Examining Team Cohesion as an Effect of Software Engineering Methodology

Carol A. Wellington
Department of Computer Science
Shippensburg University
Shippensburg, PA 17257
(717)477-1424
cawell@ship.edu

Thomas Briggs
Department of Computer Science
Shippensburg University
Shippensburg, PA 17257
(717)477-1354
thb@ship.edu

C. Dudley Girard
Department of Computer Science
Shippensburg University
Shippensburg, PA 17257
(717)477-1016
cdgira@ship.edu

ABSTRACT

This paper describes an experiment in which student teams developed products using two different methodologies: the Team Software Process (TSP) as a plan-driven methodology and Extreme Programming (XP) as an agile methodology. We carefully define cohesion and derive instruments appropriate for measuring cohesion. Then, throughout the projects, the teams were surveyed to measure various aspects of team cohesion and those results support conclusions about how methodology was affecting cohesion. The results show that the measures developed lead to interesting observations that can be applied to current, non-academic projects. In addition, future work in broadening this study is justified.

Categories and Subject Descriptors

K.6.3 [**Software Management**]: Software Process and Software Development.

General Terms

Management, Measurement.

Keywords

Team cohesion, Extreme Programming, Team Software Process.

1. INTRODUCTION

The "ecosystem"[6] of a software development project, including things like room layouts, people issues, and communications channels directly affect the methodology related decisions the develop team makes[5]. These decisions can directly affect the quality of the resulting product. For example, issues relating to managing people have been shown to affect architecture decisions[5].

While ecosystem issues affect choice of methodology, we are interested in the reverse affect: how the choice of methodology affects the communications channels in the ecosystem. In particular, we are examining the effect that methodology choice has on team cohesion.

There have been many qualitative claims about agile methodologies' affects on team morale. For example, when Highsmith[7] says, "Agile Software Development helps create a workplace culture that attracts knowledge workers in our Information Age," he is attributing "culture" to development methodology. In academic situations, it has been shown that students enjoy their work more when they are pair programming[13]. In fact, Williams[12] makes the claim that pair programming improved team strength and companionship. We do not disagree with any of these claims, but these are weak claims in that they are predominantly from anecdotal experiences and qualitative.

Prior work has shown that the personalities of team members have an effect on team cohesion within XP[9]. We are studying how the methodology itself affects team cohesion.

The software engineering community has begun to study the effects of agile methodologies. Williams, *et al* have proposed a framework for quantitatively studying the effectiveness of agile methodologies[14]. This framework includes the measurement of "job satisfaction" which is part of the cultural effects of methodology. They acknowledge that formal experiments have external validity, but are difficult to apply to real projects as a "control group" working on the same project with a different methodology is unreasonable in industrial settings. In our academic environment, we have a unique ability to create such controlled experiments. While the fact that our subjects are students in a class must be remembered in the analysis, this controlled experiment will allow us to investigate tools that measure cohesion within the context of different methodologies.

To begin this study, we designed an experiment involving students in two software engineering classes: Traditional Life Cycle (TLC) and Testing and Extreme Programming(XP). In these classes, the students work in teams developing a product for a customer. In the semester in question, we had one team in each class work on the same problem statement with the same customer. The teams consisted of 15 and 16 upper division computer science students, respectively. Throughout the semester, we administered surveys to observe the team cohesion over time.

2. TEAM COHESION

We are interested in cohesion because it is associated with success[4]. Within software engineering, cohesion leads to

increased communication and knowledge sharing[10]. This should lead to improved product quality.

Before embarking on measuring team cohesion, it is useful to carefully define the concepts associated with cohesion. First, a "team" is not just a group of people. It is important that a team is a group of people that share an identity and a purpose (for the rest of this paper, the terms "group" and "team" refer to teams that meet this criteria). "Cohesion" is the degree to which the team sticks together as they pursue the team's purpose. Clearly, software engineering teams meet our criteria and this definition of cohesion has value because it is the degree to which they work together and should reflect some aspect of their ability to succeed.

The amount of cohesion that a team exhibits changes over time. Within a software engineering organization, cohesion can be affected by many things: personality of team members[9], length of time the team has been together, overcoming obstacles together, etc. Therefore, it is important to measure cohesion over time and the changes in cohesion may be more insightful than its actual value.

Within software engineering, cohesion can be thought of in two very different ways: the social attachment within the team and the team's connection to the project itself. Carron, *et al*[4] called these social (S) and task (T) cohesion, respectively. In addition, we can think of these types of cohesion at two levels of granularity: at the individual level and for the group as a whole. Carron, *et al*[4] called these Individual Attractions to the Group (ATG) and Group Integration (GI), respectively. Combining these two dimensions of cohesion results in four measures of aspects of team cohesion:

- GI-T: The group's attachment to the task
- GI-S: The group's social connection
- ATG-T: Individual attachment to the task
- ATG-S: Individual connection to the group

3. MEASURES OF TEAM COHESION

In order to increase our confidence in our results, we have used a number of tools to measure team cohesion. These were administered in the form of two surveys: Group Environment Questionnaire and a survey we developed.

3.1 Group Environment Questionnaire

Carron, *et al*,[3] believe that all four aspects of cohesion can be measured by measuring the group and individual perceptions of cohesion. They developed the Group Environment Questionnaire a survey that measures GI-T, GI-S, ATG-T, and ATG-S. This questionnaire was devised for sports teams and has been shown to highly correlate with perceptions of cohesion in that environment. Because some of the questions were specific to sports teams, we have modified those questions to reflect software engineering teams.

3.2 Peer Rankings

At five milestones in the projects, we administered a survey in which the students ranked their teammates in two ways. First, they ranked each team members affect on the team's success on the following scale:

1: detrimental to project success
2: useless to project success
3: average impact on project success
4: above average impact on project success
5: critical to project success
N/O: No Opinion on effect

Second, they were asked, "If you were picking a team for a new project, which of your teammates would you like to include?" This question required a binary response for each teammate; they were not allowed the "No Opinion" option.

From these questions, we gathered a number of statistics following standard cohesion analysis experiments[1]. We analyzed the averages of the peer rankings in the first question and the number of peers being selected in the second question. These give us a measure of individual perceptions of the degree of "value" of each team member, but not a direct measure of how closely the team is connected.

To get a measure of the similarity in responses, we calculated interrator agreement[1] for ranking made for each team member. First, responses of N/O were given a ranking of zero and, for the i^{th} team member, the standard deviation of the rankings given to that team member was computed (s_i). The maximum value that this statistic can achieve is denoted s_{max}. Interrator agreement for the team member is computed as follows:

$$ira_i = \frac{(s_i - s_{\max})}{s_{\max}}$$

This gives a measure of similarity of response about a particular individual. The range of this statistic is 0 to 1 and increases with the similarity of the rankings. This gives us a strong measure of the closeness of a team and can also be used within subsets of the team to measure their cohesion within the context of the team.

Finally, we looked for mutual relationships within the answers to the binary question. These are pairs of teammates who each selected the other to be on their team. The number of such relationships should give another perspective on the closeness and mutual respect within the team.

3.3 Project Interest

Our survey also asked two questions related to the interest the individual had in pursuing the project:

1. On a scale of 1 to 5, where 1 means "not at all" and 5 means "very much," to what extent do you want to work on the next release of this project?
2. If it didn't affect your grade in this or any of your other classes and if you could pick your team, how many more or less hours would you work on this project a week? Zero means no change in your workload.

We found that the second question was not insightful as the magnitude of the responses depended on the number of hours the students were currently working on the project. As we did not have reliable numbers for hours they were working (students are not good at measuring themselves), we did not include the results of this question in our analysis.

4. Experiment Design

The students in this experiment were enrolled in two classes: TLC and XP. The TLC team consisted of 17 upper division

students and used a variant of the Team Software Process (TSP) [8]. After completing a use case model for the requirements and a high-level architecture, the team was divided into three sub-teams of five or six members each corresponding to the main components of the architecture. The XP team consisted of 16 upper division students using most of the twelve XP basic practices[2]. After a two-week "spike" in XP techniques, we completed three four-week iterations. In each iteration, we played the Planning and Iteration Planning Games, the students used test-driven development and pair programming, and the iteration culminated with a presentation to the customer.

The GEQ was administered twice: midway through the project and at the end of the project. Our survey (including peer rankings, peer selection and project interest) was given five times through the project (about once every three weeks). This was deemed sufficient to let us see how cohesion was changing over time without overwhelming the students with the surveys

4.1 Teams

The students in both teams had roughly equivalent academic experience with the exception that some of the member of the XP team had completed the TLC course and therefore had experience in larger projects. Both teams were given the same problem statement and the same customer to control as many variables as possible. It is important to note that the students did not get to choose methodologies. They were enrolled in two different classes and had selected the classes to meet graduation requirements. Therefore, the student's perceptions about the methodologies did not taint methodology selection.

While we understand that these methodologies are typically utilized in very different physical environments, both of our teams met in computer classrooms for two hours two days per week. The XP team was given a wall where they posted the story and task cards for the current iteration and posted a Big Visible Chart with the metric that they were currently tracking. The XP team started each class with a 15 minute stand up meeting and then used pair programming for their development. These activities were periodically interrupted by whole class lab activities as they requested them.

While we discussed roles in the context of both methodologies, roles were not officially assigned in either class. Teams selected individuals to take on roles or individuals accepted those responsibilities unofficially.

4.2 Project Description

Both teams were given the same problem description. (The XP team had a choice between this project and continuing a project from a previous class). They were to design a system to help an architect layout a house. In this system, the architect had to be able to specify rules about the relationships between rooms. As the architect modified/moved the rooms, the application was to highlight the rules that were broken. In addition, an option to have the system automatically modify the rooms to fix the broken rules was suggested.

There were a number of factors that went into selecting this project:

It was large enough to challenge a team of 15 students.

It had a non-trivial GUI to challenge our graphics students.

The data to be stored would require good design to achieve efficiency.

5. Results

5.1 Social Cohesion Measures

The results of the GEQ survey are shown in Figure 1.

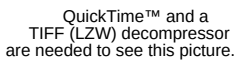

Figure 1 - GEQ Results

In general, these results show very little difference between the individuals' perceptions of the social cohesion of the team (GI-S) and the individuals' connection to the team (ATG-S). Both teams show slight increases in individual connection to the team, but that change is probably not significant.

Table 1 - TLC Ranking Survey Number 1

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
A	4	3	5	3	3	4	5	4										
B	5	5	5	5	5	5	5											
C	4	2	3	3	4	2	5				4			4			4	
D	4	3	4	3	3	3	5											
E	5	5	5	5	5	5	5											
F	4	3	5	3	3	3	5											
G	5	3	5	3	4	3	5											
H			4					4	4	4	4			4			4	
I								4	4	3	3	3	3	4			4	3
J								3	3	3	4	3	3					
K	4		3					4	4	4	5	4	4				4	
L								4	5	5	5	3	5					
M								4	4	4	4	4	3					
N			5		4	4		4	5	3	3			4	4	4	5	4
O							4	4	4	4				4	4	3	5	2
P			4					4		4	4			4	3	4	4	3
Q														5	4	3	4	3
R														4	3	3	5	

Table 1 and

Table 2 show the results of the first and last ranking surveys of the TLC team. Each row of the table is the responses given by a student. The students are ordered by subsystem, so the boxes represent rankings given by students in the same subsystem. Initially, very few rankings are given to students outside of a subsystem. That number does not increase dramatically by the end of the semester, but the increases seen are related to the architecture of the software being developed.

The three subsystems are: rules (students A-G), GUI students (H-M), and structures (students N-R). The GUI layer interacts with

both of the underlying layers, but the interface to the rules subsystem was much simpler than the interface of the structures subsystem. In addition, the structures sub-team considered their software to be a "service" layer and were very interested in determining what functionality would help the GUI layer. The rules subsystem provided an interface that they thought would suit anyone who wanted to use rules. This meant that the rules sub-team had very little interaction with the other sub-teams. By the fifth survey, the GUI and structures teams show much more interaction while the rule subsystem is relatively

isolated. Members of other sub-teams are ranking members of the rule sub-team, but rules members are still not confident enough in the capabilities of other sub-team members to rank them. In fact, the rules people rank fewer outside people at the end of the project than they did at the completion of requirements analysis.

Figure 2 shows the average percent of teammates that each individual chose to rank. The XP team shows a significant increase in this statistic after the first survey with little change from that point in the project. The first survey was given very early in the first iteration of the project, so it is not surprising that the students had not formed opinions about all of their colleagues, yet. By the end of the first iteration, the second point at which they were surveyed, they have opinions about virtually all of their colleagues. In addition, with the exception of a slight decline after the first survey, the average rank they gave their colleagues didn't change throughout the project.

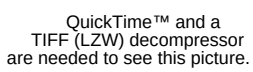

Figure 2 - Percent of Teammates Ranked

The TLC team ranked a much smaller percentage of their colleagues throughout the semester. The colleagues they chose to rank initially came almost entirely from their subsystem. As the semester progressed, the number of colleagues an individual ranked that were outside of their subsystem increased, but, by the end of the semester, they were still unwilling to rank half of the members of the team. This shows that the subsystem organizational structure that resulted from dividing the design according to the architecture has limited the interaction between the engineers.

While the percent of teammates they rank is higher within a subsystem in the TLC team than the XP team, Figure 3 shows that the percent of colleagues they would select for a future team is roughly the same within a TLC subsystem as within the entire team in XP. In fact, a number of the selections made outside of a subsystem were the result of students who selected every teammate for each survey. When those individuals are eliminated, the percent of students outside the subsystem that get selected is less than 10%.

When mutual relationships are studied, in the TLC team, almost all are within a subsystem and almost all pairs within a subsystem exhibit mutual relationships. The XP team has more mutual relationships presumably because the team works without the organizational barriers of subsystems. Since everyone owns and works on all aspects of the project, the number of people that they have worked with enough to be comfortable selecting them is increased.

Table 2 - TLC Ranking Survey Number 5

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
A	4	3	4	3	2	4	4											
B	5	4	5	4	3	4	5											
C	4	3	4	3	1	3	5											
D	4	3	4	3	2	3	4											
E	5	5	5	5	5	5	5											
F	5	4	5	4	3	4	5											
G	4	3	4	3	1	3	4	3										
H			3					4	4	3	4	3		4			4	
I								4	4	3	4	3	3	4			4	
J				3				4	5	3	4	4	3	4	3	3	4	
K	3		3				3	4	4	4	4	4	4	4			4	3
L								4	4	3	3		3					
M								5	5	5	5	5	5					
N			4			3	4	4	4	3	3	3		4	3	3	4	3
O		3	3				4	4	4	3	3		3	4	3	3	5	2
P	4	4	5	4	4	4	5	5	5	5	4	5	4	5	5	5	5	4
Q			4				4	4	4	4	4	3	3	5	4	3		1
R							4		4			4		5	3	4	5	3

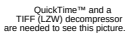

Figure 3 - Percent of Teammates Selected

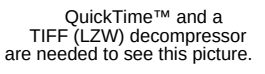

Figure 4 - Number of Mutual Relationships

5.2 Lack of Cohesion

The rankings survey can also be used to find lack of cohesion. In particular, each team had individuals that were given both high and low rankings in a single administration of the survey. For example, one member of the XP team was given ranks of 5 (critical) and 2 (useless) by different teammates in the same administration of the survey. In every case, these individuals became the source of conflict within the team. Those conflicts ranged from mild debates over design decisions to arguments over the general direction the team was taking. The XP class had a second team working on a different project. That team did not have individuals with conflicting rankings and did not have the level of conflict seen in these teams.

5.3 Attachment to the Project

In Figure 1, both teams, at both times in question, scored more highly on the task-related measures of cohesion than on measures of social cohesion, and the TLC shows higher task cohesion than then XP team.

In fact, the TLC team showed an increase in GI-T. This implies that individuals perceived the team's interest in the project as increasing. However, the GEQ measure of individual attachment to the project (ATG-T) in unchanged. Our question that directly asks how much they want to continue working on the project, shows a different picture: Figure 5 shows that individuals on both teams are losing interest as the project continues. The TLC team shows a slight increase in this interest in the fourth administration of the survey. This corresponds with the beginning of the implementation phase of the project and could be a result of the fact that students are more comfortable with implementation than previous phases of the project. It is interesting to note that the XP team showed a significant decrease in interest at the same time. This corresponded with the initiation of the third iteration of their project. The anecdotal explanation for this came from the students. They were now comfortable with the process and were finding the project repetitive. Completing "yet another iteration" was less interesting than starting something new. This was somewhat relieved after the third iteration when the deliverable began to show significant functionality.

QuickTime™ and a
TIFF (LZW) decompressor
are needed to see this picture.

Figure 5 - Interest in Subsequent Release

5.4 Effect on Project Quality

Both teams delivered working prototypes. However, the quality of the code delivered by the XP team was significantly greater than that of the TLC team. While the TLC team took the time to create a well-structured, object-oriented design, they did not implement according to that design. In particular, while their code had a non-trivial class hierarchy, they pushed much of the functionality into the leaves of the hierarchy that resulted in significant amounts of copy-and-paste code. In addition, the GUI subsystem's implementation was very different from their design specification.

The XP team took advantage of multiple iterations and automated tests to refactor their code on multiple occasions. As a result, their source code exhibited generally good object oriented design throughout the project. They attributed this success to the fact that, even though their design had subsystems, they were all aware of all of the code and could refactor across those boundaries. This is a direct result of the cohesion of the team.

6. Conclusions

This study has been a feasibility study into measuring how methodology affects team cohesion. We have developed instruments that measure team cohesion and can be applied to widely varying methodologies. These instruments are easy to administer and the resulting statistics are not difficult to compute or interpret.

Within our study, we can make a number of interesting conclusions. It is interesting to note that the TLC team showed equal or higher scores for every aspect of cohesion at both administrations of the GEQ survey. However, the study indicates significant lack of cohesion across subsystems in TLC, it is likely that those interfaces will have a higher defect rate than code within a subsystem. Organizationally, assigning individuals to coordinate activities between subsystems should increase inter-sub-system cohesion and begin to address this problem. However, our results show that XPs philosophy of everyone owning the source code increases overall team cohesion.

Our results measured the isolation of sub-teams, particularly regarding the rules team in the TLC project. This isolation exhibited itself in an interesting way. When the project was coming to completion, the rules team inferred that the GUI team was not going to complete the user interface to the rules functionality. Without talking to the GUI team, the rules team implemented the required GUI functionality. This led to conflict because it duplicated work that was in progress within the GUI team. This is a clear example of isolation reinforcing itself and resulting in wasted effort.

In addition to measuring cohesion, these instruments can be used to detect pockets of lack of cohesion that can predict potential conflicts, This could allow preemptory management of the situation.

The results demonstrate that cohesion changes over time and can be affected by methodology. In addition, the results show that these techniques can be applied to widely differing methodologies. Applied to non-academic projects, this means that cohesion can be measured and used as an indicator for problems in the methodology.

7. REFERENCES

[1] Arrow, Holly, Stability, Bistability and Instability in Small Group Influence Patterns, *Journal of Personality and Social Psychology, 22, 1,* (1997), 73-78

[2] Beck, Kent, *Extreme Programming Explained – Embracing Change,* Addison-Wesley, 2000.

[3] Carron, Albert V, Brawley, Lawrence R., Widmeyer, Neil W., *G.E.Q. The Group Environment Questionnaire Test Manual* Fitness Information Technology, Inc. 2002.

[4] Carron, Albert V. and Brawley, Lawrence R., Cohesion: Conceptual and Measurement Issues, *Small Group Research, 31, 1,* (2000), 89-106.

[5] Cockburn, Alistair, The Interationc of Social Issues and Software Architecture, *Communications of the ACM, 39, 10* (October 1996), 40-46.

[6] Cockburn, Alistair, *Characterizing People as Non-Linear, First-Order Components in Software Development*, Human and Technology Technical Report, TR 99.05, Oct. 1999, 7691 Dell Rd., Salt Lake City, UT 84121 USA.

[7] Highsmith, Jim, Agile Software Development – Why it's Hot! in *Extreme Programming Perspectives*, edited by Michele Marchesi, *et al*. Addison-Wesley 2003.

[8] Humphrey, Watts S. *Introduction to the Team Software Process,* Addison-Wesley, 2000.

[9] Karn, J. S., and Cowling, A.J., An Initial Observational Study of the Effects of Personality on Software Engineering Teams. In *Proceedings of the 8th International Conference on Empirical Assessment in Software Engineering (EASE 2004)* (Edinburgh, Scotland, 2004). 155-164.

[10] Sommerville, Ian, *Software Engineering 7,* Addison-Wesley, 2004.

[11] Syed-Abdulla, Sharifah, Holcombe, Mike, and Gheorghe, Marian, *Agile Methodology in Practice* Memorandum 03-04, University of Sheffield Computer Science Department, 2004.

[12] Williams, Laurie, Pair Programming: Why Have Two Do the Work of One, in *Extreme Programming Perspectives*, edited by Michele Marchesi, *et al*, Addison Wesley, 2003.

[13] Williams, Laurie and R. Kessler, W. Cugginham, R. Jeffries, Strengthening the Case for Pair-Programming, *IEEE Software*, 2000.

[14] Williams, Laurie, Giancarlo Succi, Milorad Stefanovic, and Michele Marchese, A Metric Quite for Evaluating the Effectiveness of an Agile Methodology, in *Extreme Programming Perspectives*, edited by Michele Marchesi, *et al*, Addison Wesley, 2003.