Three AntiPractices while teaching Agile Methods[*]

Alexandre Freire, Fabio Kon, Alfredo Goldman

[1]Departamento de Ciência da Computação
Instituto de Matemática e Estatística
Universidade de São Paulo

`ale@ime.usp.br, kon@ime.usp.br, gold@ime.usp.br`

Abstract. *This article gives educators and consultants some insight into three very simple and common problems, when one tries to introduce XP to a development team or teach it in a University: the "Bootstrap", "Split Personality" and "Abandon Complex" antipractices. Bootstrap describes how teams learning XP have difficulties when starting a project with little or no code base. Split Personality describes the difficult task one has to endure when assuming both the Coach and Customer roles in an XP team. Abandon Complex describes problems that arise when a coach has to leave his/her team. We present these organizational antipatterns as antipractices we have identified while (1) teaching XP in the context of an XP laboratory course in the University of São Paulo, (2) an experience helping a start-up transition to XP and (3) another transition in a governmental project. We discuss various simple solutions to the antipatterns based on reflections from these experiences and describe concrete situations where they were effective.*

Resumo. *Este artigo apresenta um pouco de reflexão para educadores e consultores sobre três problemas simples e comuns quando tenta-se introduzir XP a uma equipe de desenvolvimento ou ensiná-la em uma Universidade: as anti-práticas "Bootstrap", "Split Personality" e "Abandon Complex". Bootstrap descreve dificuldades sofridas por equipes que estão aprendendo XP quando começam um projeto com uma base de código pequena ou inexistente. Split Personality descreve a tarefa árdua que uma pessoa deve executar ao assumir ambos os papéis de Treinador e Cliente em uma equipe que segue ao método de XP. Abandon Complex descreve o problema que surge quando um Treinador precisa abandonar sua equipe. Apresentaremos estes anti-padrões organizacionais como anti-práticas que identificamos (1) ensinando XP no contexto do curso "Laboratório de Programação eXtrema" na Universidade de São Paulo, (2) em uma experiência ajudando uma empresa* start-up *migrar para XP e (3) outra transição em um projeto governamental. Iremos discutir várias soluções simples para estes anti-padrões baseadas em reflexões sobre nossa experiência e descrever situações concretas onde as soluções foram efetivas.*

1. Introduction

While teaching XP [Beck and Andres 2004] many problems arise that have to be dealt with, mostly due to different contexts and heterogeneous teams that have to learn prac-

[*]Relatório Técnico RT-MAC-2007-03

tices and adapt them to their reality. We have had extensive teaching experience in academic and business contexts, trying different solutions to common problems by adapting practices and refactoring XP rules to fit local realities.

From these experiences, we identified three very common, simple and recurrent antipatterns. The first one deals with the time a team takes to start being productive and picking up project velocity using all of XP's practices. We show that having little or no code base to build upon can be the source of difficulties and propose concrete solutions so a team can start to work in parallel as soon as possible. The second one concerns the difficulties some teams have to deal with when the same person plays the Coach and Customer roles in an XP project. The third one deals with difficulties a team might go trough when the time comes for the Coach to leave the project.

Just as many agile practices (e.g., "Stand Up Meetings" or "Developing In Pairs") have been cataloged as organizational patterns by [Coplien and Harrison 2004], many organizational and process antipatterns have been observed as well [Brown and Thomas 2000]. Antipatterns are apparent solutions commonly applied to solving a problem, but that in fact create a bigger problem. Antipatterns propose a refactored solution to the problem. Having observed these apparent solutions in our projects and in other research papers relating experiences teaching XP [Mugridge et al. 2003, Meszaros 2004, Jackson 2004, Tomek 2002], we will define them as antipatterns and present refactored solutions that might be of use to the agile community.

We will use "AntiPractices" as a metaphor to present these organizational antipatterns, as proposed by [Kuranuki and Hiranabe 2004], having slightly refactored the pattern template used there, to allow a simple and quick exposition of this pattern language.

In the next section we will describe the context of projects where we tried different approaches to solve each antipractice. We will then introduce the antipractices in turn, presenting different contexts and countermeasures in a narrative story format, and later generalizing them in a pattern-like format, proposing possible solutions for each. Finally we will conclude recommending the use of some of the proposed solutions in specific contexts where they have been proven useful.

2. Projects and contexts - The XP Lab, Paggo and ALESP

The XP laboratory course at IME/USP has had success teaching Computer Science students how to eXtreme Program while developing real world projects [Goldman et al. 2004, Freire et al. 2004, Freire et al. 2005a]. We will look into two of the projects realized during the 4th edition of the lab.

The first project, "Colméia", built upon a Java-based Web application for managing a University library with up to hundreds of thousands books and thousands of users. Development for this project began in the previous edition of the laboratory , it had a big code base, with many automated tests. The team had six student members, a senior student that had taken the laboratory course in the previous year acted as coach. The customer was the manager of the Institute of Mathematics library and was present for weekly short meetings and monthly Planning Games. The system today counts with 31252 lines of executable code, while during this edition of the lab students wrote (or changed) 8607 lines of code implementing 12 user stories

The second project, "Cigarra" [Freire et al. 2005a], was built in a partnership with the Ministry of Culture in Brazil [Brazilian Ministry of Culture 2003], the requested system solves the need of Cultural Hotspots to distribute multimedia artifacts. The team was composed of twelve members: eleven students and a teaching assistant for the course, who also worked as a Ministry of Culture consultant responsible for the technological aspects of the Cultural Hotspots project; thus, he played both the Coach and Customer roles in the team. Since we had no code to start with, we chose to use a few existing free and open source tools and frameworks to develop the system. The system had 2995 lines of executable code in 27 classes implementing 17 user stories.

Having acted as industry consultants as well, we experimented implementing XP at Paggo [Paggo 2003], a start-up venture in the credit card business in Brazil [Freire et al. 2005b]. The first author worked there as a consultant coaching a 6 person team in XP practices, frameworks, and OO concepts. The main objective was to have an XP proficient team ready to be independent from the coach within 6 months. One of the company's founders played an in-house customer and was present daily. During the consultancy we managed to go through 12 releases, using mostly two week iterations. We produced four applications, successfully implementing 269 stories out of an original 340, of which 42 were later discarded or deemed unnecessary by the customer. From a technical point of view, we delivered 90% of wanted functionality, fully tested and free of bugs

We also helped a team working in a governmental legislative body (the House of Representatives of the São Paulo State - ALESP), transition successfully to XP, acting as consultants in this project where the team was composed of 8 people with no previous experience in XP plus one of the students that had participated in our XP lab. Their challenge was to start the development of a new system for managing human resources in the House. During the first 8 iterations of this project we spent some time training the team on the technologies and on XP, after that we decreased our level of participation in the project. The system during this time had 48517 lines of executable code delivering 106 stories.

3. Bootstrap - "We want to do XP, but can't get all practices going, we are waiting for code that doesn't exist yet"

The Bootstrap antipractice describes how teams learning XP have difficulties when starting an XP project with little or no code base, which is common when teaching XP. In our experience, developers in large XP teams tend to have a hard time to start integrating a system or even test-first when there is no code to build upon. It is hard for pairs to move people around because the pair that started coding a component is reluctant to let other pairs work on it, fearing what a new person might think of their tests or their work, or even because they want to be the ones to continue developing upon the code they started to build. Task coordination is complicated, especially for the bootstrap story (the first story a team implements) [Andrea 2001], getting the team to work in parallel is complicated since many stories depend on other stories that are not yet developed or completed.

Techniques have been proposed to help scale the bootstrap and other stories [Meszaros 2004]. However, we found that when a project begins, and little or no code exists, programmers that have no experience with XP have many difficulties to start being productive, and the team takes a while to establish a good project velocity and adopt all practices fully.

We note that this problem is simple to solve in some cases, specifically if the team is mature and has had previous experience with agile methods. Among the possible solutions one can find, (1) breaking up the bootstrap stories using storyotypes[1], (a pattern language that defines different types of stories, some which are easier to bootstrap [Meszaros 2004]) or by creating specific tasks to focus on setting up development infrastructure or researching new technologies, (2) building upon an existing free and open source code base, (3) developing bootstrap code with a pair or smaller team using TDD. When the team is learning XP, and especially if they are also learning how to test and refactor, we propose that some of the rules of XP be relaxed, such as allowing integration of untested code for a short period of time, just to bootstrap the code base, so that everyone can then work in parallel. We know this goes against XP values, but in exceptional cases, like the ones we'll observe, we think that this relaxation can be beneficial.

3.1. Cigarra

The Coach was having trouble with the team developing "Cigarra", he was under pressure from his team and the course professor: they were the only team in the XP lab that had to push back the deadline for their first release so that they could show the Customer a minimally acceptable system.

The coach observed that two developers waited around for others to work so that they "had something to do". The story they had accepted during the planning game was dependent on another pair finishing a story before they could proceed.

Two other developers were reluctant to integrate their code; they had finished their story, but were not convinced they had enough tests. As they had little testing experience they were afraid of committing their code until they were sure they had good tests and all of them passed.

3.1.1. Action

The coach saw that action had to be taken so the team would remain motivated; they had to deliver their first release soon. Talking with the developers, he decided to split stories into development and testing tasks, and to refactor the bootstrap story written by consultants for the Ministry into smaller stories using storyotypes. We found that many stories, of the *variation* or *new business rule* storyotypes, depended on other stories, of the *new functionality* or *infrastructure* storyotypes, being completed, we prioritized the second set, so that code would exist to be worked on.

The coach also created research tasks related to free and open source software and frameworks that were going to be used in the project. These tasks were aimed at testing functionality we planned on using, or even code we wanted to refactor into our application. Some of the developers mention that they did not see the point in writing

[1]Storyotypes define four different types of concrete stories, which (1) help set up the infrastructure, (2) create new functionality in the system, (3) alter existing functionality, and (4) create new business rules.

tests for 3rd party code, but after some negotiation agreed that it was better to do that than just waiting around for dependencies to clear up with no work to do.

A temporary rule was agreed upon, that after each lecture period, everyone would commit their code, even it was not fully tested. We made it clear that this was an exceptional agreement, valid for just enough time to get a code base with which everyone could work in parallel. After that, continuous integration would depend on passing tests.

3.1.2. After effect

The decisions seemed to work. No one would just wait for others to finish stories to work on dependencies. The team was glad that new test tasks where created. The developers that were afraid of committing their code with few tests, now felt comfortable with that, and finally integrated their code into the repository, allowing other developers to start working. They also had an opportunity to try test-first programming, committing their tests before any code was created. It helped to have the coach review the code and reassure that their tests were good, and to have the rest of the team to help develop the code to pass tests, not necessarily working with the same pairs that created the tests. They were proud to learn how to test effectively.

Confidence on code quality rose and the team managed to deliver their first release. As a side-effect from testing the free and open source frameworks and software, the team was confident on the choices and grew knowledgeable on their use. Team members also saw a great opportunity to give back to the open source community, submitting patches to projects with the test-suites we created.

3.2. ALESP

The team working in the ALESP (São Paulo House of Representatives) project had more programming experience, they were new to XP but were eager to show they could code. The Coach was worried about the bootstrap antipractice, feeling that, since they were working with a large (8 person) team, if people were not able to start working in parallel right away they would loose motivation in embracing XP.

3.2.1. Action

The Coach decided to start the transition by holding an agile design session with the team. During this meeting, with plenty of white board space, everyone agreed upon the basic overall architecture and a common view of the business model. We knew that, since we were adopting an agile method, the architecture would change in the subsequent months, but the team felt much more comfortable having some concrete models to talk about. Then, we had one of the consulters, a student that had attended the Xp Laboratory, pair with another developer and, working throughout the night, code all of the classes related to the business model using TDD, so the team would have something to work on in the following day.

3.2.2. After effect

When the team came in the next day, many classes were already in the repository, and the team was able to start working in parallel in stories related to new business rule and variation storyotypes.

3.3. Paggo

At Paggo, the customer had a small project he would like to use as a test for introducing XP. One of the developers did not want to pair program, and wanted to code all his stories, only integrating the code when he finished everything. Two other team members had never written tests and delayed committing their stories, holding back others. The coach was also worried about the quality of the code being produced and resistance from some team members to embrace change. Again, we saw that some team members were too comfortable in the position of waiting for others to finish their work so they could start working.

Developers were frustrated that progress was slow and were conscious that not all practices were being followed throughly; “we want to do full XP, but we are slow because we don’t have enough code, we have to wait for others to finish their stories so that we can proceed; because we are new to XP, some are reluctant to integrate their code, and we can’t start working in parallel”.

3.3.1. Action

The coach reached an agreement with the customer: they decided that the first application would be treated as a prototype, its main objective would be to teach XP practices and would probably be discarded after it was finished. We explained this to the team and got them to focus on improving their practices. We wrote many tests and team members were glad to have a chance to explore possibilities. They knew that even tough some tests did not seem good enough, this would not be a problem because the application was not critical. The programmer that did not want to do pair programming was convinced to try it as an experiment, and a commit rule was created. All code would be committed twice a day, once before lunch and once before leaving the office, even if everything was not tested; again, this rule had an expiration date, after which code could only be integrated if all tests passed.

3.3.2. After effect

Developers managed to incorporate all XP practices after the first prototype project. They evolved their testing and continuous integration skills. Some were conscious that many tests written during the prototype project were not necessary and code quality was not very high, but the customer was comfortable with this. When the project was completed, we found that it was good enough to pass the customer’s acceptance tests, and it was even put into production.

3.4. Crystalize

We named this AntiPractice “Bootstrap” and have suggestions for avoiding trouble in the future. Causes for the problem are directly related to the size of the team, available code base upon starting the project and developers lack of fluency in XP. When starting a new project we now try these three alternative approaches: (1) splitting large stories via storyotypes and allowing a pair or a small subset of the team to write the base business model classes overnight with TDD, so that everyone can start working in parallel in *new business rule* or *variation* storyotypes, (2) building upon an existing free and open source code base to allow the team to strengthen other techniques, such as testing, and (3) allowing a short period of time, where the team can integrate code that is not completely tested. The third option is a solution to pick up speed bootstrapping the project code base, we feel that this no longer breaks XP rules, as long as testing is done early enough.

We have had success with allowing a pair to work overnight using test-driven development, bootstrapping the code base for the rest of the team, however, this is not very consistent with XP values, and should be considered carefully. It might be best to allow a smaller subset of the team (from 2 to 4 developers) to work on a design session and a metaphor with all of the team, and then code the base business model classes quickly.

Using storyotypes to split large stories in the first iteration, creating tasks for setting up continuous integration, development environments, writing build scripts, and researching technologies (especially if looking for free and open source projects to build upon) so as to occupy all of the team even if it is not yet possible for everyone to write code, are good countermeasures for this antipractice of starting XP learning projects with no code base. See Table 1 for a summary.

4. Split Personality - “Who am I? Coach or Customer?”

Studies point out that one of the challenges of teaching XP is practicing on-site customer correctly [Mugridge et al. 2003, Tomek 2002]. The Split Personality antipractice describes the difficult task one has to endure when assuming both the Coach and Customer roles in an XP project, this antipractice should not be observed in *vanilla* XP environments, however it is very common solution applied in academic contexts. Having the same person act as Customer and Coach may lead to schizophrenic results, it confuses the team and makes it hard to distinguish business and development forces in day-to-day activities and especially in the planning game.

It is hard for developers to establish when one is guiding the team, and making sure everyone follows the rules of the game, as a Coach; or when he/she is trying to request stories, validate them, or give clear feedback about the requirements, acting as a Customer. It is also difficult for this person to avoid introducing a Customer’s concerns into her/his coaching duties, and vice versa.

It has been reported that both the Customer role [Martin and Noble 2004] and the Coach role [Hedin et al. 2003] are complex, challenging, and time consuming, they should not overload a single person.

When trying to address this issue we will discuss different solutions such as using someone from the team as a Customer Proxy[2], or, when it’s not possible to get another

[2]A person responsible for interacting with the customer regularly and representing his/her needs for the

Table 1. Bootstrap

Name	Bootstrap (We want to do XP, but can't get all practices going, we are waiting for code that doesn't exist yet!)
Background	The project is starting, the bootstrap story is large, there is little or no code to build upon, the team does not have much experience in XP.
Symptom	Some members are not productive because they are waiting for other stories upon which their story depends. Others delay committing their code fearing it is not good enough or tests are not comprehensive.
Cause	Team has little experience with XP. There is not enough code to coordinate stories in parallel. Continuous Integration is not working effectively.
Ideal	Having a good code base to build upon so that no one is waiting for dependencies and pairs can work in parallel. Everyone commits working and tested code frequently.
Refactored Solution	Allow the team to commit code with no tests for a short period of time. Build upon existing free and open source code. Use storyotypes to divide large stories, create stories aimed at strengthening weak practices, and tasks aimed at setting up a development environment. Allow a small subset of the team to create a code base with the business model classes overnight, so the next morning the rest of the team can start working focusing on stories that follow the new business rule and variation storyotypes.

person, something as simple as clearly distinguishing when one is acting as Coach or Customer by using a hat.

Straight forward solutions are proposed for controlled contexts. In the XP lab, the ideal practice is to simply have coaches for the whole period, using graduate students or senior students [Goldman et al. 2004] and a real Customer that can at least participate in weekly meetings.

4.1. Story

The coach for the Cigarra project, was having trouble addressing the Customer role without taking into account his Coaching desires. It was not clear to developers when he expressed Customer's concerns, or when he was giving advice as a Coach. Even for himself it was not clear when he decided something because he knew the Customer wanted it, or because he saw that it would be best to Coach the team. Difficulties were augmented by the fact that the team was large (twelve people), and that it was difficult to communicate clearly to everyone what the Customer intended and what the Coach wanted.

During the planning game of the exploratory phase, the team tried to convince the Coach that it was best to code spikes to try out different peer-to-peer technologies such as BitTorrent, JXTA, or CORBA. The team wanted to test all of these interesting technologies and the Coach thought it might be a good opportunity to allow the team to explore possibilities. However, he was confused: was he representing his customer or giving a coach's advice? Could he do both at the same time? In retrospective, this choice wasted precious time, added difficulty to bootstrap the project, and made the team change the date for their first release with consent from the customer (who was not very happy). He now believes that he, as a Coach, should have backed himself as a Customer (knowing that BitTorrent was by far the best choice and not wanting to waste precious time with an exploration that would lead to a foreseeable result) instead of going along with the team.

4.2. Action

The Coach decided to try different approaches with his team. He invited other Ministry consultants to come act as proxy customers during Planing Games and to run acceptance tests when a release was delivered. He also started bringing a hat and some gadgets to the lab. He would put on the hat when he wanted to act as a customer, making it clear to the team when he was addressing business rules from the client's perspective, or what part he was playing during a Planing Game. He even managed to stage conversations between the Coach and the Customer, addressing his multiple personalities with gadgets, to both organize things in his mind and communicate more clearly with the team. Figure 1 shows the use of a hat by a split personality coach-customer.

4.3. After effect

Using proxy customers other than himself and a hat when he had no other choice, really made the difference for the Coach. He found that at important moments, such as acceptance testing of a release, it was very valuable to have a proxy customer, so that he could be completely concentrated on his coaching activities and the team would feel that they were facing someone that would really benefit from the system they were developing.

team.

Figure 1. Split Personality

When a proxy could not be present, he found it really useful to use a hat and other gadgets acting both as coach and customer. He could understand what he had to do better, and it was easier to keep the coach locked away in his mind when he had the customer hat on. It was also fun for the team, and added a healthy schizophrenic dynamic to planing games and stand up meetings.

4.4. Crystalize

We called this antipractice "Split Personality". Having to act both as Customer and Coach is stressful and can be quite confusing to developers and the person in question. This problem is not that hard to solve if one has courage and discipline, or if the team is small. When possible, using proxy, or real customers, can be of great help and alleviates load.

When all else fails and the person coaching has to act as Customer, using a hat and gadgets to clearly distinguish the Coach from the Customer can be a good humored way to carry this burden, and it was very useful for us, helping the person clear his/her mind into acting the appropriate way. These are proven solutions to the common antipractice of having the same person act as Coach and Customer. See table 2 for a summary.

5. Abandon Complex - "The Coach left us! Who is the Coach?"

In the natural course of a Coach's job there comes a time when the team is comfortable with XP and he/she can leave them to follow their own path. When this happens, it is advisable not to overload someone (possibly an experienced developer) with coaching duties as well. Even in a controlled context, such as the XP lab, students can drop out of courses, trouble can come up if the Coach has to leave the team. More often than not, the most experienced developer does not make the best choice for coach. We propose a refactored solution to this antipractice, having one or more Champions of the

Table 2. Split Personality

Name	Split Personality (Am I the coach, or the customer?)
Background	One person is overloaded with too much responsibility, possibly playing both Customer and Coach.
Symptom	The team is confused, they can't make sure when they are talking to the Customer or to the Coach. The person tackling more than one role is overwhelmed and stressed, not knowing how to separate his/her responsibilities clearly.
Cause	Being a Customer is a complex task with many challenges, so is being a Coach. Having to do both things can drive anyone insane.
Ideal	Having a real customer and a real coach.
Refactored Solution	Use proxy customers when possible. When all else fails, clearly identify if you are the Customer or Coach, using a hat and other gadgets.

Court[3] [Jackson 2004] act as coach, and rotating Champions using the Coach Of The Week practice [Freire et al. 2005b], or even something as simple as the new coach Accepting Responsibility. Other roles in XP teams have been discussed in the literature by [Dubinsky and Hazzan 2006].

5.1. Colméia

In the library team, the senior student that was coaching the team, dropped out of the lab to pursue a trainee opportunity. The team was concerned, for they understood the importance of the coach in keeping everyone aware of the XP rules and in conducting Planning Games with the Customer. Some of the developers, though, were enthusiastic about XP and did not want their team to fall behind.

5.1.1. Action

Developers decided that someone would be elected as a Champion of the Court [Jackson 2004]. One of the developers, was eager to act as Coach; he was elected by his peers to conduct discussions and make decisions when they had to be made. He was comfortable with the position and made it clear that input from others was desirable.

5.1.2. After Effect

The new Coach did great. Since by then most developers were comfortable with XP practices, he could count with their help most of the time. Discussions were democratic, but when decisions had to be made, the team was relieved the Coach clearly had the role to make the consensual solution come out. He also made sure to spread knowledge, with the help of experienced members, getting them to pair with weaker developers and helping ensure everyone followed XP practices.

[3]Persons who have experience with agile methods and drive the XP process from within the team.

5.2. ALESP

After a 8 iterations coaching the ALESP team, we decided that it was time for them to be on their own. The team elected a Champion of the Court, the result was that the most senior developer would assume Coaching duties, he was a bit reluctant but agreed to give it a try.

The next week did not go well, the Coach was depressed and did not manage to give clear guidance to the team. He would not decide on anything, and would simply go along with suggestions from the team, even if they were contradictory.

5.2.1. Action

After a week without any productivity, the team called us to solve the problem. The Coach admitted that he was not doing a good job. We then tried to see if anyone on the team really wanted the responsibility. One of the developers did, and he was placed as the new Coach.

5.2.2. After Effect

The new coach, although not as experienced technically, tried to do a good job, communicating with the team and making decisions when he had to.

5.3. Paggo

At Paggo, near the sixth month using XP, the consultant acting as Coach and the Customer decided it would be best that the first no longer remain on the team. Some of the developers, and the Customer himself, had embraced change, enforcing values and practices when the Coach could not be around. However they did not know who could take on the coach's responsibilities, how to proceed without overloading anyone?

5.3.1. Action

The customer was worried about what would happen to the team after the coach left. Who would become the coach? When approaching the end of our period working there, we suggested a new practice, Coach of the Week [Freire et al. 2005b], where a different champion of the court would be elected each week to act as Coach.

5.3.2. After Effect

The first elected Coach of the Week performed his coaching duties very well, conducting stand up meetings and making sure everyone followed the practices. In the second week, a new developer assumed as champion, to much relief from the previous Coach, who could get a break from being overloaded by his developer activities and coaching duties. After a few weeks, everyone had been Coach at least once, and the customer was convinced that this practice would be beneficial in always having someone acting as Coach but, at the same time, not stressing any single person by rotating the responsibility weekly.

This practice also ensured the customer that if anyone on his team left, he/she could quickly hire a new developer and get him to work with XP, and even come to share coaching duties. It has been observed [Jackson 2004] that having only one Champion of the Court is not as effective as when everyone in the team becomes a Champion. The Coach of the Week practice enforces this, as everyone has a chance to realize that XP is a methodology that requires discipline and can help to drive the process, by taking upon themselves the responsibility of making sure the team is following the rules of the game. By conducting stand up meetings and planning games this person realizes that the customer's needs should be negotiated with the team, and that it is important to have a very clear idea of what she/he wants. By helping the tracker to display metrics, the coach comes in greater contact with the team reality, and when he/she changes with the next Coach of the Week she/he has a clear understanding of his/her part in getting the whole team to function well. This helps ensuring that everyone is working as they should even without being the official coach that week.

5.4. Crystallize

Having a Champion of the Court assume coaching duties can help, especially if adopting the Coach of the Week practice, permitting the whole team to get involved in driving the XP process, and making sure no one is overloaded.

However, we would not advise that everyone on the team act as coach if the team has weaker developers, or ones not as familiar with agile methodologies, in this case, it might be better to rotate the coach role only between more eXPerienced developers. Also, it is important that the Champion really accept responsibility for her/his new coaching duties. See table 3 for a summary.

Table 3. Abandon Complex

Name	Abandon Complex (The coach left us! Who is the Coach?)
Background	The Coach has to leave the team, someone has to take his/her role.
Symptom	The team is lost. The person chosen to be coach is overwhelmed and stressed.
Cause	Being a Coach is a complex task with many challenges. The most experienced developer technically might not be the best choice for the new Coach.
Ideal	The coach does not have to leave.
Refactored Solution	Have a Champion of the Court accept to take the coach's responsibilities, rotate the champion by practicing Coach of the Week.

6. Summary and Conclusions

We have identified three organizational antipatterns that are common and recurrent both in industrial and academic environments based on our eXPeriences in these contexts. We have presented "Bootstrap", "Split Personality" and "Abandon Complex" in the form of a small antipattern language and discussed different solutions to these antipatterns based upon reflections from our experience.

Bootstrap addresses the issue of a team that starts to learn XP in a project with little or no code base. The team is new to XP and needs to be quickly productive. When starting a project from scratch, we have shown that allowing a small subset of the team to bootstrap the code base with the business model classes and then focusing on *new business rules* or *variation* storyotypes, is a simple solution to bootstrap. Using free and open source software as a initial code base is also a solution that can help teams strengthen other practices and techniques, such as testing. When the team is learning XP and other techniques, using storyotypes to split up the bootstrap story, and allowing code to be committed without complete test coverage, only during a short period of time, is a solution to bootstrap.

Split Personality addresses the issue of a person on the team being overloaded with the roles of Coach and Customer. When there is the possibility, one should use one or more Customer Proxies or a real Customer. When everything else fails, rely on the simple solution of using a hat or gadgets to distinguish clearly when one is acting as Coach or as Customer.

Abandon Complex describes troubles a team might face when the Coach has to leave. Instead of electing the most experienced developer to act as Coach, the best solution is have Champion of the Court accept responsibility, or also practice Coach of the Week.

We believe the proposed solutions will be valuable to the community. In our ongoing work, we are looking for more antipractices and their refactored solutions as to make the adoption of agile methods easier in a wider variety of contexts.

References

Andrea, J. (2001). Managing the Bootstrap Story in an XP Project. In *Proceedings of XP 2001*, North Carolina,.

Beck, K. and Andres, C. (2004). *Extreme Programming Explained: Embrace Change, 2nd Edition*. Addison-Wesley, 2 edition.

Brazilian Ministry of Culture (2003). Digital culture department. `http://www.cultura.gov.br/foruns_de_cultura/cultura_digital/index.html`.

Brown, W.J., H. M. and Thomas, S. (2000). *AntiPatterns in Project Management,*. John Wiley & Sons.

Coplien, J. and Harrison, N. (2004). *Organizational Patterns of Agile Software Development*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA.

Dubinsky, Y. and Hazzan, O. (2006). Using a role scheme to derive software project metrics. *Journal of Systems Architecture*, 52:693–699.

Freire, A., Gatto, F., and Kon, F. (2005a). Cigarra-A Peer-to-Peer Cultural Grid. In *Anais do 6o Workshop sobre Software Livre (WSL 2005)*, pages 177–183.

Freire, A., Goldman, A., Ferreira, C. E., Asmussen, C., and Kon, F. (2004). Mico - university schedule planner. In *Anais do 5o Workshop sobre Software Livre (WSL 2004)*, pages 147–150, Porto Alegre.

Freire, A., Kon, F., and Torteli, C. (2005b). Xp south of the equator: An experience implementing xp in brazil. In *Proceedings of the XP 2005 Conference*, volume 3556 of *Lecture Notes on Computer Science*, pages 10–18. Springer.

Goldman, A., Kon, F., Silva, P. J. S., and Yoder, J. W. (2004). Being Extreme in the Classroom: Experiences Teaching XP. *Journal of the Brazilian Computer Society*, 10(2):1–17.

Hedin, G., Bendix, L., and Magnusson, B. (2003). Coaching Coaches. In *Proceedings of 4th International Conference on Extreme Programming and Agile Processes in Software Engineering (XP 2003)*, volume 2675, pages 154–160. Springer.

Jackson, A., e. a. (2004). Behind the Rules: XP Experiences. In *Proceedings of the 2004 Agile Development Conference*, Salt Lake City.

Kuranuki, Y. and Hiranabe, K. (2004). AntiPractices: AntiPatterns for XP Practices. In *Proceedings of the 2004 Agile Development Conference*, Salt Lake City.

Martin, A., R. B. and Noble, J. (2004). The XP Customer Role in Practice: Three Studies. In *Proceedings of the 2004 Agile Development Conference*, Salt Lake City.

Meszaros, J. (2004). Using Storyotypes to Split Bloated XP Stories. In *Proceedings of the XP/Agile Universe 2004*, volume 3134, pages 73–80, North Carolina,. Springer.

Mugridge, R., MacDonald, B., Roop, P., and Tempero, E. (2003). Five Challenges in Teaching XP. In *Proceedings of 4th International Conference on Extreme Programming and Agile Processes in Software Engineering (XP 2003)*, volume 2675, pages 406–409. Springer.

Paggo (2003). Paggo. `http://www.paggo.com.br`.

Tomek, I. (2002). What i Learned Teaching XP. In *Proceedings of the 2002 OOPSLA Educators Symposium*, Seattle.