Book Review –The Pragmatic Programmer

To begin with, I would like to say that this book has helped me to see things differently when it comes to programing. And when I say this I mean my way of thinking, but I also realized that found myself in the characteristics of the pragmatic programer such as: indiscreet, I ask when I have unanswered questions. I try to keep up with technology and I also try to understand any problem that I might come against. It is a book that I could understand , but there were some things that I may not have understood fully.

The fact that it gives examples from real life regarding the programing, it helps for a better understanding of the text.

At the beginning the title did not speak to me, even though I knew what the word pragmatic meant, but after a further deepening of the book I realized that it meant the writing of a clean code gestionated in a more efficient way, readable not only for the persons making the code but also the persons using the code in the future.

From everything that I read from this book what I would like to talk about the three ideas that I would use every day. The concept from ‘The broken window’ from chapter one, A pragmatic Philosophy, says that a building which has broken windows that are not quickly replaced, after a while that building will have more broken windows, for it to be vandalized in the end. It is a concept that made me think if I should ignore a problem when I know something is wrong with my code. A lot of the time I tend to ignore it because I want to finish the project quicker, but I learned that at some point my code will crack. The second one is “Orthogonality” from chapter two “A pragmatic Approach”. Two lines that are orthogonal and they meet in a right angle and are in fact independent, it refers to the fact that, as long as the project is divided in more than one part that doesn’t depend on each other, but acts together, the better it is. Why? Because that is how you get better results like the growth of productivity, they can be reused and they can be combined, especially when there are bigger projects made by a team. When the components are isolated from one another, the problems that occur later are detected easier, and their behavior is easier to comprehend and to modify. I like the example of the helicopter very much. The way of piloting is non orthogonal, the commands are independent from one another. The third one and the most important one is “Refactoring” from chapter six “While you are coding”. The software refactoring is a process of restructuring of the code without changing his behavior. The author emphasizes the fact that it should be made more carefully and with more attention especially when you try refactoring not to add more functions. It is a process that I learned recently and I realized that it is a necessary one. It is a must when you finish a function, for it to be reviewed and refactoried if it is needed. Most of the time, you need to do it often. One of the author’s affirmation ,in particular, is that a code is never perfect and it should be watched from a critical point of view for it to be perfect. ‘A broken window’ and the software refactoring are two concepts that go hand in hand.

The suggestions that the author gives to the reader are everywhere in the book. Some of them are repeated maybe to emphasize their importance. The most important one is specified in the prologue through the old motto IBM “Think!”, but also in many other places. It basically means the fact that you must find a solution, no matter what. Others are: to invest more in your portfolio of knowledge, to learn a new programing language every year, read more technical books, to be aware of everything around you and many other things. Communication is another suggestion of the author that is very important. He informs the reader that if a good idea is not expressed in a efficient way then it will not be received well. Furthermore, he emphasizes that if you want to be heard, you have to listen. Be a good listener. Or when you ask a question, answer politely and also when you are asked, if you have the answer, do not hesitate to respond.

In any of the eight chapters of the book, the author suggests a solution for different situation in which the programer may find himself. I will present some that I believe are more important. The reversibility is one of them that is very well put, “Nothing is more dangerous than an idea if it’s the only one you have” by Emil-Auguste. It refers to the fact there will always exist more ways to solve a problem, you don’t have to get stuck at one. Also, when choosing the right algoritm. As the author says the fastest way is not always the most efficient one. Another solution is “Programing by coincidence”. Sometimes when we create a code it happens that some methods work, but is just a coincidence because as we move forward we may take a wrong path. It is possible, in the future, to restart everything because we reached a point where the app doesn’t work. And if we notice something wrong, we have to restart and not to ignore it. A good example that is given by the author is the story of the soldier that walked through battlefield about which he had known it was also a minefield, he walked and walked until he thought he was safe and then he was made to shreds. That is what happens with the code every day, there are always new traps, the point is not to make hasty decisions. The choosing of a good editor. The writer emphasizes the fact that is very important for the editor with which you work to be complex and full of functions. It is highly important and of high utility to know the programs in which we work. A good knowledge of the IDE can simplify the deeds that must be done. But is important for the editor to have a automatic indentation, configurable to your preferences including fonts, colors window sizes. For it to be extensible, to contain more programing languages. The better it knows the language, the better it does the job.

The strategies that the author uses to transmit the key ideas of the book are the fact that he uses quotes, examples from real life, stories, he uses exercises or challenges. All the quotes from the book are a good example and the most important part is that they are placed at the beginning of the paragraph. Even before you read the paragraph, you kind of know what it means. I wrote some quotes that I really liked and I will mention three: ‘The greatest of all weaknesses is the fear of appearing weak.’ By J. B. Bossuet, ‘The limits of language are the limits of one’s world.’ By Ludwig Wittgenstein and ‘He who hesitates is sometimes saved.’ By James Thurber. The examples from real life and the little stories help you understand better the main idea that the writer wants to send. Without them, I would not understand what the author meant. The exercises are interesting, some of them I could solve, some of them I did not know how to answer. Most of them represent the programing language Java and C++. The challenges made me think, they are a bit difficult and they need time, but they, mainly, stimulate the thinking.

When I first saw the title I thought of the word efficiency because is a synonym of the word pragmatic. I have heard the word ‘pragmatic’ in a tutorial, the man was saying how the code must be clean and easy to read. It is related, somehow, to the main subject of the book which is the evolution of the programer, even if he is an expert, a newbie, to solve all the problems that may occur and to give advice for them to be solved and also, the creating of a clean code in a more efficient and readable way.

There are many aspects of the book that I do not fully understand. I will describe one of them, how I understand it. Design by contact. From this, the only thing that I understood was that this concept was made by Bertrand Meyer, for the programing language Eiffel. It is a simple and strong method that suggests agreeing and documenting the behavior and the obligations of the methods of a project, to assure an efficient and a good functioning. From this point on, the explications lost all meaning for me. And are more other thinks that I didn't understand.

The most pleasant parts of the book for me are the explications that are given through examples from real life or the stories that already create a main base for the things that are explained later on. The most interesting story was the one called ‘Stone Soup and Boiled Frogs’. A very interesting way you can basically trick people. Another aspect that I really enjoyed was the idea that no matter how perfect the code is, you must see it from a critical point of view and to always look for the best solution that may be different everytime. One of the challenges that I liked very much was the one with four points that I had to link with the three lines. At first I did not realize, I tried a couple of times. I realized I needed a triangle, but I did not know how. And all of a sudden on the same page, I saw ‘Don’t think Outside the Box –Find the box’ and that, basically, made me realize that I could make a big triangle to fit all the four points and when I saw the answer at the end of the book, I was very happy. What I did not enjoy were the challenges and the exercises that I did not understand. I would like to add that I searched the book on the web and I found it in pdf from. There is a difference between the printed one and the pdf form, because the pdf one mentions on every white page before the next chapter ‘This page intentionally left blank’, and on the printed one it does not. I looked it up, but I could not find the meaning.

In my point of view the book is very efficient and it represents a true collection full of advice for the career as a programer.