Increasing Learning in an Agile Environment: Lessons Learned in an Agile Team

Mauricio Finavaro Aniche, Guilherme de Azevedo Silveira
Caelum Learning and Innovation
São Paulo - Brazil
{*mauricio.aniche, guilherme.silveira*}*@caelum.com.br*

***Abstract*—Learning is an important part of the software development process. There are many advantages for developers willing to learn: increased internal and external quality of the produced software, and a reduced learning curve as beginners become high-skilled developers much faster than usual. However, learning is not taken seriously by many teams. This paper shows how to build a learning environment by doing some well-known practices, such as Book Club, Brown Bags, Dojo sessions, Pair Programming, Open Spaces, etc. It also presents some adaptations we developed in these techniques in order to improve their effects in our work environment.**

***Keywords*-agile; learning; learning practices; brown bags; dojo sessions; pair programming; blogging.**

I. Introduction

Agile is based on evolution and adaptation. Learning is the basis of these two. Therefore, agile and learning are highly related. The Agile Manifesto itself says that teams should have the environment and support they need [1]. In the software development world, where new techniques and technologies emerge frequently, a work environment that provides developers a way to keep on learning and improving their development skills has a technical advantage upon others.

Building a learning environment is not easy: it requires time and motivation. However, the benefits are clear, such as software with higher quality, continuous improvements, new team members achieve a higher productivity level faster, and so on.

We work for Caelum Learning and Innovation, a Brazilian company focused in training, innovation and consultancy. In the last 3 years, the team's knowledge increased in both technical and social aspects. Everyone, from interns to more experienced co-workers, ended up as highly-skilled developers, using and understanding the recommended best practices.

The team practices many techniques to increase learning, such as Book Club, Brown Bags, Dojo sessions, Pair Programming, Open Spaces, etc. All of them have their effects in the learning process. This paper presents the techniques used to enhance learning in this work environment, and the adaptations done in all of them in order to try to maximize the benefits. Also, researchers conducted interviews with some members of the team, in order to validate the findings.

II. The Company and the Team

Caelum Learning and Innovation is a company currently focused in the Java, Ruby and Agile environment. It has trained more than 16000 students in the last 7 years, and student feedback were always positive. The company also develops software and give consultancy for external customers, but it's main goal is to improve the development quality of the industry.

The company has a strong connection with the development community, creating and actively taking part in both Brazilian's largest architecture and Java forums. At the same time, it continuously contributes with open source projects, such as XStream [11], Restfulie [3], VRaptor [2], and dozens of others.

Caelum has 35 developers, from interns to more experienced developers. 10 of them are located in our offices in different cities. They all have different tasks during their daily work: developing software for internal and external customers, contributing to open source projects, educating students, or writing learning material. In average, a developer will spend 40% of his time teaching, working close to the students.

Caelum has also been involved with a few important events on the Brazilian market. It has brought QCon to Brazil in 2010 and will do the same in 2011. Together with QCon, Caelum's other events try to bring some of the best current speakers on software development from around the world to share their knowledge with the local community and also select the most important

Part of each developer's time is spent studying. It is a requirement in order to create the training material for a new course, prepare himself to teach a course for the fist time, or even to dig deeper into something that might be important for the market in the near feature. One is free to study in anyway s/he feels more productive, but its fundamental that he gain as much real life experience with that technology as possible.

Due to the training schedule, it is also common to have some time without any scheduled tasks. The developers are

expected to actively manage their own time, studies, and growth within the company. No one will complain about the lack of productivity in one single day, or whether the time spent on websites today was counter productive. Instead, they are expected to execute a high quality job with excellent gradings and approval from the clients. This requires a level of maturity from the developers that can be easily seen when asking someone about their following tasks: none and everything.

It might be dangerous to depend on a developer's maturity in order to be productive, as not everyone takes their tasks as an important job. Since the first day at Caelum, the developer knows the company will offer the opportunity to make the difference on teaching or developing every day, and it is up to him to do it or not - not up to a manager.

The most common teaching method for software development is still a lecture. At Caelum's environment and when with its clients, teaching means educating, helping a student to develop the ideas on his own. The teacher's goal is not to make the student memorize APIs, rules, patterns or practices, but to be able to understand the problems and construct the solutions on their own. *"It is important that teachers stop being mere conferencists"*, according to Piaget [9]. All these ideas matters because, besides teaching outside students, employees teach each other.

III. Interview

The main goal of the interview was to find out how people have learned in their work environment. Developers that started as interns or were beginners at software development and, in the company's perception, rapidly evolved, were interviewed.

Researchers interviewed 7 different developers from the team. Each interview took between 30 to 40 minutes. The questions are listed below:

1) What do you like in Caelum?
2) How would you evaluate your own learning here?
3) How does that happen?
4) Do you give classes? What do you like when giving classes?
5) Do you consider yourself a good developer? What does a good developer mean for you? How do evaluate the developer you were before joining the team?
6) Besides the technical stuff, is there anything else that you learned? How did that happen?
7) Is there anything you did not learn here?
8) Do you like the company's environment? What would you improve here?
9) What will you do to keep learning?

The interview was semi-structured. Researchers could ask more questions according to the participant's replies. When asked about how they learned at Caelum, most interviewees commented about practices, so the researchers provided deeper questions in that subject.

IV. Learning Practices and Lessons Learned

The most discussed topic during the interviews was clear: the most efficient way to learn is by doing it with a peer. They all mentioned that they learn a lot by asking questions, programming, and discussing with their work colleagues. The role of Caelum is to encourage and ensure that developers communicate and learn with each other, and that knowledge is exchanged most of the time.

This section will explain what practices the team uses to enhance their learning capabilities. Most of them are well-known by agile practitioners. This section also presents lessons learned while executing them, and how the team improved those practices in order to maximize their gains.

A. Open Spaces

Learning with the peer was by far the most common topic during the interviews. The open space plays an important role in this as it allows people to be in close contact all the time.

Our room is an open space. There are tables with no partitions and no walls (Figure 1). People can sit wherever they want, next to anyone else. The developers themselves chose to have this freedom instead of their own tables or seat. As all members are together, the development room encourages making questions anytime. Anyone can answer and sometimes a debate will arise. If the discussion gets really interesting, the author of the question is asked to write a blog post on that topic. For this reason, Caelum's blog is a very popular one in Brazil [1]. Posts cover a lot of different subjects, many of which appeared in a casual discussion. This was mentioned by one of the participants: *A discussion usually happens. And then it becomes a blog post or a topic in some forum. A simple question will sometimes become something bigger.*

Everyone is encouraged to give his opinion and no one is criticized because of their ideas. It benefits even the beginners as they can ask questions or even try to join the discussion, exposing another point of view.

The problem with the open space is its noise. It has been reported by the developers and discussed every year. It was remembered by almost all interviewees during the interview. Sometimes people will talk too loud, bothering others. We proposed a Silent Thursday, just like Jason Fried's suggested in his TED Talk [6] and Neal Ford in his Productive Programmer presentation [7]. The idea is to have people learn how to moderate themselves, talking only when necessary. This is a new practice for us and we are still measuring results. Yet, developers feel very productive on this day.

B. Pair Programming

Pair programming is the technique of using two people to write code together at a single computer [4]. Almost all

[1] http://blog.caelum.com.br

Figure 1. Our work environment: an open space.

participants mentioned pair programming as a good way of learning. When a developer is working with someone more experienced, there is a tendency of absorbing knowledge from the experienced one. A participant mentioned: *"I like when a new intern starts to code with you, the same insights that you had in the beginning start to pop up into his mind. ... It is very nice to see people having the same reactions you have had two years before. It helps you noticing how much you evolved."*. Not only the knowledge but the capacity of logical reasoning and questioning the status quo of a solution are also taught from one to another during pair programming.

In addition, even experienced developers learn with not so experienced ones: *"When you pair with the interns, they often have ideas that we may not come up with, or it would even take too many time to have!"*. A positive point in our environment is that, as teachers, our coworkers will typically use their teaching skills to teach while pair programming, facilitating the process.

However, pair programming is not natural, and sometimes the practice may not be efficient. As an example, developers sometimes tend to adapt the practice for some special cases. Practicing it with three or more people, or having two people working on different tasks and a third person helping both at the same time are examples of possible variations. However, we noticed that they are not efficient at Caelum's environment. The third one often gets distracted and/or does not get enough change to interact with the development process.

Nowadays, when programming, we try to avoid having more than two people tackling the same problem. The third one does something else - some task that one can do alone. If there is no task available, one is free to join another group to pair program for a while.

Developers also tend to pair program with the same person quite often. That reduces the learning propagation as knowledge can be kept within that pair. The team proposed a Pair Programming Matrix on the wall and, after pairing, people stand up and put a mark on that board. This way, the whole company can see whether they are rotating pairs and who still did not pair with whom. The figure 2 exemplifies this matrix.

At Caelum, pair programming is not only done within the same project. Someone from one project might join another team for a couple of hours for any reason he judges necessary. Open spaces help, as developers are free to move around. It is common to see such movement to help, for example, on a field where one's knowledge is highly valuable and will save a lot of work from the team. In other situations, someone just want to escape from his current task for a short period of time. Because everyone is responsible for his own productivity results and time control, again, this freedom requires some maturity from the developers. Even interns realize soon the value of that freedom and the responsibility that it takes, usually adapting themselves fast to such environment.

For the same reason, changing projects happens very often, even if the project continues to be developed. Class schedule, traveling, events, blog posts, and any tasks with a fixed date might require a project change. This has become natural and it is unexpected to stay for a long time in the same project, but enough to mature, learn its technologies, design, architecture, domain and help improving it. In a world with constant change, it becomes harder and harder to keep developers motivated with one single task. Therefore there is no fear to move anyone in or out of a project; no project should have a truck factor of one [4].

Even though people know that, when pair programming, pilot and co-pilot should communicate all the time, sometimes the most experienced developers leads the session and ignores the co-pilot. What we frequently do is to talk to the more experienced developers in the team, and remind them to always give feedback to co-pilots. Experienced developers have a huge role in the beginners' learning process.

Although participants said that experienced developers also benefit from learning from any other developer, the team noticed that if the developers have a totally different level of experience (e.g. a very experienced with a novice), pairing might not be effective. It depends on the teaching abilities of the experienced one and how much space he leaves for the less experienced to exercise his logical and inventing process: two skills that Caelum values in its teachers. Currently, the team is putting together developers with similar levels of experience, such as an advanced with an intermediate, or an intermediate with a beginner. This way, information exchange happens more naturally and both learn faster.

C. Programming Sunday

Although not explicitly commented during the interviews, another way to encourage programming is to organize a

Figure 2. Pair Programming Matrix

"Programming Sunday". The main idea is to schedule a sunday when developers get together and implement different code for internal or open source projects. In order to incentivate this knowledge sharing opportunity which also produces code, the company pays the lunch for all participants.

Also, developers should take turns in each project. This way, developers learn with other ones (as they are rotating pairs all day long), and learn about the projects they are working on.

However, as it happens on sunday, organizers try to do it once every two or three months, and to schedule it in advance, making everything that is possible to create a fun environment. Thus, people can go and they treat it as a moment to meet friends and do what they love: code.

During the last few experiences, several projects and intentions were written on the board prior to beginning. The group then vote on which ones they want to implement during that day. Pais are formed and the work starts. After every half an hour, the group does a small retrospective on what they did and pairs change.

In our last experience, three different programming languages were used for four pairs, and it was important that there was at least one person who fully understands the language. To be productive, not just learn, it is also important that someone in the starting pairs know the code or problem they are dealing with. As pairs rotate, it seems possible to work with a pair that are not experts on the problem or the language, but those two should not be impediments for productivity.

D. Dojo Sessions

A Coding Dojo is a periodic meeting (usually weekly) organized around a programming challenge where people are encouraged to participate and share their coding skills with the audience while solving a problem. The most common Dojo format is the Randori. In this format, the participants try to solve a problem together, following TDD and Pair Programming in time-boxed rounds (usually between 5 and 7 minutes). The problems are usually simple ones, and the goal is not to solve it, but to share their knowledge, practice and learn.

At the start of a turn, the pilot writes code, while the co-pilot discusses the current and next steps. At the end of each turn, the pilot goes back to the audience, the co-pilot becomes pilot, and a new co-pilot taken from the audience joins the pair. An extra rule is that discussions and suggestions should only be given when the tests are passing [5].

The team was used to do a lot of Randori sessions. However, there were many dropouts in some sessions. Participants also mentioned that during the interviews and also as a feedback from the last meetings.

During our retrospectives, people mentioned that they do not feel like they were evolving, because steps were too small. One of the possible causes is that the group is used to learn in a faster pace, and sticking to such small baby steps that most dojo practices insists on, will soon become bore-some.

What is currently working for the team is a Kata format, in which a developer prepare a problem and solution in advance (or a new practice, a new technology and so on), and presents the same steps he took to get to his final solution - showing every thought that came through his mind, exposing his thinking process. The discussion over it tends to be much more useful. Also, in the last retrospective, the team suggested to define a "goal" to the Dojo session: learning the language, improving OO design, etc. This way, if the goal is to learn the language, people will not focus on TDD, for example.

In the last couple of Dojo sessions, the language used was Python, and the problem solved was somehow more complex than the usual trivial problems chosen for Randori sessions. As most of the developers are already fast learners, they may got bored if the baby steps are too small, and therefore the steps taken were a little more fast paced.

As usual, the red time, when no one but the pair is supposed to talk, was considered boring as most developers will know how to proceed and want to share their thoughts.

There is another dojo format called Kake. In this format, there are two or more pairs working on the same problem, but using a different language. developers work on all of them. The benefits of the Kake format in our team is still a mystery. Sometimes it goes well, sometimes it does not. Although it is fun, people still have the feeling that they did not learn much.

E. Brown Bag

Brown Bags [2] are trainings or seminars that may occur during lunch time. Its name comes from those bags that food is generally delivered. The idea of this practice is to get the team together during 1 hour or so, and use it to exchange information. While some member of team is presenting something, the others are eating and discussing about it. Figures 3 and 4 show our team during one brown bag.

The practice appeared to be very useful. People are very motivated to talk during lunch. However, the session was not taking more than 15 minutes. What we are currently doing is to encourage questions during the presentation. All subjects presented at the session are deeply debated by everyone. In our experience, the presentation may take few minutes (5 or 10 minutes) and the discussion will always flow for almost an hour.

In the retrospective, people mentioned that one advantage is that all team members participate in the discussion. Everyone can ask questions or raise another discussion point.

We are also avoiding doing it in our work place as it is easy to get distracted; if some discussion may not be interesting for a specific developer, s/he opens the notebook and start working. Food may also distract developers. If people starts eating from the very beginning of the presentation, the noise may bother the presenter. We start eating after the presenter is finishing his presentation, and people are free to ask.

Brown bags were mentioned by the team during the interviews and the year retrospective. Because of that, the team increased the frequency of brown bags: they are doing 2 per month.

Figure 3. Team members eating and discussing.

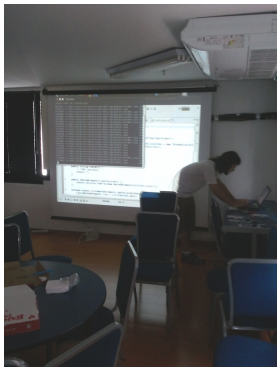

Figure 4. Team member presenting some idea to the team.

In our last brown bag, the developers chose one of their current projects, one that historically had no tests and involved smelly code, and did a one-hour refactoring of the code. Although no tests were developed, several refactoring practices came up, and many developers gave suggestions on what to do at the same time. The resulting code was amazing and a lot of developers get to know deeper the code and the domain of a new project. This shows that brown bags can be done in any format, depending only on the presenter's creativity.

F. Internal Discussion List

Caelum has as internal discussion list which allows anyone to research and share any idea even when not physically present. All interviewees mentioned that the level of the discussions that happen in there are high. Some of them even said that, when an intern, they did not understand most part of the discussions and they knew they evolved because they started to participate on the discussions.

A discussion list via e-mail turned out to be useful as people can research and think before writing their opinion. It is common to see academic papers or books being referenced during these discussion. A participant of the interview mentioned that these references are useful, and he reads all of them.

The other big advantage that a mailing list has is that it allows developers who are currently out on clients to take part of the discussions or get help.

Moreover, important developers outside the company are invited to be part of it. It makes the discussion even richer because it may help reducing the discussion bias that may appear. With Twitter's revolution in daily life, some of those discussion where commented more and more often on it, therefore we started an online forum [3] where discussions that can be shared are post and open for the public to talk.

Because of this forum and the decreasing focus on outsourcing - increase on education and teaching - the number of emails on the list has decreased, although every discussion usually goes on for a week or more.

In 2002, there was no major online forum on the Java language, so the founders of Caelum decided to create a site called GUJ (Grupo de Usuários Java or, in a free translation, Group of Java Users) [4], which during a couple of weeks was basically a place for them and their friends share their questions in Java and help each other. GUJ is currently the biggest portuguese website on Java with over 126.000 users, 1 million messages, 10.000 new messages per month and 597 thousand pages indexed by google. Java ranch's English forum has 695. Oracle's forum in all languages are only 24 times bigger using the same measurement.

Some of the early developers who joined the forum are, nowadays, big names in the software industry in Brazil. Some of them even became Caelum's teachers. Throughout the forum, one can easily see the questions and answers that those teachers did over the time and compare the amount

[2]http://en.wikipedia.org/wiki/Brown_bag_seminars

[3]http://www.tectura.com.br/

[4]http://www.guj.com.br

of knowledge acquired in, for example, one year. All the knowledge exchanged that those forums and mailing lists provides is also used by Caelum when teaching new trainees or developers, as they are asked to answer questions on GUJ so they practice their teaching skills.

G. Open Source Development

Paulo Silveira, Caelum's co-founder, started contributing to open source with Java frameworks while attending university. In 2003, VRaptor [2] was released, which became the first and most well known Java framework in Brazil, a local approach that has been even referenced as innovative by the members responsible for the Java EE specifications [8].

During those years, Caelum has realized the importance of open source in its environment, the developer's knowledge, learning and teaching abilities. Good open source code also works as a marketing feature. For all those reasons, Caelum has invested its developers time into developing for projects that do not have direct financial value.

Going through the code of other developers is a good way of learning, and open source allows one to do it. With XStream [11], for example, one of Caelum's developer has had its inflection point of productivity and software quality, also being his first real contact with good testing practices. Learning from code that others developed has been greatly boosted with the raise of GitHub [5], and nowadays it is easier than ever to share code, contribute and learn from others.

Stella [10] is an attempt to bring local issues from Brazil into programing code, such as zip code and Brazilian social security number handlers. With Restfulie [3], a hypermedia aware REST framework, Caelum's open source got international recognition. Restfulie is a good example where the company was able to get a better understanding of comparing implementations of the same framework and ideas in several different languages, always with aid from external developers.

Because all of that, developers are encourage to create or to be become part of an open source project. Open source was and will still be the catalyzer of learning and exchanging ideas with external developers, without any limits due to political decisions that a company might have, allowing the teachers to have a better, non-biased, understanding of every possible approach to a problem.

H. Blogging

The act of writing a technical article makes developers think more about a specific subject. Because of that, our team is encouraged to write about any topic they have studied. Team members sometimes even write it in pairs. The final article can be published it in his personal blog or even it the company's technical blog.

[5]http://www.github.com/

After it get published, the all team is invited to read and make comments about the findings. As it can be read by anyone outside the company too, a simple post blog becomes in a huge discussion. Because of this initiative, our blog has more than 220 posts, 3500 comments, and had 500.000 visits in 2010.

An English blog was also started in late 2009 to discuss some more advanced topics related to Caelum's research and innovation, allowing the outside developers to learn and also share ideas with the team. Currently, our blog has 65 posts and 136 comments.

Also, our team members are invited to write to Brazilian technical magazines very often. It allows the developer to have even more feedback about what he is currently studying. The founders of Caelum are also technical editors of the largest Brazilian Java magazine, while the company employees writes for both Brazilian magazines.

I. Conferences

Conferences are a good way to get in touch with cutting-edge technologies and practices. Developers are encouraged to frequent conferences. As our focus in mainly Java and Agile, conferences such as Agile, Agile Brazil, QCon and JavaOne are the targeted ones. Although the content of all presentations may not be advanced, they serve as an incentive for developers to study about that topic.

Also, all these learning techniques makes the team members to researches about different topics. Sometimes they find interesting results and, because of that, developers are also encouraged to submit presentations. As mentioned in the interviews, conferences are also used to validate ideas and receive feedback about them.

The networking during the conference is also important. Team members can find other people that are currently researching or studying the same topic. This is important to the developer and to the company: the developer finds a way to learn even more about the subject, and the company get to know a new developer that may join the team in the future.

Caelum's employees are incentivated to do talks in events all over Brazil. In 2010 there were more than 40 events where Caelum sent someone from São Paulo to present. Since 2010, Caelum also started sharing its experiences and results in international events. Those are great opportunities to seed interest in the attendees on new technologies, methodologies or practices. For Caelum, this helps spreading its image within and outside the country, while it helps the developers to become more famous within the communities.

J. Life and Work

Our society has changed and the dynamics of work and personal life too. While previous generations experienced clear separation between social life with co-workers from the one with friends, work is now an extension of one's

personal life. One's experiences out of the workplace is also an extension of work life.

Because of that, encouraging cultural experiences and any other social activities amongst co-workers helps creating a stronger team feeling where everyone feels more comfortable talking to each other. This freedom helps co-workers feel free on being critic during pair programming, not to feel shy while showing code they are not that proud of, or ask maybe foolish questions to the team.

V. Conclusion

The most mentioned way to learn was, by far, learning with a peer. Based on that, what we are currently doing is trying to maximize the opportunities in which the team get together. Also, most of interviewees commented that high-skilled developers raise high-leveled discussions. However, hiring only advanced developers is not easy. We argue that, if the learning is effective, developers will evolve in much less time than the average, and they will be able to discuss advanced topics that they were not able to.

Maximizing the possibilities a developer has to show its study is also a good idea. If a team member is researching about something, he has many opportunities to have feedback about it: a blog post or an article in some magazine, a presentation in a brown bag, a discussion in some list, a conference, and etc. We encourage them to be part of all of it.

One of the open questions in this environment is how to deal with the growing expectations from a few developers that learning is something that must be done only during work hours. If not incentivated by the company during this time, the developer might not try to improve its knowledge off work. Neither this nor the opposite are the best solutions.

Also, some practices sometimes work better for a period of time and, after that, their outcomes become less than enough for their costs. When that happens, the team should stop practicing that for a period of time. Even with those practices help developers to share learning, it is up to them to make it happen. If a practice is not being efficient enough, the team should stop doing it. A team should only use practices that work for them.

We do not claim that the discussed practices will work in all environments, neither all the time, nor with every kind of developer's group. The effectiveness of any kind of practice depends on time, environment, and the people involved. Choosing good ones and dropping the ones that are not working is what seems important in order to not simply waste time. However, all of those practices have the same principle behind: increasing the learning process with peers. This research also suggests that companies that use learning practices may repeat the experiment and do the proposed interview. That may help validate the findings in this paper.

Acknowledgment

The authors would like to thank Caelum's team members which allowed us to ask questions and interview them about their learning process.

References

[1] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland Dave Thomas. Manifesto for agile software development. http://agilemanifesto.org/, 02 2001. Last access in February, the 22th, 2011.

[2] VRaptor. http://vraptor.caelum.com.br/en. Last access in March, the 21rd, 2011.

[3] Restfulie. http://restfulie.caelum.com.br/. Last access in March, the 21rd, 2011.

[4] Beck, K. Extreme Programming Explained: Embrace Change. Addison-Wesley Professional, 2nd edition, 2004.

[5] Sato, Danilo Toshiaki; Corbucci, Hugo; Bravo, Mariana Vivian. Coding Dojo: An Environment for Learning and Sharing Agile Practices. AGILE Conference, pp. 459-464, Agile 2008, 2008.

[6] Fried, Jason. Why work doesn't happen at work. http://www.ted.com/talks/jason_fried_why_work_doesn_t_happen_at_work.html TEDxMideast. 2010. Last access in March, 20th, 2011.

[7] Ford, Neal. Productive Programmer. http://www.parleys.com/#st=5&id=2217 Devoxx. 2010. Last access in March, 20th, 2011.

[8] Sandoz, Paul. JAX-RS, Java EE 6 and the Future. http://parleys.com/#id=2096&st=5 Devoxx. 2010. Last access in March, 20th, 2011.

[9] Piaget, Jean. Où va l'éducation. Gallimard, 1988.

[10] Caelum Stella. http://stella.caelum.com.br/. Last access in March, the 3rd, 2011.

[11] XStream. http://xstream.codehaus.org/. Last access in March, the 3rd, 2011.