it is easier to say goodbye to someone that you are mad at than to someone you share a bond with.

B-1 -- Separation Issues

Problem:

The group's project has come to an end and the group is being disbanded. The group members are having difficulty letting go of the strong relationships that they built with each other in order to become a Highly Effective Group. There is increased tension among the group members.

Context:

The group has been working very well together. So well in fact that the project has been completed and now it is time to move onto new challenges. They have been spending a great deal of time with each other and may feel as close as family. They are not looking forward to saying good bye to people they have forged such strong bonds with.

Forces:

The group members all feel very strongly about each other. The group members have forged very strong bonds of positive interdependence and friendship. They know that they each must go their separate ways to their new groups. When preparing to separate people will often bicker and fight. It is easier to separate from someone that you are angry with. They are feeling frustrated anxious and snippy with each other. They may be starting to look forward to their new positions and starting to identify with their new group. Some group members may have feelings that resemble mourning and may go through the stages of mourning.

Solution:

It is important to acknowledge the way the group members are feeling and help them to understand their feelings. The group must engage in activities to reduce their dependencies on fellow group members. They must acknowledge this process of separation and allow it to happen, while minimizing the strife that can occur through talking with each other.

It often helps to plan to meet socially on a regular basis over the next few months. The group may be disbanding but the friendships do not have to. Be sure to exchange contact information so that all group members can stay in touch.

Example:

When a single group member is leaving the group, either voluntarily or through a layoff, it is helpful for the team to take them to a farewell lunch. This is a way of recognising the bonds of friendship and appreciation for the leaving team mate.

When the entire team is being disbanded or regrouped into several new teams it is important to mark the success of the group. A group photo or group T-shirts are a good way to mark the existence of a group that will no longer exist.

Resulting Context:

The group members will still feel sad and mourn the loss of their friends but they will understand what is happening and why they are feeling the way that they are. This understanding will stop the snappiness and anger that some group members may be feeling.

Rationale:

Whenever there is an emotional bond there will be anxiety at the thought of breaking that bond. By recognising this fact and talking about the way the group members are feeling, the anxiety is lessened. They realize that this is a normal stage that people go through and that the relationships that they have forged within this group can be the basis for ongoing friendships. When the group members know that it is not really an end but a new beginning they do not feel quite so anxious. Knowing why you are feeling an irrational anger at a person helps you stop and think before you react in a hurtful manner.

Known Uses:

Separation issues are a well known concept in the realm of group theory. Separation issues arise often for military families. When one member has to be away from the family for periods of time, separation issues become a reality that must be dealt with. Community Resource Centres on military bases council for

separation issues and provide workshops and seminars so that members can understand why they are feeling and acting the way they do.

Social Services also deal with separation issues when taking a child into care and placing them with foster parents. Foster Parents are trained to recognise the behaviours and help the child move through this stage.

Class reunions are common. They provide a way in which classmates who have worked together closely and bonded are able to keep in touch. Alumni activities are another way of keeping the bonds of friendship strong.

B-2 -- Post Mortem

Problem:

The valuable lessons, both technical and organizational, learned within this group are at risk of being forgotten.

Context:

The group is disbanding or the project has run to completion. The group members are all focusing on finishing up the last details of the project. They may not have realized nor had time to recognise that this is then end of the project.

Forces:

The group members are completing the remaining tasks and shifting their focus to the next project. It may be difficult to get them to focus on analysing their group experience. There is a great deal of benefit to both the individuals and the organization, in analysing the group's experience.

Solution:

Complete a no-blame post mortem. The goal of this process is to celebrate the successes and learn from the mistakes.

General questions and sections on the questionnaire should include:

- Invitation to evaluate the group processes. What did we do well at, what could have been better and how?
- Evaluate the product – Did we accomplish what we set out to?
- Ask questions about specific components of the project.
- Evaluate interactions between our group and other groups within the company.
- Document the group's improvement.
- Any other questions relevant to your project

Distribute the questionnaire and schedule a meeting to discuss the results. Allow anonymous submissions of the questionnaire. Include questions specific to the group's experiences as well as general questions relating to the group members interactions and the group's dynamics. When preparing the presentation paraphrase and generalize all comments. Focus on trends and constructive comments made by group members. Try not to leave any comments out.

Example:

The following list includes some common questions from post mortems that the author has personally participated in.

1. What were your own individual role and goals?

- Describe how you measured these goals.
- Analyse whether or not you achieved your goals, and why.

Suggested format:

Goal:

Measure:

Analysis: Present your rationale for whether or not the goal was met.

2. Lessons learned:

- a. What did you like/dislike about the project?
- b. What did you learn about the technology, group dynamics and yourself by participating in this project?
- c. What would you have done differently? What will you remember to do on the next project you work on after this experience?

The resulting questionnaire would look something like the following:

1. What were your own individual role and goals?

My role was that of a developer on the Business Designer project.

Goal: Complete the properties framework in time for release 1.2.

Measure: Percentage functionality complete by original release date.

98% completion

Analysis: The original goal was met, but late additions to requirements led to a slip in the schedule.

2. Lessons learned:

a. What did you like/dislike about the project?

I liked the work environment and Java tools that were provided. Resources were available when required.

I disliked the propensity for the project management team to allow the addition of features late in the development cycle.

b. What did you learn about the technology, group dynamics and yourself by participating in this project?

I increased my knowledge of Swing, architecture and windows development.

I learned that everyone needs to be active participants in the health of the group and that individuals can affect the health of the group. I learned that I know more about how to improve group dynamics within my team.

c. What would you have done differently? What will you remember to do on the next project you work on after this experience?

I would have addressed interpersonal issues more quickly, instead of "giving them time to work out on their own." I've learned to fix any broken windows I find whether it be regarding group dynamics or software. If I cannot fix it right away I comment what the problem is and what I think the solution should be.

Resulting Context:

The group members have been given a forum to supply feedback. Their input will improve the next group situation and potentially processes within the company. By holding a post mortem of the project the group members will be given an opportunity for closure. It signals the end of the project and provides a mechanism to express nostalgia over the history of the project, while taking the best experiences with them.

Rationale:

If the group members do not explicitly analyse the good and the bad about this group experience they may be doomed to repeat the same mistakes in their next group. It is important to learn from this group experience and take that knowledge to the next group, so that the same mistakes are not repeated.

People almost always have comments on the processes used within our groups. They do not often come forward with comments or suggestions unless they are solicited. Groups' members need an indication of closure; that the project is over. A post mortem provides a mechanism to solicit that feedback and provide closure at the same time.

Known Uses:

The IEEE Computer Society advocates the use of a post mortem meeting in its process for designing software development courses. (<http://www.computer.org/software/homepage/2002/05hil/>)

Most group centered courses in University have a post mortem component. A quick search on the web shows the use of post mortems in team projects in many areas including: software development, academia, medical, corporate, etc.

The Introduction to the Team Software Process (Humphrey 2000) and *The Personal Software Process* (Humphrey 1996), advocates the use of a post mortem in order to learn from the experiences of the individual and the group.

The author's personal experience in both an academic and a professional capacity at several companies is the post mortem analysis leads to improvement in the group dynamics.

5 - Discussion and Decisions

In order to find a solution to the whole problem, we had to first find a way to disseminate the information derived from group theory. We did this by writing those solutions in the form of patterns; using both existing patterns and patterns derived from the solutions found in group theory and teamwork resources.

Once we had a collection of patterns or a pattern language we had to find a way to organize them such that the solutions are easier to find. We used divide-and-conquer by partitioning the domain of solutions using the Sequential Stage Theory (Tuckman 1977), which Tuckman introduced. Each stage in the Sequential Stage Theory (Tuckman 1977) is mapped to a state in the resulting state machine. The triggers on the transition to the next state are the resolutions of all the gating issues that the group is facing during the associated stage of development.

Mapping each stage in the Sequential Stage Theory (Tuckman 1977) to a state in a state machine partitions the domain of solutions cleanly. By providing this framework with clear partitions we can identify what the gating issues are at each state/stage. Gating issues are problems or challenges that the group must overcome before they can progress in their development. All these gating issues must be resolved before the group can move to the next stage of development

and state in the state machine. It is important to note that most groups resolve some or all of these gating issues without specific conscious action on the part of group members. For those groups that are unable to resolve gating issues on their own, the contributions of this thesis will be very beneficial. These gating issues will be different at each stage and provide the basis for the rapid identification of pattern(s) that are most likely to provide a solution to the problem at hand. Each pattern will address a gating issue specific to the stage of development that it is associated with.

The stages of the Sequential Stage Theory (Tuckman 1977) also provide a clear understanding of the context within which the group is functioning, at each stage of its development. This context can be described as a list of symptomatic behaviours that are most likely to be exhibited by individuals within the group.

Dr. Glyn Bissix and his class of Group Theory students, at Acadia University, reviewed an early draft of the pattern language resulting from this thesis. Their feedback, through a survey, has led to the concept of a meta-context. The early draft followed the canonical format exactly and this resulted in the contextual information being repeated in every pattern representing a solution to a gating issue on the same transition. They reported that it was difficult to understand because of the repetition. They were absolutely right. As a result of their comments, the People Patterns are presented using the Canonical Format, and organized into stages which define a shared or meta-context. The added benefit is that the stage itself is fully defined and described and this results in a user being able to more-readily identify what stage their software development group is currently in.

There are two main benefits derived from presenting People Patterns within this organizational framework, using meta-context at each stage.

1. The context of all the patterns found in a specific stage/state will share a great many aspects. We define the common state context at the level of the stage and allow each pattern in that stage to inherit this context in addition to any modifying context that may be specific to the solution. This results in the patterns within the language not unnecessarily repeating state information. If we repeated the inherited state information in each pattern it would result in the language as a whole being onerous and repetitive. It is understood that each pattern inherits some state (meta-context information) from the stage in which they are defined. The user can refer to this information once and then apply it to all patterns within the stage they are interested in. The point of this solution is to make it easier for people to access and understand these solutions.
2. By organizing the meta-context information we provide a clear description of the context in which the group is currently operating. This works as a diagnostic tool, in that the user can scan this information to identify which stage their software development group is currently operating within.

This organizational paradigm differs from that used in most current research. For instance, the area of communication is very broad and is often handled as a separate topic. Organizing the information in this manner makes it difficult to quickly find a solution to a communication issue that is specific to a group that is in the norming stage. In fact it can often lead to choosing an incorrect solution for that stage of group development. In the Norming Stage, group members are just getting to know one another; they do not know each other's communication styles and are on their best behaviour as the group defines the group's rules and individual roles. Therefore the communication style, group members choose, needs to be more diplomatic at this stage. The style may change as the group

members become more familiar with each other and grow into the subsequent stages of development. Therefore the solution of how to communicate in a specific circumstance differs depending on what state of maturity the group is currently operating within.

6 - Summary of Approach

This Chapter has described the approach used to present the solutions to the problems group face during their development and the evolution of the organizational framework within which the solutions were presented.

- The solutions to the problems that software developers face when working together in groups were presented in the form of People Patterns.
- The organizational framework is in the form of a state machine modelled on the Sequential Stage Theory (Tuckman 1977). Each People Pattern inherits some context from the stage in which the problem it solves is found. Each People Pattern is located in the framework within the stage described by its context, and associated with the transition that will lead the group members to the next optimal stage in their development.

The process used to develop the deliverables of this thesis (People Patterns and their Organizational Framework) used techniques rooted in computer science. Inheritance, divide-and-conquer, partitioning and state machines are basic tools understood by all software developers. The solutions are all derived from the domain of group theory as it is experienced by software development groups. The

resulting pattern language provides a clear format or tool that allows people to access a solution, to a problem that their group may be encountering, in a contextually relevant and timely manner.

IV - RESULTS

The context within which people interact in groups is, for the most part, implicit. The forces that act upon the group and the group members reactions to those forces within a certain context have been both well documented and reported in social science research after careful scientific observation. From this body of research arises the problems that are known to plague groups during each stage of their development. Social Scientists have applied the scientific method to find repeatable solutions to these problems.

The contributions of this thesis include these well known and independently verified solutions, rewritten into the form of People Patterns. The People Patterns are then organized into a framework based on a very well known and respected theory called the Sequential Stage Theory (Tuckman 1977). Both of these contributions have been independently verified by the social scientists and by their use in existing organizational and Process Patterns.

The unique contribution of this thesis includes the organizational framework that models the Sequential Stage Theory (Tuckman 1977) as a state machine. Another unique contribution is the criteria for defining and identifying People Patterns and rewriting existing solutions into a consistent format. In addition, the definition of meta-context at each stage/state and the overall presentation of the data in a form that is organized, consistent and therefore easier to use and extend is a truly unique contribution. These unique contributions are all based on independently verified and well known concepts for data management and modeling used within the realm of computer science.

While all of the component parts of this thesis' contribution are independently validated within their respective discipline (social or computer science), the combination of these solutions, theories and design constructs is a unique contribution that provides a benefit to software development teams and has not, to my knowledge, ever been proposed in public research. This sentiment was echoed by the impressively knowledgeable participants in the JavaOne™ 2002 Birds of a Feather, which was presented on the pattern language described in this thesis.

Patterns are by their definition known solutions to a problem. They state several known uses in their descriptions, which validates them as solutions to the described problem. The solutions chosen as People Patterns in this thesis are all solutions, taken or adapted from current research in group dynamics, team work as well as process and Organizational Patterns. The validity of the patterns themselves has already been independently determined before they were adapted to be People Patterns. For instance, the following patterns were derived from and validated for the following sources:

<u>Pattern Name</u>	<u>Known Uses</u>
Getting to Know You	<p>"Group Goals and Social Interdependence" in <i>Joining Together: Group Theory and Group Skills</i> (Johnson 1997)</p> <p>Repeated use in personal experience</p>

Go for the Goal	<p>Birds of a Feather held at JavaOne 2002</p> <p><i>The Team Handbook</i> (Scholtes 2001)</p> <p><i>Joining Together: Group Theory and Group Skills</i> (Johnson 1997)</p>
Our Team	<p>“The Cat Ate My Source Code.” And “Software Entropy” in <i>The Pragmatic Programmer</i> (Hunt 2000)</p>
Value Your Neighbour	<p><i>Values Individuals</i> [AUE 1998]</p> <p>(pg 517) <i>Joining Together: Group Theory and Group Skills</i> (Johnson 1997)</p>
Resolving Conflict	<p><i>The Team Handbook</i>(Scholtes et.al. 2001)</p> <p>“Self Selecting Team” [COP1]</p> <p><i>The Non-violent Communication Model</i>, published by Marshall Rosenberg [ROS]</p> <p>Repeated use in personal experience</p>
Harnessing Competitive Behaviour	<p>“Overbearing Participant” from <i>The Team Handbook</i> (Scholtes 2001)</p> <p>Repeated use in personal experience</p>

Roles, Right & Responsibilities	<p><i>The Mythical Man Month</i> (Brooks 1995)</p> <p><i>The Team Handbook</i> (Scholtes 2001)</p> <p><i>Joining Together: Group Theory and Group Skills</i> (Johnson 1997)</p>
Only Productive Meetings	<p>“Check In” in <i>Software for your Head: Core Protocols for Creating and Maintaining Shared Vision</i> (McCarthy 2002)</p> <p><i>The Team Handbook</i> (Scholtes 2001)</p>
Creative Communication	<p>“Communicate!” in <i>The Pragmatic Programmer</i> (Hunt 2000)</p> <p>“Office Protocol (pg 29) in <i>The Peopleware Papers</i> (Constantine 2001)</p> <p><i>Joining Together: Group Theory and Group Skills</i> (Johnson 1997)</p>
Celebrate Success	<p><i>Joining Together: Group Theory and Group Skills</i> (Johnson 1997)</p> <p><i>The Team Handbook</i> (Scholtes 2001)</p> <p>Repeated use in personal experience</p>

Oil & Lube	<i>Joining Together: Group Theory and Group Skills</i> (Johnson 1997) <i>The Team Handbook</i> (Scholtes 2001)
Separation Issues	Military Family Community Resource Centers Social Services Foster Care Class Reunions Repeated use in personal experience
Post Mortem	<i>The Team Handbook</i> (Scholtes 2001) IEEE Computer Society <i>Introduction to the Team Software Process</i> (Humphrey 2000) Repeated use in personal experience

The framework within which the patterns are placed is based upon a very well known and widely accepted theory called The Sequential Stage Theory (Tuckman 1977). The meta-data shared by all patterns within a given state is defined by this theory. In *The Team Handbook* (Scholtes 2001), the authors describe the Sequential Stage Theory (Tuckman 1977) and use it to present an understanding of what group members go through as the groups develop as well as presenting solutions

to some problems that groups are known to face during each stage of development.

This thesis takes these concepts, which have been independently validated and merges them into a multi-disciplinary solution to the problem of how to present these solutions to software development group members, using well known and validated computer science design techniques. All the component parts of the product of this thesis are each independently validated, it follows that the sum of those parts is itself validated.

V - CONCLUSION

1 - Summary

The reader has been introduced to the concepts and research behind the Group Theory used in the approach. This thesis has introduced Patterns, Software Patterns, Pattern Languages and Organizational Patterns. A multidisciplinary approach has been presented that uses divide-and-conquer, state machines, inheritance and patterns to partition and present the solutions to the problem groups face during their development, in an organized and useable fashion.

In the introduction we stated that the main problem is that there is currently no clear format or tool that allows people to access a solution, to a problem that their group may be encountering, in a contextually relevant and timely manner.

We have accomplished that by achieving the twofold goal. The first goal was to identify and present the solutions to the problems encountered by groups in their quest to mature into effective groups as a candidate pattern language of People Patterns. The second goal was to provide an organizational framework for this new pattern language that will allow us to partition the solutions to the problems encountered by software groups and present the solutions in a manner that will allow people to more readily identify their problem, and find a solution.

These goals were in turn both achieved by creating and presenting the contribution of this thesis. The contribution of thirteen patterns presented in the pattern language "Growing and Maintaining Highly Effective Groups," provide a tool that allows people to access solutions, to problems that their software development group may be encountering, in a contextually relevant and timely manner. The solutions presented as candidate patterns have been identified and

presented as solutions to the problems encountered by groups in their quest to mature into highly effective groups.

The organizational framework containing the new pattern language allows us to partition the solutions to the problems encountered by software groups and present the solutions in a manner that allow people to more readily identify their problem, and find a solution. This structure provides a single inheritance mechanism allowing the patterns at a specific stage to inherit context and forces from their encompassing stage. This enables the patterns themselves to contain only relevant contextual information rather than repeating contextual information that is actually inherited from the meta-context described in the state.

2 - Future Work

The organizational structure presented in this thesis provides a repository for the mining of many additional patterns. There remain many solutions that did not appear in this thesis, which are as relevant and as useful as the thirteen that were presented. The true power of this approach is the fact that it can grow with the addition of solutions that will provide more benefit to software development group members. The more solutions and variations on solutions that are accumulated and made available from one source, the better and more useful the pattern language will become.

Some of the references and patterns in this thesis were derived from works published to the WikiWeb and Portland Pattern Repository. Both of these formats have been designed to provide a platform from which to disseminate patterns and pattern languages, such as the one introduced in this thesis. It would be beneficial to make this pattern language available through such a method,

which would encourage feedback and refinement of the patterns as well as the addition of new patterns.

While the thought of publishing the resulting pattern language is intriguing, a traditional book format is not the best mechanism for disseminating this information. The key to the just-in-time learning of this material is a fast and logical mechanism for determining which stage the group is in and what category of problem we are dealing with. A traditional textual format is neither fast nor logical with respect to a reader finding specific information, as we have seen with virtually all the research mentioned in this thesis. We need a format that is more conducive to our needs, like the children's book series "Choose your Own Adventure." A format where the reader is led through a series of questions and makes decisions or choices that leads them to the best solution would be a better way of presenting this material.

As this pattern language grows it would be a good candidate for implementation as an Expert System. A web interface would be useful to allow a group member, who is struggling with a problem within their group, to answer a few key questions about their group's behaviour. These questions would be designed to diagnose the placement of the group within a specific stage of the Sequential Stage Theory (Tuckman 1977). Once the stage had been identified the user would be presented with a category of problem. For instance, does the problem involve issues concerning interpersonal communication, adhering to group norms, inability to align personal goals with group goals or emotional detachment from the group? The user would then be presented with a group of problem statements from the available patterns. They could then choose the solutions that seemed most appropriate to their problem in a timely and contextually relevant manner.

IBM has a website devoted to Patterns for e-business [IBM 2002]. They have utilized the structure of web links on the internet together with an organizational structure that asks the user's key questions, which helps them determine the correct solution for their situation. A similar structure could be used to present the People Patterns presented in this pattern language.

APPENDIX 1 - GLOSSARY

Term	Meaning
Acceptance	The communication of high regard toward others and their contributions to the group's work.
Aggregate	Collections of individuals who do not interact with one another.
Balanced Participation	All members of the group contribute to the group's product more or less equally.
Cohesiveness	All the forces (both positive and negative) that cause individuals to maintain their membership in specific groups. These include attraction to other group members and a close match between individuals' needs and the goals and activities of the group. The attractiveness that a group has for its members and that the members have for one another.
Communication	A message sent by a person to a receiver (or receivers) with the conscious intent of affecting the receiver's behavior.
Conflict of interest	When the actions of one person attempting to maximize his or her needs and benefits prevent, block, interfere with, injure, or in some way make less effective the actions of another person attempting to maximize his or her needs and benefits.
Consensus	A collective opinion arrived at by a group of individuals working together under conditions that permit communications to be sufficiently open and the group climate to be sufficiently supportive for everyone in the group to feel that he or she has had a fair chance to influence the decision.
Differentiation by Roles	Using roles to help members identify their place within the group separate and apart from each other.
Effective group	A group whose members commit themselves to the common purpose of maximizing their own and each other's success.
Evaluation apprehension	Concern over being evaluated by others. Such concern may increase arousal and may play an important role in social facilitation.
Goal	A desired place toward which people are working; a state of affairs that people value.
Goal structure	The type of social interdependence specified among individuals as they strive to achieve their goals.

Term	Meaning
Group	Two or more individuals in face-to-face interaction, each aware of his or her membership in the group, and each aware of their positive interdependence as they strive to achieve mutual goals.
Group effectiveness	Success by the group in (1) achieving its goals, (2) maintaining good working relationships among members, and (3) developing and adapting to changing conditions to improve its ability to achieve (1) and (2).
Group dynamics	The area of social science that focuses on advancing knowledge about the nature of group life. The scientific study of behavior in groups to advance our knowledge about the nature of groups, group development, and the interrelations between groups and individuals, other groups, and larger entities.
Group goal	A future state of affairs desired by enough members of a group to motivate the group to work towards its achievement.
Group Processing	Monitor performance and process feedback, both formal and informal
Hidden Agendas	Personal goals that are unknown to all the other group members and are at cross-purposes with the dominant group goals.
Individual Accountability	Assessing the quality and quantity of each member's contributions and giving the results to all group members.
Leadership	The process through which leaders exert their influence on other group members.
Norms	The rules or expectations that specify appropriate behavior in the group; the standards by which group members regulate their actions.
Positive interdependence	The perception that one is linked with others in such a way that success is impossible without them (and vice versa) and that group effort must be coordinated in order to complete a task.
Promotive Interaction	When individuals encourage and facilitate each other's efforts to complete tasks and achieve in order to reach the group's goals.
Role	A set of expectations defining appropriate behaviours associated with a position within a group. The part played by a member of a group. Rules or understandings about the tasks persons occupying certain positions within a group are expected to perform.
Trusting behavior	Openness and sharing.
Trustworthy behavior	Expressing acceptance, support, and cooperative intentions.

Table 6 - Glossary of Group Development Terms

All Group Theory definitions are derived from Johnson & Johnson, (1997, pp. 561-571)

BIBLIOGRAPHY

Alexander, Christopher (1977). *A Pattern Language*. New York: Oxford University Press.

Alexander, Christopher (1979). *A Timeless Way of Building*. New York: Oxford University Press.

Ambler, S. (2002). *Agile Modelling, Effective Practices for Extreme Programming and the Unified Process*. New York: John Wiley & Sons, Inc.

Bales, R. (1965). The equilibrium problem in small groups. In A. Hare, E. Borgatta, & R. Bales (Eds.), *Small groups: Studies in social interaction* (pp. 444-476). New York: Knopf.

Brooks, F. P. (1995). *The Mythical Man Month*. Anniversary Edition. MA: Addison-Wesley.

Brown W.J., Malveau R. C., McCormick III H.W., Mowbray T. J., (1998) *Anti Patterns Refactoring Software, Architectures, and Projects in Crisis*, New York: John Wiley & Sons, Inc.

Constantine, L. (2001). *The Peopleware Papers. Notes on the Human Side of Software*. NJ: Yourdon Press.

Cooper, A. (1999). *The Inmates are Running the Asylum*. Indianapolis, IA: SAMS

Coplien (1994). *A Development Process Generative Pattern Language*. PLoP

Gamma, Erich, Helm, Richard, Johnson, Ralph, Vlissides, John.(1994). *Design Patterns*. MA: Addison-Wesley.

Highsmith, J.(2002). *Agile Software Development Ecosystems*. MA: Addison-Wesley.

Hill, W., & Gruner, L. (1973). A Study of development in open and closed groups. *Small Group Behaviour*, 4, 355-381.

Humphrey, Watts S., (2000). *Introduction to the Team Software Process*. MA: Addison Wesley.

Humphrey, Watts S., (1996). *Introduction to the Personal Software Process*. MA: Addison Wesley.

Hunt, A., & Thomas, D. (2000). *The Pragmatic Programmer*. MA: Addison-Wesley.

- Johnson, D.W., & Johnson, F. P. (1997). *Joining Together: Group Theory and Group Skills*. Sixth Edition. M.A.: Allyn and Bacon.
- Martin, Robert C.; Riehle, Dirk; Buschmann, Frank. (1998) *Pattern Languages of Program Design 3* Reading, MA: Addison-Wesley.
- McBreen, P. (2002). *Software Craftsmanship. The New Imperative*. MA: Addison-Wesley.
- McCarthy, J. & McCarthy, M (2002). *Software for Your Head. Core Protocols for Creating and Maintaining Shared Vision*. Reading, MA: Addison-Wesley.
- Metzger, P. W. (1987). *Managing Programming People. A Personal View*. NJ: Prentice-Hall Inc.
- Moreland, R., & Levine, J. (1982). Socialization in small groups: Temporal changes in individual-group relations. In L. Berkowitz (Ed.), *Advances in experimental social psychology* (vol. 15, pp. 137-192). New York: Academic Press.
- Moreland, R., & Levine, J. (1988). Group Dynamics over time: Development and socialization in small groups. In J. McGrath (Ed.), *The Social Psychology of Time* (pp. 151-181). CA: Sage.
- Olson, D. S., & Stimmel, C. L. (2002). *The Manager Pool. Patterns for Radical Leadership*. MA: Addison-Wesley.
- Rising, L. ed. (1998). *The Patterns Handbook. Techniques Strategies and Applications*. Cambridge UK: Cambridge University Press
- Shambaugh, P. (1978). The development of the small group. *Human Relations*, 31, (pp. 283-295).
- Sholtes P. R., Joiner B. L., & Striebel B. J. (2001). *The Team Handbook*. Second Edition. WI: Oriel Inc.
- Tuckman, B. W. (1965). Developmental Sequence in Small Groups. *Psychological Bulletin*, 63, (pp.384-399).
- Tuckman, B. W., & Jensen, M. A. C. (1977). Stages of small group development revisited. *Group and Organizational Studies*, 2, (pp. 419-427).
- Warfield, J. N. (1976). *Social Systems: Planning, Policy, and Complexity* (pp. 78-88). New York: Wiley

WEB REFERENCES

- [BRO 1998] Brown W. J. et.al. (1998) *The Software Patterns Criteria*
<http://www.antipatterns.com/whatisapattern/>
- [AUE 1998] Auer, Ken. (1998) *Effective Leadership in Software Development*. Fuquay-Varina, NC:
RoleModel Software, Inc.
<http://www.rolemodelsoft.com/patterns/philosophy.htm>
- [AMB 2002] Ambler, Scott. (2002) *The Process Patterns Resource Page*
<http://www.amblysoft.com/processPatternsPage.html>
- [COP1] Coplien, James. *Organizational Structure Patterns*.
<http://www.bell-labs.com/cgi-user/OrgPatterns/OrgPatterns?OrganizationBookPatternTable>
- [COP2] Coplien, James. *Self Selecting Team*.
<http://www.bell-labs.com/cgi-user/OrgPatterns/OrgPatterns?SelfSelectingTeam>
- [FOX 1997] Fox, P. G. (1997). *The Trainer's Role in Building Successful Teams*.
Fox Performance Training.
<http://www.foxperformance.com/consulting5.html>
- [IBM 2002] *IBM Patterns for e-business*.
<http://www-106.ibm.com/developerworks/patterns/>
- [MES 2001] Meszaros, Gerard., Doble, Jim. (2001) *A Pattern Language for Pattern Writing*. Object Systems Group
<http://hillside.net/patterns/writing/patternwritingpaper.htm>
- [PPR-AlexF 2002] *Alexandrian Form*. Portland Pattern Repository. last revised Feb 26, 2002.
<http://c2.com/cgi/wiki?AlexandrianForm>
- [PPR-BeckF 2002] *Beck Form*. Portland Pattern Repository. last revised Aug 22, 2002.
<http://c2.com/cgi/wiki?BeckForm>
- [PPR-CockF 2001] *Cockburn PM Form*. Portland Pattern Repository. last revised May 30, 2001.
<http://c2.com/cgi/wiki?CockburnPmForm>

[PPR-CompF 2001] *Compact Form*. Portland Pattern Repository. last revised May 30, 2001.

<http://c2.com/cgi/wiki?CompactForm>

[PPR-ConoF 2002] *Canonical Form*. Portland Pattern Repository. last revised Oct 20, 2002.

<http://c2.com/cgi/wiki?CanonicalForm>

[PPR-CoplF 2001] *Coplien Form*. Portland Pattern Repository. last revised May 30, 2001.

<http://c2.com/cgi/wiki?CoplienForm>

[PPR-FowlF 2002] *Fowler Form*. Portland Pattern Repository. last revised Aug 22, 2002.

<http://c2.com/cgi/wiki?FowlerForm>

[PPR-Goff 2001] *Gof Form*. Portland Pattern Repository. last revised May 30, 2001.

<http://c2.com/cgi/wiki?GoFform>

[PPR-PF 2001] *Portland Form*. Portland Pattern Repository. last revised July 27, 2001.

<http://c2.com/cgi/wiki?PortlandForm>

[ROS]Rosenberg, Marshall, *The Nonviolent Communication Model*.

<http://www.nonviolentcommunication.com/BookPages/nvcmodel.html>

[SPA 1996]Spargo, L., & Kelsey, B. (1996). *How Two Universities Crossed the Border*.

Toronto: School of Administration and Information Management. Ryerson Polytechnic University.

http://info.isoc.org/isoc/whatis/conferences/inet/96/proceedings/c8/c8_1.htm