One of the emerging techniques for managing software project is eXtreme Programming (XP), which surely is changing the way in which we develop and manage software. XP is based on four values and 12 rules that explain how to develop software in an XP-compliant mode. In this chapter, all values and rules are addressed and the XP lifecycle of a project is introduced in order to guide the reader in discovering how XP can aid maintenance by keeping maintenance costs constant over time. Before drawing conclusions, the direct impact on the maintenance process due to the adoption of XP values and rules is also exploited at the end of the chapter.

**EXTREME PROGRAMMING: A GENTLE INTRODUCTION**

One of the emerging techniques for managing software project is eXtreme Programming (Beck, 1999 and 2000). XP surely changes the way in which we develop and manage software, but in this chapter we will explore also how it can...
change the way we maintain software. The most interesting feature of eXtreme Programming is the fact it is “human oriented” (Brooks, 1995). XP considers the human-factor as the main component for steering a project towards a success story. It is important to evidence that XP is guided by programming rules, and its more interesting aspects deal with the values that are the real guides for the design, development and management processes.

The values on which XP is based are:

• **Communication**: it is the first value, since XP’s main objective is to keep the communication channel always open and to maintain a correct information flow. All the programming rules I will describe later cannot work without communication among all the people involved in the project: customers, management, and developers. One of the roles of management is to keep communication always up (Blair, 1995).

• **Simplicity**: the first and only question that you have to answer, when encountering a problem is: “What is the simplest thing that can work?” As an XP team member, you are a gambler, and the bet is: “It is better to keep it simple today and to pay a little bit tomorrow for changing it, instead of creating a very complicated module today that will be never used in the future.” In real projects with real customers, requirements change often and deeply; it is crazy to make a complex design today that have to be completely rewritten in one month. If you keep it simple initially, when the customer changes his mind or when the market evolution requires a change, you will be able to modify software at a lower cost.

• **Feedback**: continuous feedback about the project status is an added value that can drastically reduce the price that you must pay when changes have to be made. Feedback is communication among management, customers and developers, that is, direct feedback; feedback is also tracking and measuring, so that you can maintain control of the project with indirect measures on the code, on its complexity, and on the estimated time to make changes. Feedback is not only communication but also deals with problems of software measurement and then with metrics.

• **Courage**: courage is related to people, but it is always related to the way in which you develop your project. XP is like an optimization algorithm, such as simulated annealing in which you slightly modify your project in order to reach an optimal point. But when the optimum you reach is only local, you need the courage to make a strong change in order to find the global optimum. XP with its rules encourages you to re-factor when your project is stalled, but in order to re-factor in deep you need courage.

These values are the bases for the 12 XP rules. The following is a short description of these rules.
Governance of Cross-Organizational Healthcare Document Exchange through Watermarking Services and Alerts
www.igi-global.com/article/governance-cross-organizational-healthcare-document/61317?camid=4v1a