Using Extreme Programming for Knowledge Transfer

Luigi Benedicenti
University of Regina
3737 Wascana Pky
Regina SK Canada S4S 0A2
Tel: +1-306-585-4701
Luigi.Benedicenti@uregina.ca

Raman Paranjape
University of Regina
3737 Wascana Pky
Regina SK Canada S4S 0A2
Tel: +1-306-585-5290
Raman.Paranjape@uregina.ca

ABSTRACT

This paper presents the application of eXtreme Programming to the software agents research and development group at TRLabs Regina. The group had difficulties maintaining its identity due to a very rapid turnover and lack of strategic polarization. The application of eXtreme Programming resulted in a complete reorientation of the development culture, which now forms a continuous substrate in every individual. This allowed the creation of a group identity, thus promoting the integration of short-term projects in the strategic group vision, and complete information sharing.

Keywords

Extreme Programming, Knowledge Transfer, Software Process, Software Lifecycle

1 INTRODUCTION

One of the largest research groups that the authors co-chair in their academic duties is the Software Agents research group. Software agents are relatively small autonomous programs that can accomplish complex tasks by virtue of two capabilities: mobility and collaboration [6]. Although the group involves University professors and students, and it is research-oriented, the environment in which this research takes place is that of a company: TRLabs, a pre-competitive research venture. Student employment is part of the co-operative education program at the University of Regina, and serves well to help our future graduates cope with the industrial reality. TRLabs's research partners include University and Industry. Thus, TRLabs represents a different environment for students and professors alike, an environment with precise deadlines and specific targets.

In mid-1999, our research group on software agents was facing a number of difficulties. To experiment with software agents, our group had to develop a large number of them, each with different behaviors and goals. Moreover, the group had to agree on a common execution model. Finally, since a co-operative work term only lasts four months, we were hard pressed to find limited-scope projects with long-lasting value.

To solve these problems, we decided to adopt eXtreme Programming for both software development and knowledge transfer and integration in TRLabs' corporate culture. Extreme Programming (XP) is a recent development technique [1] to accelerate and streamline software development. Since its initial presentation, XP has been the subject of a substantial amount of controversy, which has given the opportunity to refine it and apply it to many different development environments [3, 7].

In section 2, the rationale for this choice will be illustrated. Section 3 will present the roles and responsibilities of the group. Section 4 will detail our approach. Results will be presented in Section 5. Section 6 will summarize the results into conclusions.

2 RATIONALE

The software agents group at TRLabs and the University of Regina is a relatively large group (around 12 people) whose mission is to explore and harness the power of software agents for research and commercial applications. On the research side, the group has long-term goals for novel agent interaction models (for example, insect-like behavior or data clinging). On the commercial side, the group is researching in distributed economic models for e-business and better human-computer interaction [2].

The group is involved in three kinds of projects. The first kind of projects is strategic, which sets up a theme lasting several years. The second kind of projects is topic-related, usually lasting one to two years. The third kind of project is incubation, lasting for approximately four months.

The main goal of the group is to gain a thorough knowledge of how software agents can best be used and in which areas software agents obtain the best results. In order to achieve this goal, it is necessary to have a pyramidal structure that concentrates knowledge and distills it into strategic and operational rules.

Such a structure is easy to build on the administrative side, but it is very difficult to enforce knowledge concentration. This is especially true when a high turnover rate is present.

Soon after XP was introduced, the group decided to adopt a less stringent XP approach for agent development. The reasons behind this choice were simple: the sheer number of agents required for a realistic simulation required it. In addition, software agents do not lend themselves to be developed via pseudo-random modifications such as with

genetic algorithms.

More recently (beginning of 2000) the group has started to use XP also as a means to transfer the knowledge acquired from incubation projects into a stable corporate culture base. It is interesting to notice that this choice was not entirely conscious, as XP-related behavior had pervaded the group's activities so much that its adoption as a means to share information became 'natural.'

3 ROLES AND RESPONSIBILITIES

The group is organized as a team with a pyramidal structure (Fig. 1). Top management is technically not part of the team, but they affect its strategy. Project members at the strategic level include managers and professors. They have a better view of the group, as they are generally involved for a longer time. There have been up to three such people in the group. Project members at the topic level are usually research engineers or graduate students. Project members at the incubation level are students in the co-operative employment program or temporary "overflow" personnel.

The goal of incubation project members is to carry out specific investigations or small software developments (such as a special class of agents). Topic project members work on single topics for longer, and often their work spawns smaller incubation projects. They benefit directly from incubation project knowledge. Strategic project members manage the group and integrate the knowledge in the group's corporate culture. They also manage external technology transfer.

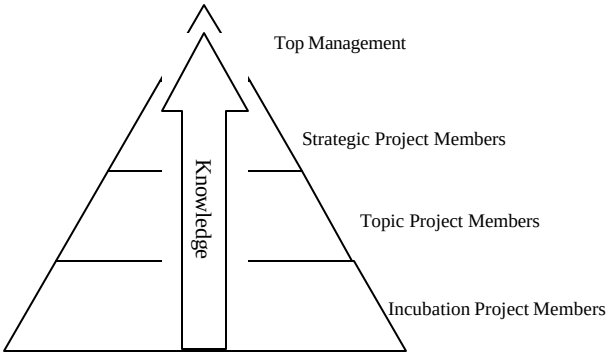

Figure 8: Team structure

4 APPROACH

After the decision to use XP was taken, a strategic meeting was called to choose the best way to introduce it to the group. The management was aware of some specific techniques for introducing XP to professionals [1] and students [8]. After careful consideration, it was decided that XP would be introduced as a natural extension of the software process. In fact, up to that point, no software process was enforced, but everyone in the group was expecting a process to be established. Hallway conversations hinted that the process would be tedious, clumsy, and very structured.

The introduction of a highly flexible, lightweight process that was aimed at maximizing compatibility of the approaches was highly welcomed. It was generally perceived that programming agents was still a lot of fun – and this is one of the main tenets of XP.

The process regulates the programming activity and the reporting/planning activity. Programmers are asked to comply with the general coding standards that emphasize code understandability [5]. Group programming is encouraged, but not mandated. Reuse in software agent is systematic, not occasional, and thus specific guidelines have to be given as to how to facilitate it. Refactoring is a crucial point in software agents, as it allows them to change easily. Refactoring was present in the agents group even before the group knew it was called refactoring.

The reporting and planning activity centers on weekly meetings. Meetings take place in a boardroom equipped with computers, video links, and a remote connection to TRLabs Saskatchewan's headquarters in Saskatoon. All project members are required to participate. Each meeting begins with a presentation from a group member. Then informal status reports are given. Everyone is free to ask questions and give suggestions. There is a general feeling of unity and no comment is ever interpreted in a negative way. Problem reports and setbacks are regarded as a group opportunity to solve a challenge, not as something to condemn. Lastly, replanning takes place. The group meeting often reaches consensus, but at times it is necessary to exercise leadership and redefine the group's priorities.

The boardroom is also ideal for top management if they wish to attend the meeting sessions. The conference video link also allows externals to be invited for presentations that inject new knowledge in the group. The computer facilities in the boardroom are used often for live agents demonstrations.

Documentation is not optional in this kind of endeavor. However, there is no defined standard for documents, as they are built only when there is reason to build them. The group does not write user manuals before the software is written. A byproduct of presentations is a good set of tutorials and programmer's manuals.

The group does not use code protection mechanisms, although versioning is employed. Continuous refactoring and reuse had a very interesting effect: no one was ever affected by the 'second-system effect' [3]. As a result, the agent system is undergoing profound changes all dictated by contingent necessities and strategic goals, but one cannot say that the whole system has a version number. Rather, each component has a unique version.

For example, the agent execution environment kernel took approximately four months to write, as it was an incubation project. Since the kernel is a strategic asset, a more conservative and traditional approach would have

had most resources dedicated to it, with potentially disastrous results if the system were ever to be touched again. When the agent group decided to change the agent execution environment to separate the kernel's user interface and write a new status agent, the whole task was completed in just four weeks.

In conclusion, our approach to XP is tailored, and not 'pure.' For example, although pair programming is encouraged, it is not mandated. Group programming often takes place, and it is very welcome. Forty-hour weeks are difficult to achieve, as everyone is always very excited about the week's achievements. However, last-day crunches are dutifully avoided. The presence of a deadline becomes accessory to the development challenge that makes the group's achievements so important and rewarding. Group members are always free to offer comments and suggestions on every agent project, and no question is a bad one.

5 RESULTS

The first result is that XP for agent programming works. The group has been successful in developing an agent execution environment, numerous collaborating agents, an economic marketplace for CPU resource allocation, and many other projects.

Moreover, it is now ascertained that project results survive radical turnover rates. This was demonstrated with the agent execution environment, which is being used and modified presently, even if it was the result of a short-term project. The strategic importance of this fact is invaluable to the group.

XP's evolutionary development eliminated completely the 'second-system effect' in all projects [3]. This becomes crucial when the system has to be released in a commercial environment that is now used to fast development times and just-in-time delivery.

The continuity of this approach, the availability of tutorials, and the communication-centered approach selected make it possible for team members to learn the basics on agents in less than two weeks. Everyone is immediately productive, signifying that the group was effective in transferring knowledge. This knowledge is not only theoretical (as one would expect, with the involvement of University professors) but it is also practical, leading to very fast development cycles.

The use of a lightweight process was also extremely beneficial. New group members do not have to learn a large process structure, and so they become accustomed to the process much faster (a week) than if they had to learn a full formal process model.

Finally, it is also worthwhile to mention that during the group's life, turnover has happened at all three levels of the team structure (Fig. 1). In all cases, the specific choices made avoided loss of knowledge, even though in case of the strategic project member the whole group's strategy was realigned. Strategic focus was maintained and production rates never dropped, although of course individual productivity did vary.

6 CONCLUSIONS

This paper presented an account of the application of XP to a group developing software in the software agents' area. The goal of this operation was to improve software development and preserve and transfer knowledge throughout the group so that the group could still work and be productive in an environment with high turnover.

XP proved itself successful in every occasion, and was invaluable not only in raising productivity, but in keeping it high. As a result, group members feel more productive and group morale is soaring. Proactivity has become the norm, and problem solving has become a welcome challenge. Boredom has vanished.

The advantages do not stop with the agent group, however. On the company side, TRLabs has increased its production rate, and has confirmed its commitment to the co-operative program at the University of Regina. On the University side, students employed for a work term feel a sense of purpose and unity that confirms their willingness to learn. For the co-operative program, this achievement is a confirmation of its usefulness. Students that took part in this program are better-motivated individuals whose placement rate is noticeably higher.

The approach we selected for XP is certainly different from what an 'XP purist' might produce. The more outstanding characteristic of our approach is that we are using XP mainly for knowledge transfer. XP for development, in fact, does not make full use of XP's principles and thus may appear lacking. However, XP originated mainly because other methods lacked results.

We believe that the adoption of XP in our group has created advantages, and we are collecting evidence of this on a daily basis.

ACKNOWLEDGEMENTS

The authors wish to thank TRLabs for providing the environment and the means to support this experience. Some of the work described in this paper has been supported by the National Science and Engineering Council of Canada (NSERC).

REFERENCES

1. Beck, K. Extreme Programming explained: Embrace change. Addison-Wesley, 2000.
2. Bredin, J., D. Kotz, and D. Rus. Economic Markets as a means of open Mobile-Agent Systems. Proceedings of the Workshop ``Mobile Agents in the Context of Competition and Cooperation (MAC3)" at Autonomous Agents '99, pages 43-49, May 1999
3. Brooks, F.P. The mythical man-month, anniversary edition. Addison-Wesley, 1995.

4. Gamma, E., and K. Beck. Test Infected. Web Site On-Line at: http://members.pingnet.ch/gamma/junit.htm

5. Holmes, N. Why Johnny can't program. Computer V33 N12 (December 2000), IEEE Press, p. 158-160.

6. Maes, P. Software agents tutorial. Web Site On-Line at: http://lcs.www.media.mit.edu/people/pattie/CHI97/index.htm

7. McBreen, P. Applying the lessons of extreme Programming. Proceedings of the: Technology of Object-Oriented Languages and Systems (TOOLS-34"00), August 2000.

8. Wege, C. and F. Gerhardt. Learn XP: Host a bootcamp. Proceedings of XP2000, Italy, 2000 (CD Edition).