Extending Patterns for Fearless Change

Daniel Cukier, Department of Computer Science - University of São Paulo
Fabio Kon, Department of Computer Science - University of São Paulo

The software industry is very dynamic and new ideas arise all the time from virtually any part of the world. It is not guaranteed that these ideas will be adopted, mainly because, among other obstacles, the solution may imply on having people change their way of thinking. Different from people, computers receive well defined commands and execute them precisely. We should take into account that human beings are independent and unpredictable. Despite this unpredictability, we can find some behavioral patterns to help us deal with several situations, allowing us to achieve our objectives.
In this paper, after a small introduction to the Patterns for Introducing New Ideas proposed by Mary Lynn Manns and Linda Rising, we propose four new patterns that can be added to the original catalog. In one of these new patterns, we show the great importance of combining artistic activities with day-to-day activities of people who work with software development and how Arts can help us to introduce new ideas. The study of some practices such as theater, painting, poetry, music, and meditation allowed us to find some connective elements between the purely mathematical side of the human mind and its creative, artistic one. Software development should be approached as a "human activity", rather than a solely technical or logical one.

Categories and Subject Descriptors: D.2.9 [**Software**]: Programming Techniques—*Management*; J.4 [**Computer Applications**]: Social and Behavioral Sciences—; J.5 [**Computer Applications**]: Arts and Humanities—

General Terms: patterns, patterns for introducing new ideas, art, agile
Additional Key Words and Phrases: behavioral patterns, patterns for introducing new ideas, do art, let them play, Mary Lynn Manns, Linda Rising, fearless change, art, agile

ACM Reference Format:

Cukier, D. and Kon, F. 2011.Extending Patterns for Fearless Change. 18th Conference on Patterns Languages of Programs (PLoP) - Portland, Oregon, USA (October 2011), 10 pages.

1. INTRODUCTION

To explain the patterns for introducing new ideas, we start with an example scenario: the adoption of the *eXtreme Programming (XP)* [Beck 1999] method in a company. XP is one of the best-known agile methods and is widely used in both academia and the software industry [VersionOne 2009]. Introducing XP to be used in a company is not trivial. Teaching the methodology practices is not enough, because practices, sometimes, are based on values that go against the company values. For example, XP suggests pair programming. For that, people need to sit together in front of just one computer and discuss about what they are trying to solve. In an office where people are not used to discuss or sit together (no problem to believe this exists), to make them change their behavior can be shocking [Ryan and Carter 2010]. Implementing XP, specially in already consolidated companies, means implementing deep *cultural changes* and this task can take months, even years.

Mary Lynn Manns and Linda Rising also experienced a scenario in which we need cultural changes and they wanted to convince people to change their mindset. Their experience resulted in an excellent catalog of patterns: the book *Fearless Change - Patterns for Introducing New Ideas* [Manns and Rising 2005]. These patterns provide some insight into the difficult task of introducing new ideas into any organization. When the new idea brings cultural

changes of any kind, then the task is even more complicated. The nature of patterns are not revolutionary ideas. They are simply things that can sometimes be easily applied, with great results. We studied these patterns and used them in a Brazilian software company [Cukier 2009]. Our main mission was to introduce agile methods in the company. While using some of the original catalog patterns, we realized that something more could be added; we then identified four new patterns, which also helped us in our mission. These patterns are described in the sections below.

2. FOUR NEW PATTERNS ADDED TO THE CATALOG

In this section, we present four new Pattern proposals, which were experienced by us. We are sure other people experienced similar situations, which could be added to our experience to enrich the pattern usage examples.

Except for the Pair to Share pattern, which can be used by mostly everyone, the other three patterns are targeted to the Evangelist, the person whose job is to spread the new idea in his environment. The focus of these four patterns is on the task of learning a new idea. Let them play and Do art can be used to create a friendly and light environment, creating an appropriate atmosphere to introduce and teach new ideas. Pair to Share is used to spread the idea more quickly throughout the organization. Deep Dive will be useful when you lost a good idea and want to have it back.

We will use a format similar to the one used in Marry Lynn Manns and Linda Rising's book [Manns and Rising 2005]. In the book, the authors tell us a short story related to the pattern. This story comes in *italics*, followed by a very short and objective phrase in **bold**. After this, they describe the pattern in more detail. Sometimes, we will also cite some of the original catalog's patterns. We will use a Special Format when citing them. In Appendix 2.4, we give a short description of the patterns cited here.

2.1 Let Them Play

> "It is through creative perception, more than everything, that an individual feels the dignity to live his life." - *Donald Winnicott*

When a specific subject is considered extremely serious and hard, create a game that is related to this subject and let people play it.

A long time ago, I had the big wish to implement Agile Methods in our company. When I tried to talk about the topic with the CTO, everything always looked so hard and heavy... Then I decided to organize some rounds of the XP Game [Succi et al. 2002]. After these events, people started to get enthusiasm and talk about Agile spontaneously. The game woke up their playful, artistic, human side and people began to enjoy this kind of experience inside the company.

You are an Evangelist who is trying to spread an idea across your organization. The culture in the organization is open to creativity and innovation. Even when people are busy with their work, they accept to stop for a coffee and have an informal conversation. Managers and employees are open to participate in exercises and workshops, so you can play with them to introduce your idea.

There is a specific subject, considered extremely difficult for people to understand it.

For instance, learning and absorbing an agile method such as Extreme Programming [Beck 1999; Beck and Andress 2004] is complex and requires time. People are already very busy with their own jobs, and if they fill they will have more work, they will naturally reject it.

Therefore

Create a game that is related to this subject and let people play it.

By applying the XP game[Succi et al. 2002], one does not have the intention to teach precisely all the methodology practices. But some practices of this methodology can be seen in the game. Real situations are simulated using playful and fun activities. The customer stories are fun and motivate people to know more about the methodology.

The human being is used to learn things by experiencing them or through real stories. By experiencing a real story, one has the opportunity to capture the "essence" of the innovation (just as pattern writers try to capture the essence or heart of the pattern in the opening story).

Domenico De Masi describes a space where (1) work, (2) study, and (3) play coexist harmoniously. He calls this space "Creative Idleness" [De Masi 2000]. This space is represented by region number 7 in Figure 1. The industrial revolution brought to modern world an idea that divided people lifetime into two segregated parts: personal activities and working time. The production lines have the objective of maximizing time and space usage to obtain greater profits. In production lines, the first worker screws a screw, the next one hits with the hammer, the next verifies whether the screw is well screwed, and so on. In the post-modern world, machines substitute human work more and more. Then, man passes from manual and body work to intellectual activities and thinking. Creative work is the right new model for the 2000s. In this context, man can retake his medieval habits, when work, leisure, and study were integrated in the same environment.

Let them play is a way of bringing the "Creative Idleness" to the usually formal and serious work environment. Staying out of formalities, people become naturally more open, not just to listen to new ideas, but also to create.

This means that man comes back to the state where it is allowed to work at home. It is allowed to stop in the middle of the day for a break, to change night for day, to have flexible hours. The measure of quantity of work done becomes more subjective. It is not the number of lines of code that matters, but the software quality, the quantity of useful features that a system has, or how user-friendly the system is.

This pattern is related to the Do Art pattern, described later on this paper. In Let them play the focus is on encouraging people to have a good time on playing, while in Do Art people do not necessarily need to be part of the play. In the Do Art pattern, art is a central idea and mandatory, while in Let them play, art can be useful, but is optional, since games and people's actual experience is the central point.

A possible difficulty on applying this pattern is that some people are just too serious to play. There are people who don't like playing in the work environment, and you have to respect them. You may also find some resistance from the executives and directors on letting people play during working hours. Some modern technology companies have video game and playing rooms in their offices, but today this is still more an exception than a rule. Most companies continue to be conservative on mixing work with fun and, on these conservative environments, it is better for you to go easy. Maybe, before using this pattern, you should do a Trial Run and go Step by Step.

An important practice in XP is talking frequently and face-to-face to the customers. Sometimes, developers (or IT departments) are not used to talk directly to their clients or people who are interested in their software. They are used to communicate using email or, in the best case, instant messaging tools. Face-to-face conversations are unusual. "The Communication Game" has the objective of convincing people to change their behavior when communicating software requirements. In the game, the team is divided into two groups, one representing the customers and another the developers. The developers objective is to draw, in a paper, a figure described by the customers. The customers can only write messages in plain English to describe what they want. They never meet the developers. If the developers have any question, they can ask also using written messages. At the end of the game, people get really convinced that it is not possible to collaborate efficiently (and create good software) just using email or written messages.

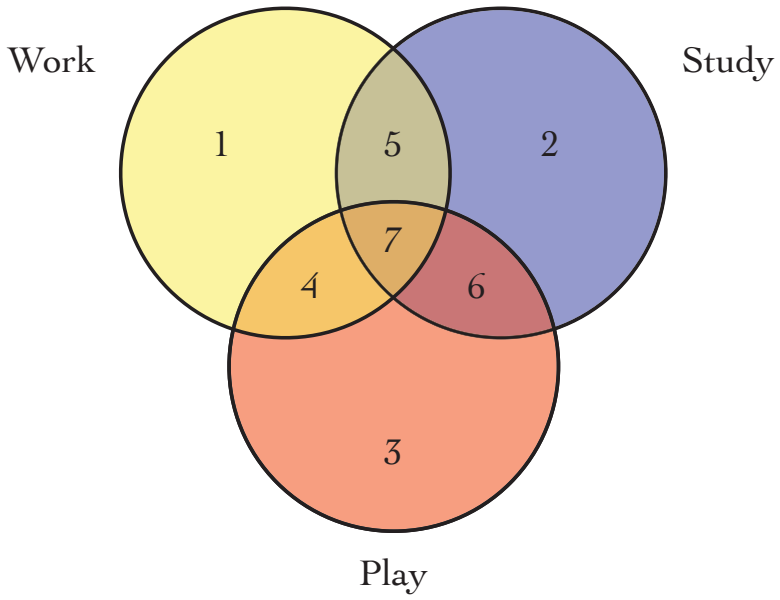

Fig. 1. Work x Study x Play

Another usage example of this pattern is the "Lego Lean Game" [Sato and Trindade 2009]. The Lean Software Development method [Poppendieck and Poppendieck 2003] is hard to learn and even harder to put into practice, since it demands cultural changes. The game proposed by Sato and Trindade is an excellent joyful approach to get people interested in the topic. The Lean method concepts are explained while people build houses with Lego.

You can also use this pattern in Universities classes. For instance, if you want to teach your students to build websites, start telling them to use as many of the functions as they can in the software to make the messiest webpages they can. Create a small contest and vote for the best one. After this mess, you can get serious and start building real websites. You will certainly laugh about the creations – it is a nice "bonding" experience between teacher and students.

This pattern is observed in the Domain Driven Design (DDD) play: we organized an event where we put software engineers to act this play. After that, we realized that the concepts of DDD were more widely accepted and understood[1]

2.2 Deep dive

> "If you want to catch little fish, you can stay in the shallow water. But if you want to catch the big fish, you've got to go deeper" - *David Lynch*

To recover some idea or get deep into it, deep dive and stay a whole day focused.

[1] A video of this play can be found at http://goo.gl/dvonk .

I was tired of writing my paper. The ideas were not clear, the arguments did not come to my mind. I was starting to get depressed. The less I looked to my text, the less I wanted to know about it. Ideas started to disappear. One day, I decided to dive into the work. I proposed to myself to stay 12 non-stop hours thinking and writing about the subject. It would be only that time, shock treatment. Even if nothing came to my mind, I would stay focused on the subject. I would not answer the phone, read emails, surf on the Web. Nothing! Just stay the whole day over articles, books, and texts. This day was very productive. I realized that fatigue came much more from the aversion of having to do the work than from the work itself. After this long day, motivation came back and I could recover the rhythm I had in the beginning.

You are an Evangelist and realize that after some initial effort to introduce a new idea, you come back to your daily work and forget the idea. You know the idea is good. People already showed interest in it. Some change started to happen, but this change was not enough to keep you motivated.

You have lost the idea you were trying to introduce in the middle of your daily work.

Many times a good idea gets lost in the middle of other priorities and things to do. People are always too busy and lose their focus easily. This pattern helps to recover focus. With focus, you stimulate constant and deep thinking about the idea.

Therefore

Deep dive and stay a whole day focused on the lost idea.

Make people stay for some hours talking, hearing, and thinking about the subject. If possible, make them dream with the idea. This pattern is useful to recover a good idea that started to get lost. It will give you strength to move on and apply the change. For that, promote intense sessions where people can permanently stay in contact with the idea.

The main objective of this pattern is to recover the lost idea (the one you want to introduce in your organization) and bring it back to your daily work. Since you recover this idea after a deep dive on it, you can release the pressure and get back to the light way of doing things. If you really succeed after a deep dive, do not forget to celebrate this Small Success.

Eventually, this pattern can bring some discomfort to some people. Not all of them are prepared, or want, to dive deep. Try to choose the right people for these activities and do Just Enough. Make things clear to people: this intense activity is temporary and try to explain your objectives of recovering a lost idea. But notice that this pattern does not tell you to be exhaustive. It is just a matter of deliberated focus, just a way to program yourself not to be interrupted by others or by your own mental distractions. This pattern may actually not work at all for people who do not like to work under pressure, even if this pressure comes from the inside, not outside. This pattern requires a lot of self discipline, and not everybody may be prepared to it.

This pattern is related to the pattern Location, Location, Location in *Fearless Change* [Manns and Rising 2005], but it suggests something else. You will not just remove people from their ordinary habitat. You will put them into an intense treatment. You will stimulate everybody to dive deeply. The topic you want to teach will not be observed just superficially. Close details will be given, so they can reflect, question, argue. From this intense work, the outcome can be both compliments or criticism and skepticism. Be prepared to clear all doubts and look straight to problems.

Some students who worked in the InteGrade project at IME-USP promote programming marathons once in a while. Their objective is to dive deep into the project's code. Basically, marathons are long meetings, where InteGrade developers program new features, create automated tests, solve bugs *or refactor existing code. These*

meetings used to be focused on refactoring, but since 2008, they had a more general purpose and started to include other activities [int 2009].

This pattern is also related to the Pomodoro Technique [Cirillo 2007], but differently, it tries to get focus more intensively and for a longer period. All this intensity does not mean that it needs to hurt. Everything still can be done in an organized and joyful way. You can use the Do Food patterns to make things easier and lighter. You can also use Art on the process, stimulating people to create plays or music, as explained on the (Do Art 2.4) pattern.

2.3 Pair to Share

> "Switching pairs often is an effective strategy for spreading knowledge and information around" - *Laurie Williams*

To spread an idea, encourage people to work in pairs.

Some time ago, we wanted to use RubyOnRails to develop a new application in the company, but the official development platform was .NET and it was very difficult to convince the directors that, for that particular case, we should use RubyOnRails. One day we asked a skeptical (but very respected) manager to pair with us in a RubyOnRails prototype. We were just asking him to sit with us and watch how this new language and framework was great. After pairing two days, he became an enthusiast in Rails and helped us to convince the directors.

You work in a place where people are used to collaborate with each other. This usually happens in meetings and also when someone asks other people's help. Communication happens through emails and sometimes (but not frequently) face-to-face conversations. You want your new idea to be spread quickly in this environment.

Some new ideas take a long time to spread out.

Sometimes, people have the natural tendency of isolating themselves and being selfish. This behavior leads to the retention of good ideas in people's mind. You want your idea to get spead, not to be stuck in peoples heads. To get a new idea spread and glued, it needs to be communicated. It needs to be clearly understood by everybody. Communication is carried out by at least two parts: the part who communicates and the part who listens. There is no communication without the pair Sender-Receiver. Moreover, the new idea message needs to be spread throughout the whole network of people involved in the ideas subject.

Therefore

Encourage people to work in pairs to facilitate spreading the new idea.

When we talk about an idea, the action of speaking about the idea helps us to develop the idea in ourselves. When we speak we have the opportunity to listen to ourselves. We also have the chance of listening to other people and trying to understand their different points of view, giving us the opportunity to elaborate a better approach about that subject.

By putting people of different skill levels together, the newbies have the great opportunity to learn from the experts. This is a practice that motivates people a lot. It is also great for the experts to develop their teaching skills, improving their ability to transfer knowledge and consequently feeling good to share their experience with others.

Working in pairs itself leads to communication. The pattern Bridge-Builder (See Appendix 2.4) proposes to put the skeptics and the early adopters to work in pairs. In our pattern Pair to Share, we have to pair everybody with everybody, so the message flows more quickly. After using Bridge-Builder, we can get the already convinced skeptic and pair him with another skeptic. There is no better person than a skeptic to convince another skeptic. By

doing this, the idea spreads in a cell system. There is no centralized or single point of information. Knowledge and information exist, but in a decentralized form [Brafman and Beckstrom 2006]. One transmits to the next, who transmits to the next, and so on, like a viral dissemination, which is much more efficient.

Besides facilitating the spread of knowledge, this pattern is not always effective, specially for people who are not prepared to work in pairs. Some people simply refuses to work in this way. You need to go easy with them and respect their limits. Another possibility that cannot be ignored is the danger of two skeptics starting a movement trying to undervalue your idea. When you give people permission to organize themselves for a common "good", this "good" is not always what you expect it to be, so be prepared to face these situations.

Encouraging people to interact with each other helps them to recover the contact and realize that when having contact, they are more productive and have more pleasure. This kind of exchange is very known nowadays in what we call the *Gift-Economy* (a kind of economy based on cordiality, gratifications, and moral and ethic values [Cheal 1988]). This type of economy is what moves some of the most important Free and Open Source Software initiatives [Gabriel and Goldman 2005; 2000].

You can use a practice similar to Pair Programming – one of the most important practices in XP [Beck 1999] – in this context. One of the objectives of this practice is to disseminate knowledge within the team quickly. By pair programming, collective ownership is maintained and valued by the group of developers. The practice in pairs has many benefits, described in many scientific publications, such as those by Laurie Williams [Williams and Layman 2007; Williams and Kessler 2002]. They prove the effectiveness of this way of working in many aspects.

2.4 Do Art

"When technology and art are together, magical things happen" - *Walt Disney*

To plant an idea in the people's unconscious, do Art.

I wanted to convince people that using automated tests had benefits and was fundamental to high quality software development [Bernardo and Kon 2008]. Even after talking a lot about this subject and showing books and papers about tests to people, it seemed that they were not completely convinced that using tests was really important. Then I decided to compose a song about this theme. I realized that, after listening to the music, people started to get more sympathy towards tests.

(singing with "Tiro ao Alvaro" melody, a samba by Adoniram Barbosa)
After so many dirty
Hacks to make it work
My code looks like
You know what man?

Mammmaaaa's Macarroni
I'll have to refactor
(it can't be like this)

If this method is pulled up to the parent class
And this other we kill forever
With a test I will complete it.

I'll test
To eradicate this big suffering
I can't handle
I'll create
The automated test.

You and people in your organization are open minded. You love informal conversations, talking not just about work, but also about movies, restaurants, music, and other personal preferences and activities. You are also interested in having new experiences, and learning to perform things that you are not used to. However, even those people can become very skeptical if the idea you are trying to introduce is very different from what those people are used to. Sometimes a new idea needs to be understood in a subtle level, before it is realized by the conscious mind. Moreover, once an idea get captured by the people's unconscious, you want them to express and spread this idea even without knowing.

You want to plant a new idea in people's unconscious or make them express their inner thoughts with others

Programmers who did not have, in their daily work, activities that stimulated sociability are inclined to develop a distorted vision of reality. They tend to believe that human beings functions in the same way as machines. This vision can be extremely maleficent, not only for the life of these individuals, but also for people who surround them.

A technology department purely based on computer thinking is not able to create. The computer does not create things. The one who creates is the human being. Some ideas of how to substitute computational thinking by artistic thinking are suggested by Setzer [Setzer 2006; 1997]. People need to create thoughts in an abstract (mathematical) space to program or to use a computer. Richard P. Gabriel says that

> "Traditions of computer science and software engineering have tried to turn all aspects of software creation into a pure engineering discipline, when they clearly are not [...] We should train (software) developers the way we train creative people like poets and artists." [Heiss 2002]

There are basically two groups of people involved in the software development process: software creators and users. Although a barrier appears to separate these two groups, in [Cukier 2009], we see then evidences that this division or gap can be bridged if IT professionals nurture their artistic and interpersonal relationship skills in addition to their obvious technical ones.

Therefore

Create artistic objects about the your idea or promote sessions for people to create their own piece of art.

There are many ways of doing Art with the objective of introducing a new idea:

—You may compose a song about the idea that is being introduced and teach people how to sing it;

—You may write a poem whose subject reminds the idea;

—Create a theater scene, that has the idea to be introduced as implicit or explicit subject. Invite the involved people to watch or, maybe, act in the play;

—Create a drawing or an image to be used In Your Space (see Appendix 2.4);

—Make an sculpture (it can be made with children's modeling dough). It is not necessary to be a great sculptor to create a satisfactory sculpture.

—Force people to think out of the box about your idea, to think in a playful way. Propose any kind of game, workshop or group dynamics, like in the patterns Let them play (2.1). Things can be serious and fun at the same time (as coding Dojo is [Sato et al. 2008]).

Art help us to develop social sensibility, empathy. Besides acting on the unconscious, Art promote meetings, knowledge, and experience exchange. Art also helps people to relief from stress (the practiced art as well as the

appreciated art). Give people a green field and they immediately start to create. White paper is the opportunity for a new drawing.

The study of some practices such as theater, painting, poetry, music, and meditation allowed us to find some connective elements between the purely mathematical side of the human mind and its creative, artistic one. Software development must be approached as a "human activity", rather than a solely technical or logical one.

This pattern must be used with caution. People not always face Art as something serious, specially in offices. They may think you are kidding or running away from your job. Some may not understand that artistic expression **is part of your job**. Do Just Enough and in the Right Time (See Appendix 2.4). Besides this, not everybody likes every style of art. For each style there is the right audience. You must learn how to recognize your audience and apply the right technique for each specific audience.

The patterns for introducing new ideas can be used in many contexts and to introduce any kind of idea in all kind of organizations (not necessarily software development companies). But the context we are most interested in is the software development one.

In 2009, in the Institute of Mathematics and Statistic at the University of São Paulo, we presented the master thesis ***Patterns for introducing new ideas in the software industry****. The presentation was not the usual "slides with bullets" boring talk. It was a theater play, interpreted by Daniel and other three actors. We used Art to convince people that these Patterns were good to be used. In fact, we used our own Pattern to introduce itself. We also recorded the presentation and after some time we made it available on YouTube*[2] *and Vimeo*[3]*. By the date of this writing, this video had more that 9,000 views. People from all parts of our country get in contact to congratulate us for the performance and the innovation of presenting something so different and unusual.*

Of course, not everything was good as we expected. The professors criticized a lot the master thesis and I had to change the text to satisfy the scientific rigor the professors said it was missing. Nevertheless, the outcome was that we effectively succeeded in introducing people to the Patterns and some professionals started to use these patterns to introduce their ideas in their own contexts. Maybe if the Patterns were presented in a usual form, it would never become so "famous" as it became. Our intention is to motivate people to create their own Art pieces and introduce their ideas using artistic skills. As it was said by Peter Brook, a famous English theater and film director [Oida and Marshall 1998]:

> "In our day, the tragedy of art is that it has no science and the tragedy of science is that it has no heart."

APPENDIX

In this Appendix we bring a small explanation of each pattern from the Mary Lynn and Linda's original catalog:

—Evangelist - To begin to introduce the new idea into your organization, do everything you can to share your passion for it.

—Just Enough - To ease learners into the more difficult concepts of a new idea, give a brief introduction, and then, make more information available when they are ready.

—The Right Time - Consider the timing when you schedule events or when you ask others for help.

—In Your Space - Keep the new idea visible by placing reminders throughout your organization.

—Location, Location, Location - To avoid interruptions that disrupt the flow of an event, try to hold significant events off site.

—Bridge-Builder - Pair those who have accepted the new idea with those who have not.

—Do Food - Make an ordinary gathering a special event by including food.

[2] http://goo.gl/PdwmK

[3] http://vimeo.com/6094673 and http://vimeo.com/4766693

—Small Successes - To avoid becoming overwhelmed by the challenges and all the things you have to do when you're involved in an organizational change effort, celebrate even small successes.

—Step by Step - Relieve your frustration at the enormous task of changing an organization by taking one small step at a time toward your goal.

—Trial Run - When the organization is not willing to commit to the new idea, suggest that they experiment with it for a short period and study the results.

Acknowledgements

We would like to thank Marry Lynn Manns and Linda Rising for the very precious advices given during the shepherding process. Their help were essential to increase this paper quality and we learned a lot from them. We also thank Richard P. Gabriel for his infinity patience during the Writers Workshop sessions. All participants in the Writers Workshop deserve our thank for their comments and help. Least, but not last, thanks for Lise Hvatum and all the Hillside group for making PLoP'11.

REFERENCES

2009. Refactoring - Intergrade: OO Grid Middleware. Available at: http://goo.gl/T41FL. Acessado em: 20/03/2009.

BECK, K. 1999. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional.

BECK, K. AND ANDRESS, C. 2004. *Extreme Programming Explained: Embrace Change* 2nd Ed. Addison-Wesley Professional.

BERNARDO, P. C. AND KON, F. 2008. A Importância dos Testes Automatizados. *Engenharia de Software Magazine*, 54–57.

BRAFMAN, O. AND BECKSTROM, R. A. 2006. *The Starfish and the Spider: The Unstoppable Power of Leaderless Organizations*. Portfolio.

CHEAL, D. J. 1988. *The Gift Economy*. Routledge.

CIRILLO, F. 2007. *The Pomodoro Technique* 1.3 Ed. Francesco Cirillo.

CUKIER, D. 2009. Padrões para Introduzir Novas Idéias na Industria de Software. M.S. thesis, IME-USP.

DE MASI, D. 2000. *O cio Criativo*. Sextante.

GABRIEL, R. P. AND GOLDMAN, R. 2000. Mob software: The erotic life of code. *ACM Conference Object Oriented Programming, Systems, Languages, and Applications - Keynote Speech*.

GABRIEL, R. P. AND GOLDMAN, R. 2005. *Innovation Happens Elsewhere*. Morgan Kaufmann.

HEISS, J. J. 2002. The Poetry of Programming – Interview with Richard P. Gabriel. http://goo.gl/nFT2w.

MANNS, M. L. AND RISING, L. 2005. *Fearless Change: Patterns For Introducing New Ideas*. Addison-Wesley, Boston.

OIDA, Y. AND MARSHALL, L. 1998. *The Invisible Actor*. Routledge.

POPPENDIECK, M. AND POPPENDIECK, T. 2003. *Lean Software Development: An Agile Toolkit*. Addison Wesley.

RYAN, T. AND CARTER, J. 2010. Driving technical change: Why people on your team don't act on good ideas, and how to convince them they should. *Lewisville, Tex*.

SATO, D. AND TRINDADE, F. 2009. The Lego Lean Game. *Agile Processes in Software Engineering and Extreme Programming*, 192–193.

SATO, D. T., CORBUCCI, H., AND BRAVO, M. V. 2008. Coding Dojo: An Environment for Learning and Sharing Agile Practices. In *Agile, 2008. AGILE'08. Conference*. 459–464.

SETZER, V. W. 1997. O Computador como Instrumento de Anti-Arte. *Anais do VIII Simpósio Brasileiro de Informática na Educação, São José dos Campos*, 509–530.

SETZER, V. W. 2006. Um Antídoto Contra o Pensamento Computacional. http://goo.gl/sy8tk.

SUCCI, G., MARCHESI, M., WILLIAMS, L., AND WELLS, J. D. 2002. *Extreme Programming Perspectives*. Addison-Wesley, Boston, MA, USA.

VERSIONONE. 2009. 3rd Annual Survey: 2008 - The State of Agile Development. *Available at: http://www.versionone.com/agilesurvey/* .

WILLIAMS, L. AND KESSLER, R. 2002. *Pair Programming Illuminated*. Addison-Wesley Professional.

WILLIAMS, L. AND LAYMAN, L. 2007. Lab Partners: If They're Good Enough for the Sciences, Why Aren't They Good Enough for Us? *Conference on Software Engineering Education and Training (CSEE&T '07)*.