IRM PRESS
701 E. Chocolate Avenue, Suite 200, Hershey PA 17033-1240, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.irm-press.com

ITB11781

This chapter appears in the book, *Skills for Managing Rapidly Changing IT Projects*
by Fabrizio Fioravanti © 2006, Idea Group Inc.

Chapter XIII

ADPD and UML Adoption

In this book, the world of management and Agile development always suggested that we adopt the simplest technology and methodologies that can fit the project you are dealing with.

These are the questions that can arise: How can UML fit these requirements? Can UML be employed with Agile methodologies and particularly with ADPD? How many types of diagrams should be used profitably with ADPD?

This chapter will try to answer these and other questions that are asked considering UML as a tool for aiding development and management of Agile projects. Agile developers usually draw UML diagrams on a board in order to have a topic to discuss about formalized well-know common language. On the other hand, when these diagrams are transferred to a tool capable of managing UML notation, you obtain the advantage of having a documentation automatically generated that can be updated to your needs.

As introduced in Chapter XI, UML and the automated documentation that some tools can generate, starting from a set of UML diagrams, can substitute the ADPD project document. This approach to documentation compels the project manager to make some choices and decide if some parts of the project have to be documented or not. In any case, it is better that this choice be performed by the project manager, who can select the most important features to be documented among the whole project instead of compelling the reader of the documentation to cut according to his judgment.

UML in Short

This section is not an exhaustive essay on UML diagrams and their usage nor is it an introduction focused on the general concepts of UML. The main purpose of this section is to propose the author's view of UML and UML adoption, to suggest further reading for interested readers, and to give a brief introduction to tools that can be usefully employed to approach UML for the first time.

UML is the acronym for Unified Modeling Language, and, as the name suggests, it can be usefully employed to model with a (visual) language all the processes. UML tries to unify by the adoption of a set of diagrams all the possible options for modeling business processes, interaction among software models, and human description of class hierarchy or module dependencies together with fast descriptions of modules or class functionality and responsibilities.

When this section was written, UML was at its 2.0 specification, and therefore, it can be considered mature enough to be used.

In order to give the reader a clearer view of the diagrams and to introduce some free tools (or free downloadable versions of commercial tools) for UML adoption and Java development, I have decided to adopt two free tools that can be downloaded from the Internet: NetBeans 3.6 IDE for Java (www.netbeans.org) and Smart Development Environment Version 1.1 of Visual Paradigm in its community version for NetBeans and SunOne (www.visual-paradigm.com). The snapshots of this section have been obtained during the usage of such tools. The tool used in this chapter has implemented Version 1.4, and the standard UML diagrams that it offers are described in the following:

- **Use case diagram.** Use case diagram describes what a system does from the point of view of an observer that is external to the system. The focus of a use case diagram is to describe what the system has to do without specifying how the system does such functions. Use case diagrams are closely connected to XP metaphors or to agile stories, since use case describes what happens when an actor (i.e., a person or another system) interacts with the system under description.
- **Class diagram.** A class diagram's aim is to show the classes belonging to the system and their relationships. This overview of the system is not a dynamic view, since no description of what happens when two classes interact is given with the interaction link.