# Extreme Programming Used to Establish the Culture of a High Performance Team: A Management Case Study

Clement James Goebel III, PMP

Menlo Institute LLC
212 North Fourth Avenue
Ann Arbor, MI 48104 USA

jgoebel@menloinstitute.com
http://www.menloinstitute.com

**Abstract.** Management can often be frustrated by the inability of software development teams to respond quickly to changes in requirements. The standard explanation is that the key developers on the team are too busy with critical path tasks. A more accurate statement would be that the so-called software development team is not actually a team but merely a group of individuals related by an organizational chart. This paper recounts one development organization's transformation from organizational chart to a high performance team using the practices of Extreme Programming.

## 1 High Performance Teams

Simply assigning individuals to the same project, drawing boxes on an organizational chart, or having them report to the same manager does not create a team. A team is defined by the common goal of its members and the interdependence of their efforts.[3]

Members of most software development teams would claim to have a common goal. But this common goal is often secondary to individual productivity, individual goals, and personal accountability. When it is possible for one individual in a group to be succeeding while another individual is failing, a group is not a team. Individuals or subgroups working independently without integrating their work also highlights a group that is not working as a team.

Members of high performance team pools their diverse talents, experiences, and perspectives to produce solutions that are greater than what they would have produced as individuals merely coordinating their individual activities. Extreme Programming, as explained by Kent Beck [1], can be the basis for building a high-performance, team-based culture. This paper is the case study of one organization that changed its culture from one of individual effort and efficiency to one of team effort and team efficiency.

 

### 1.1 Team Performance Analysis

Objective metrics for evaluating a team's performance can be difficult to identify. In Making the Team, Leigh Thompson structures a subjective performance analysis around four criteria: team productivity, team satisfaction, individual growth, and organizational gains.[3] For each of these criteria she provides several specific questions to help assess performance. This paper will use the assessment questions she provides to evaluate the before and after states of a company that adopts the practices of Extreme Programming.

**Team Productivity**

- Does the team have a clear goal?
- What objective performance measures have been established at the outset of teamwork?
- Who are the legitimate clients of the team?
- Does the team's output (e.g., decisions, products, services) meet the standards of those who have to use it?
- Under what conditions should the goal change?
- What sources of information should the team consider to assess whether the initial goal might need to be changed?

**Team Satisfaction**

- Do the team members enjoy working together?
- What conditions could lead to feelings of resentment?
- What conditions could prevent team members from working together in the future?
- How can team members best learn from one another?
- How are team members expected to accommodate to changes, such as additions to the team, growth, and turnover?

**Individual Growth**

- Do the individual team members grow and develop as a result of the team experience?
- Do team members have a chance to improve their skills or affirm themselves?
- What factors and conditions could block personal growth?
- Are individuals' growth needs understood and shared by group members?

**Organizational Gains**

- How does the team benefit the larger organization?

 

- Are the team's goals consistent with those of the larger organization?
- What other groups, departments, and units are affected by the team?
- What steps has the team taken to integrate its activities with those of others?

# 2 Case History

The first section of this case history describes the business background and desires of a company that wanted to dramatically improve its software development practices. The second section outlines the recommended changes suggested by a team of outside consultants. And the third section describes how the initial transition to the new practices was facilitated. (The names in this case have been changed.)

## 2.1 Company Background, The Initial State

Root Computer Products (RCP) is a publicly held corporation in Ann Arbor, Michigan that is going through a dramatic metamorphosis. RCP had successfully found a profitable niche as a producer of computer peripherals that seamlessly integrate with mainframe as well as desktop computer systems. Over time, this success has allowed the company to purchase several other related businesses. The company has also been repackaging the software capabilities that are built into their hardware peripherals as a stand-alone software package and is enjoying initial success in two emerging niche markets. These niche markets both exploit the company's expertise in translating mainframe data formats into formats that can be used by desktop computers connected to the Internet. The customer base initially showing interest in these capabilities consists of large companies that currently use mainframes to print customer statements or bills and would like to email those documents potentially saving printing and postage costs.

The company's transformation from a provider of mainframe computer hardware to an Internet company started at the top of the organization with the hiring of a new CEO. The CEO carefully evaluated the company and the management team. As a result of his efforts, the management team was reconstructed and decided that the company needed to change its focus. Producing hardware peripherals had become a commodity business and is less profitable. Therefore the management team decided to sell off or simply close each business unit involved in those operations and focus on enabling large companies with mainframes to easily produce Internet-ready documents. Going after this emerging software niche dramatically elevated the importance of the software development resources in the company. The software development group has historically focused on adding incremental value to the hardware peripherals; now it will need to provide the company's future products. This transformation effort will be funded using the cash generated by selling off the hardware business units. Therefore the software group is under great pressure to evolve the company's software products quickly to meet customer needs.

 

The company's new product line offers customers three different software packages.

*Package 1.* The core utility that translates data from mainframe formats to Internet formats. This utility is attractive to companies that have strong internal software development teams that can integrate this utility into their own systems.

*Package 2.* A product that integrates the core data translation utility with the industry's strongest third party vendor/partners of software and services.

*Package 3.* An end-to-end product that allows customers to translate documents, load the documents into an electronic warehouse, and then allow customers to view those documents using web browsers. This new product is focused on a particular vertical market and has been designed to be easy for the customer to adopt because it requires almost no support from the customer's Information Technology department.

Each of the three products is sold by different sales forces that specialize in one product's market segment. Each of the three software products also has a dedicated software development team. The team working on the core translation utility has been together the longest and has five members. The members of this group have very specialized roles and almost never deviate from their area of specialty within the utility. There is also a tendency within this group to be focused on the old goals of supporting the hardware that is no longer produced as well as a tendency to revisit old design battles. The members of this team are so specialized that they have formed individual value systems based upon their area of expertise. This lack of a common vision often leaves team members frustrated in their attempts to resolve design conflicts.

The second software development group is a three-member team whose members joined the company as a result of one of the acquisitions. This team's mission is to write software that integrates the core utility with the capabilities of the third party vendor/partners. Each new sale of this product requires significant customization of the product for the specific customer. Because the team has only three members, more than one sale of the product at a given time seriously impairs the team's ability to add new functionality as all of the team's members become focused on customizations.

Team three which is working on the company's premier end-to-end product is the most recently formed team and is the smallest with a staff of only two programmers. The size of the team, however, belies the importance of this product in the company's plans. Because this product leverages features created by the other two teams, this product is able to grow faster than the team size might otherwise indicate. But this team often faces the challenges of working with new technologies and new tools; therefore the team's progress is sometimes inhibited by the need to spend more time learning.

The work environment at RCP is quite pleasant. Each developer has his/her own office with a door. Most of the developers' offices have a window that looks out onto

 

a wooded area where wildlife such as deer can be observed. There are several conference rooms available for meetings and any employee need only call the receptionist in order to reserve one of the rooms. Beverage and snack vending machines are available across the hallway inside of the factory space where the company used to assemble hardware components. The company sponsors several events throughout the year such as volleyball, euchre, and croquet tournaments. And the company has plenty of onsite parking, a significant benefit for an Ann Arbor company.

But even with the advantages RCP has, it still has many challenges to overcome when recruiting additional talent. Like many other software companies, RCP finds it very difficult to integrate new developers into already existing projects. New developers are often asked to read manuals or to help with product testing for their first weeks or months on the job. This often translates into a race to make the new team member productive before they become demoralized. Ironically, when a new employee is quickly assimilated it is often because they are hired at the same time a new project is being started. Long-time team members who have an interest in diversifying their skills may become jealous of the new employee's position because the very specialization that makes the long-term employee valuable also makes it very difficult to backfill that long time employee's current role in order to free them up for new projects.

The fact that the developers have adopted specialized roles creates some efficiency in how quickly each individual developer can write software but creates significant organizational difficulties when one of those specialists goes on vacation. That same specialization also makes it very difficult to move developers from one project to another when market opportunities arise that can only be capitalized upon with additional effort. Because each of the new products includes portions of the other team's work, it is easy to create conflict between the teams when setting their goals. For example, the utility team might need to add features based upon marketing or sales feedback but also have a different set of priorities as defined by its two internal customers, the partner integration team and the web team. In addition to new product features, the customer support team and the quality assurance department identify bugs and add them to the programmers' to do lists. Because the development teams receive these requests from multiple sources, it can be very time-consuming and difficult to sort out the best set of priorities for each team.

Management at RCP also faces several other challenges with software development. Product planning can be hampered by inaccurate completion dates. The marketing team believes that developers often spend days improving aspects of the software that the customers do not care about. The developers believe that the feature requests provided by marketing are often vaguely defined and change while they are being developed. Because of previously missed deadlines the developers have become defensive about estimating completion dates, particularly when asked to estimate a task that they have never previously performed. As each product approaches a release, or ship date it is necessary to reduce the risk of shipping buggy software by slowing, or even stopping, the addition of new features. Because the products have interdependencies this can cause conflict. If the conflicts cannot be resolved by sending emails back and forth, meetings are scheduled with all of the

 

necessary parties. There are also times when certain development tasks just seem to never get completed. Every time that the progress is checked on these tasks there is additional work completed but the task continually remains 90% completed. This can be caused by more work being discovered as more work is completed, the programmer needing to spend significant effort in resolving customer service issues, or it can indicate that an individual programmer is stuck. Programmers have also been known to identify a need in the product and work towards fulfilling that functionality gap, then once the feature is completed they unveil it for management approval.

## 2.2 Analysis of the Initial State

### Team Productivity

- *Does the team have a clear goal?* Vague requirements leave developers feeling unclear about the goal.
- *What objective performance measures have been established at the outset of teamwork?* The only objective measurements currently available are final delivery dates.
- *Who are the legitimate clients of the team?* Currently the developers are assigned to teams by management, provided with a direction by the marketing team, and are often responsible for making significant assumptions about what the end user might need.
- *Does the team's output (e.g., decisions, products, services) meet the standards of those who have to use it?* It should be noted that the case does not address this issue directly. In reality this is because the organization has no method for assessing user satisfaction other than measuring customer support calls.
- *Under what conditions should the goal change?* There is currently no formal process defined to support the goal changes requested by marketing that leaves the developers feeling that goal changes are arbitrary and capricious.
- *What sources of information should the team consider to assess whether the initial goal might need to be changed?* Again, no such process exists.

### Team Satisfaction

- *Do the team members enjoy working together?* Most of the team members have very specific areas of responsibility and therefore do most of their work alone. Collaboration is often limited to time spent in status or planning meetings.
- *What conditions could lead to feelings of resentment?* Developers can fail to deliver what is expected of them because their work relies on the cooperation of other teams that have different goals.

 

- *What conditions could prevent team members from working together in the future?* The focus on individual effort and accomplishment makes it difficult for developers to improve their collaboration skills.
- *How can team members best learn from one another?* Some developers actively seek to help each other when help is specifically requested. In other cases developers have invested heavily in strategies that are not well coordinated with their peers. These ongoing design conflicts have not promoted learning but instead further isolated some of the developers from their peers.
- *How are team members expected to accommodate to changes, such as additions to the team, growth, and turnover?* At this time the ability to add new team members is considered an almost impossible task. Whereas the loss of certain key developers would leave large gaps in the team's knowledge about its own product.

**Individual Growth**

- *Do the individual team members grow and develop as a result of the team experience?* Because task assignments are queued for the most efficient resource each developer tends to find himself or herself forever trapped in a narrow area of specialized knowledge.
- *Do team members have a chance to improve their skills or affirm themselves?* Skill improvement opportunities tend to be limited to outside classes and self-study outside of the office. While affirmation of accomplishment amongst peers is typically limited to war stories around the water cooler.
- *What factors and conditions could block personal growth? The* senior team members are overburdened with tasks that only they understand. While new features are postponed because it is impossible to add new team members and have them learn the existing code base on their own.
- *Are individuals' growth needs understood and shared by group members?* Management and team members value the growth of individuals, but this need is often considered secondary to the current set of top priorities.

**Organizational Gains**

- *How does the team benefit the larger organization?* In its current state, the team extends its existing technology in the manner that is most effective for implementation. Then the team provides the technological improvements to marketing to describe customer value and sales strategy.
- *Are the team's goals consistent with those of the larger organization?* Often the goals of the development team lag those of the larger organization. Half completed initiatives still work towards completion even when the larger organization has changed strategy.
- *What other groups, departments, and units are affected by the team?* All other departments within the organization are affected by development. For example, the customer service team often requires assistance from developers to resolve

 

user issues. While sales of the product are often contingent upon a new feature or customization being ready within a specified time frame.

- *What steps has the team taken to integrate its activities with those of others?* Technical leads from all three projects present monthly status reports to the company's management team.

### 2.3 Recommended Culture Changes, The Desired State

The CEO and the VP of Software Development engaged a team of consultants to formulate a plan that would help the software development team become more effective. An additional goal is to build an environment that will allow the company to attract talent and help them quickly become productive contributors. After reviewing the situation and interviewing employees throughout the organization the consulting team returned with a list of several specific changes that it recommended be implemented. Many of these recommendations were based upon a recently formalized set of development practices known as Extreme Programming.

- The centerpiece of these changes is to locate all three teams together in one large open room. Given the architecture of the building, the recommendation was to remove remaining equipment from the spacious factory floor and provide both folding tables and rolling chairs so the team can freely reconfigure the space to suit their needs.
- Provide meeting spaces within the working space by having extra chairs and placing the working space next to walls with whiteboards.
- Provide a common space in the middle of the room with reference texts, free food and free beverages in order to encourage spontaneous and unscripted social interaction between team members.
- Establish and formalize the practice of rigorously building automated unit tests for all code before it is checked into the main build.
- Rotate all programmer assignments so that each programmer increases their awareness of how the entire system works as a whole.
- Every day have a short, fifteen-minute, meeting where everyone stands around the food table and gives a forty-second status report on the task that they are working on or asks for help with something that is causing them problems.
- Maintain each developer's personal space so they can return to their office when they need to do some quiet thinking or make a personal phone call.

#### 2.3.1 Update Project Plans Every Two Weeks

The next change focused on how work was assigned to developers. All feature requests, bug fixes, and other developer tasks are broken down into tasks that take two weeks or less. Shorter tasks are easier to accurately estimate, therefore any task that requires more than two weeks of effort is broken into a series of shorter tasks. Developers can also improve estimates by estimating tasks on a regular basis, and re-

 

estimating all uncompleted tasks. Continuous improvements in estimating will increase marketing's confidence in the developers' estimates. Instead of independently scheduling tasks for each developer, synchronize the entire team so that everyone is scheduled for two weeks of tasks starting on the same day. The programmers estimate the tasks then all of the product managers and product support managers can work together to update the schedule for the next two weeks worth of work by discussing with each other how each task would contribute to the company's goals and making the appropriate management tradeoffs. At the end of each two-week cycle the development team will formally demonstrate the additional capability to the sales and marketing staff so that accurate feedback can be gathered and acted upon. Any task that is left uncompleted at the end of a two-week period is re-estimated at the beginning of the next cycle and reviewed by management for inclusion in the next cycle.

### 2.3.2 Team Productivity over Individual Productivity

The consultants also suggested focusing on team productivity instead of individual productivity. Everyone's number one job should be to help any peer who needs assistance completing their assigned tasks. To make this behavior of helping each other second nature, developers are assigned to work in pairs. Two developers sitting at one computer help each other complete an assigned set of tasks. Build the internal networking within the group by rotating assigned developer pairs for every two-week development iteration, so that by the end of a year every developer has worked closely with potentially twenty-six other developer partners (assuming a large team). Then in order to cross-train individuals every iteration assign a few of the developers to work on tasks that they are unfamiliar with so they will need to rely on their partner or on other resources in order to complete their tasks. This will allow the team to practice important skills in training new hires on the entire set of products and at the same time will build the skills the team will need when asked to focus all of its resources on a few tasks.

All of these recommendations have a common focus. The goal of these steps is to build a team that is working towards on overall team effectiveness instead of merely improving the sum of their individual efficiency.

## 2.4 Creating the New Culture and the Learning System

The consulting team was asked to move forward with the changes that were suggested. To increase the probability of successfully creating a lasting change the consulting team focused on creating a learning system[2] that would allow the programmers to develop new behaviors in a non-threatening environment. An immersion exercise was planned to communicate the new processes to the team members while offering an opportunity to evaluate the practices before the transition plans were finalized.

 

On Monday it was announced that there would be a weeklong training exercise on the Java programming language starting on Wednesday. On Tuesday folding tables were arranged in the unused factory space and one computer was set up for every two participants. A short sample program was written and made available on every computer. When the developers assembled on Wednesday morning, it was explained to them that their goal was to extend the short sample program to prove that the Java programming language could be used to perform the functions required for all three products. This demonstration would not have to be comprehensive, instead it merely needed to demonstrate the ability to read a tiny set of sample data and perform all of the major processing steps that the company's current products could perform. It was further explained that this was to be 100% exercise based and 0% lecture. To ensure maximum learning every day, each person would be assigned a new partner and a new task. Any task not completed on the first day would simply be available for a pair to complete on the second day.

### 2.4.1 Experimenting with New Behaviors

At the beginning of each day, pairs would be assigned to work together and pairs would volunteer for the tasks that had been prioritized for that day's effort. As pairs ran into difficulties, others would overhear their complaints and offer advice from the next table over. By the third day, networks of knowledge were beginning to form with developers seeking help from others with whom they had previously paired. When pairs successfully completed a task, they would gather others near their machine and demonstrate the progress. Each day became a challenge to see if you could complete your task, because others were starting to complete theirs and each team member knew that another task would begin tomorrow.

Junk food was provided at a central location. Also provided were a couple of textbooks with useful information about the Java programming language. When an individual would start searching for the current possessor of one of these texts, other participants would notice their wandering and ask if they needed help. As many of the individuals anticipated, having so many people talking while they worked in a single room caused a significant amount of noise. But it quickly became evident that the noise level was a useful indicator that helped identify when problems needed attention, as well as when success was occurring and being shared. This exercise being labeled as a training exercise also provided participants with a built-in authorization to invest time in helping others. The team members themselves were building a genuine sense of shared responsibility and shared success even while they also competed with each other for bragging rights.

### 2.4.2 Including the Team in the Decision to Move Forward

At the end of five days, the team assembled for informal demonstrations of all that had been built. At the conclusion of the demonstrations the VP of Software Development asked the team several questions.

 

1. Did you have fun?
2. Did you learn more about Java this week than in a normal weeklong course?
3. Did you learn new things about several other team members this week?
4. Were we able to demonstrate more functionality as a team in one week than you thought we would be able to?
5. Did you learn new skills that have always been someone else's responsibility?
6. Did you complete programming tasks even in this noisy environment?
7. Were you able to solve someone else's problems without stopping what you were doing in the process?

For almost every member of the team the answers to all of these questions were yes. Pleased with the results, the VP of Software Development decided to move forward with a full three-month trial of the new processes and announced his decision. Among the group a variety of responses emerged. Some individuals were extremely excited. Others raised objections that they had not been asked if they wanted to work full-time in this new format. But the VP explained that he felt the team had demonstrated a great deal of productivity in this exercise and that the positive responses to his questions also demonstrated something powerful. Therefore he was committed to trying this full-time. The next week began the first full two-week work iteration incorporating all of the practices suggested by the consultants.

### 2.5 One Year Later

The consultants are gone but the development team continues to use the suggested practices. The processes are not simply followed, but are actively improved by the team. As team members have ideas on how to change the process, experiments are proposed and often acted upon, the results are evaluated, and the processes are modified.

Many significant events have occurred. Marketing's demand for earlier shipment of product required that the team grow. In a three-month period the initial team of ten interviewed almost a hundred candidates, extended offers to twenty-two, and hired twenty of them. Those new employees were incorporated into the fabric of the team so effectively that employees who had been with the team only four or five weeks would often be found mentoring newer employees on team practices as well as the collective team work product. While team productivity did not immediately scale with the influx of new team members, team productivity did rise immediately.

A few months after its significant hiring efforts, Root Computer Products was purchased by one of its strategic partners. The executive officers of the purchasing company were impressed with the team's ability to demonstrate their process. This demonstration was accomplished via a walking tour of the factory. Each pair's estimates for all of the current development tasks were tacked to a corkboard for all to review. The tasks selected as the highest priority by Marketing, Sales and Support were written on index cards and tacked to another corkboard next to the names of the pair of developers assigned to those tasks. Tasks that were completed and ready to demo had large green dots stuck to their index cards. Posters with the number of

 

quality assurance tests and the number of times the entire set of tests were run every week were hung all around the factory. While touring the factory, it was easy to observe how the team collaborated on important issues in between tables and then just as quickly disbanded back to their computers to immediately implement their decisions.

While the new parent company has decided to have the RCP team continue to use what the parent company views as an effective process, they have not attempted to implement these processes in any of its other offices. Informal polling of the employees who were affected by these new processes indicate that few if any of the employees would want to return to the old processes. The VP of Software Development still publicly expresses his satisfaction that his day is now spent resolving important issues instead of trying to determine if the members of his team are all working towards the same goals.

### 2.6 Analysis of the New Culture

**Team Productivity**

- *Does the team have a clear goal?* The developers now focus on completing two weeks of tasks as prioritized by the marketing team. At the end of every two week iteration the developers' formal presentations of the completed work are performed within the context of the developers' understanding of the product goal.
- *What objective performance measures have been established at the outset of teamwork?* The developers now focus on accurately estimating and delivering functionality every two weeks.
- *Who are the legitimate clients of the team?* The developers often offer opinions about what features customers might want to see but the developers now serve a single client, the management team as represented by the prioritized fine grained deliverables.
- *Does the team's output (e.g., decisions, products, services) meet the standards of those who have to use it?* Formal demonstrations to the marketing and sales teams help to achieve this objective. Perhaps even more importantly, the ability to provide early beta copies to selected customers provides much improved feedback over the prior development process.
- *Under what conditions should the goal change?* The developers assume that at any two week boundary the marketing team might decide to change strategies or goals. No longer is such a change considered a significant challenge.
- *What sources of information should the team consider to assess whether the initial goal might need to be changed?* By focusing the team on delivering demonstratable functionality every two weeks, marketing has been provided with an invaluable tool for discovering if the current products need to be updated.

 

**Team Satisfaction**

- *Do the team members enjoy working together?* Simply stated yes. Of course there are times where pairs disagree, but rather than hide disagreements by dividing the problem developers are expected to work together to solve problems.
- *What conditions could lead to feelings of resentment?* If an individual feels an exclusive ownership for a piece of code, then rotating task assignments can cause negative feelings to surface. Building team ownership of the code is imperative.
- *What conditions could prevent team members from working together in the future?* If two team members find it impossible to work with each other productively it is possible that their assignments could be selected to minimize their interaction. However, it is a goal of this process to quickly identify and resolve these kinds of issues rather than hide them.
- *How can team members best learn from one another?* By having team members develop using the practice of paired-programming significant learning tends to occur all day long through out the team. This is not simply technical know how, team members dramatically increase their knowledge about their peers' talents and skills. This has dramatically increased the team's ability to collaboratively solve problems.
- *How are team members expected to accommodate to changes, such as additions to the team, growth, and turnover?* Most software teams struggle with how to add new team members. RCP's team managed to triple in size in less than two months with no drop off in productivity while quickly bringing new members up to a productive level of contribution.

**Individual Growth**

- *Do the individual team members grow and develop as a result of the team experience?* All team members grow in this process. Less experienced developers learn new skills more quickly. While more experienced developers receive rapid feedback on what code is maintainable.
- *Do team members have a chance to improve their skills or affirm themselves?* By working on new tasks outside their area of expertise developers can broaden and improve their skills. During the same time period they are often acting as the domain expert for other developers. Being sought out by your peers and being asked to help is often a very affirming experience.
- *What factors and conditions could block personal growth?* While all team members now recognize the needs of all members to continue to grow it is always tempting to optimize short-term gains by assigning all work tasks to the experts. Someone charged with the long-term effectiveness of the team must influence therefore matching developers to tasks.
- *Are individuals' growth needs understood and shared by group members?* Individual growth has become an integral part of the work process.

 

**Organizational Gains**

- *How does the team benefit the larger organization?* By focusing on effectively delivering the most important functionality instead of individual productivity the team is no longer a barrier in management's product planning cycle.
- *Are the team's goals consistent with those of the larger organization?* Instead of trying to perfectly align the goals of the development team with the larger organization, the new process provides multiple forms of feedback to continuously re-sync the development team with the organization's greater goals.
- *What other groups, departments, and units are affected by the team?* The new planning mechanisms have dramatically affected the responsibilities of the product managers. Their ability to affect the development process has been greatly improved and is understood by the entire management team. The CEO credits the changes in the development organization with having a positive affect on all of the other departments in the organization as well.
- *What steps has the team taken to integrate its activities with those of others?* Beyond the planning and demonstrations, the development team allows members of other departments to fully participate in development iterations. This has allowed the customer service engineers and other professional services employees to gain a deeper understanding of the products architecture while at the same time infusing a deeper knowledge of the customers into the development team.

## 2.7 Case Study Epilogue

Change can be as exciting as it can be difficult. But often when the excitement wears off the carefully created changes give way to old patterns of behavior. The members of RCP's team were participated in an unintentional but interesting experiment. Two and a half years after these changes were adopted, the company that purchased Root Computer Products started to downsize. The remote Ann Arbor office was dramatically affected by these events. Some thirty programmers and their managers now experienced in Extreme Programming and used to working in a highly collaborative team re-entered the job market when the Ann Arbor office was closed.

As these programmers interviewed for jobs, many of them would describe the processes of their former employer and express a strong interest in using those practices in any new position that they obtained. One hiring manager tells the story of interview after interview of evangelists who expressed a strong preference for finding a new employer who would embrace these practices. At first he discounted this feedback and continued to interview other candidates. After interviewing six qualified candidates, all of whom expressed these same prerequisites, he decided to hire two of them and start a small project as an experiment. As the management benefits of improved planning, resource flexibility, and higher quality output continue to be demonstrated more of the practices are being implemented across a greater number of projects.

 

The alumni of Root Computer Products have started grass root efforts to adopt Extreme Programming development practices in many of the companies where they have found new homes. The locations where these efforts have executive sponsorship have realized the greatest benefits from these efforts.

### 2.8 Conclusions

A team is defined by the common goal of its members and the interdependence of their efforts. Focusing each programmer on the tasks that best match his or her skills can improve individual efficiency. But the pursuit of optimizing each individual resource's performance can contradicts the two measurements used to define a team. Instead the practices of Extreme Programming focus an entire team on incrementally delivering the greatest amount of business value as defined by business deliverables. For RCP the successful adoption of these practices required a fundamental change in culture that was greatly facilitated by strong executive sponsorship and learning systems that allowed programmers to build new behaviors in a supportive environment.

## References


1. Beck, Kent: *Extreme Programming Explained: Embrace Change*. Addison Wesley Longman, Inc., Reading, Massachusetts (2000)
2. LaMarsh, Jeanenne: *Changing the Way We Change: Gaining Control of Major Operational Change*. Addison-Wesley Publishing Company., Reading, Massachusetts (1995)
3. Thompson, Leigh: *Making the Team: A Guide for Managers*. Prentice Hall., Upper Saddle River, New Jersey (2000)


## Appendix: Questions and Notes for Facilitating Group Discussions

*Rank the recommendations that are most likely to inspire resistance of the team members. For each recommendation, identify steps that could be taken to help facilitate adoption of the new behaviors.*

Working in a pair seems to be the idea that draws the most resistance whenever these practices are discussed. This resistance appears to be rooted in the fact that few programmers have ever worked closely with a peer during their careers. Newly hired graduates seem far more comfortable with this system and indicate that it is something that they practiced with peers in school. While the immersion exercise described in this paper does not solve the problem, it provides a mechanism for developers to experience the positive aspects of shared victories in a low risk situation.

 

*In what ways do you think a pair of programmers might be more effective working together all day? In what ways will they lose efficiency?*

A pair working together will often produce less output than two individuals working apart. But the pair working together will often realize that a task has been successfully completed and stop working on it earlier. Unscientific observation also seems to indicate that a pair produces fewer errors resulting in less rework. This might be because the individual typing is focused on a very tactical level, but the partner freed from typing has more time to consider the strategic ramifications of their decisions. When a pair of developers is not getting along together it is often apparent to the whole team. In a traditional cube village it is common to discover that two individuals are not working well together only when trying to integrate several weeks of their individual work products. It can be far more efficient for team progress to discover that two individuals are not working well together only two or three days into an effort. By gaining this insight early in the process, management can intervene and help facilitate resolution to the problems.

*Working with a team of twelve developers and six computers how would you suggest the tables be arranged to best facilitate communication? How would you change the arrangement for a team of thirty developers and fifteen computers?*

As the team experimented, several rules seemed to evolve. It's easiest to help the team behind you, as you need only rotate your chair and slide a couple of feet. It's relatively easy to slide your chair several feet to the right or left to help folks at a table next to yours. And conversations seem to occur naturally with someone who faces you while you are working. If the tables are arranged in multiple rows, then the interaction between teams more than one row away from each other seems to be less than the interaction between teams that can meet without leaving their chairs. Another behavior that appears to discourage quality interaction is sitting at the same computer for week after week. Such an individual will spend less time visiting other people. They will also tend to treat the computer as an owned possession and thereby treat their partner as visitor. One way to deal with this is to rearrange the workspace once a month. This often unleashes creativity in the team as well as encourages communal ownership of the surroundings. The key in this reorganization is to remind everyone that the goal of the event is better team productivity not better individual productivity.

*Thinking back to the suggestion of a food table, what food would you suggest be provided? What kind of budget would you suggest? Who should stock the table with the food?*

Originally the food table was stocked via donations from the advocates of this idea. The company now funds a couple of hundred dollars for this effort. The snack table is open to everyone in the company, and encourages other departments to visit the developers more often. Every couple of months a new developer volunteers to take over the responsibility for shopping and stocking.

 

*How much time do you think developers will spend in their offices versus how much time they will spend working with their partner in the collaborative space? What are the disadvantages and advantages of this arrangement?*

As time passes, developers spend less time in their private space. Most of the new hires, although they have been provided with equivalent private space, spend almost no time there except to drop off their coat. During the original transition of the team, there was an interesting problem with team members having private space that was away from the common space. In the morning some individuals would enter the common space, see that their partner was not in yet, and return to their private space to read email or perform other solitary work. Their partner would later enter the common space, survey the room, and reach the same conclusion as they returned to their private space. This was solved by setting a time 9:30 am by which all members of the team were expected to be in the common area working.

 