INTRODUCTION TO UML

Unified Modeling Language (UML) makes it possible to describe systems with words and pictures. It can be used to model a variety of systems: software systems, business systems, or any other system. Especially notable are the various graphical charts—use case diagrams with their stick figures or the widely used class diagrams. While these diagrams aren't fundamentally new, the worldwide unification of modeling languages is new with UML, which was standardized by the *Object Management Group*(OMG), an international association that promotes open standards for object-oriented applications.

Most books about UML describe it almost in its entirety. However, our experience has shown that in reality there is often a lack of time, previous knowledge, or motivation to deal with the topic with the necessary intensity. In these cases, the material can't be completely understood and put into action. This book is meant for exactly these cases. We put together those parts of UML whose application has proven to be practical. With a little effort, anybody should be able to make use of UML.

There are several reasons to use UML as a modeling language:

- The unification of terminology and the standardization of notation lead to a significant easing of communication for all parties involved. It facilitates the exchange of models between different departments or companies. Moreover, it eases the transfer of projects between project teams or project team members.
- UML grows as the requirements for modeling grow. Because UML is a powerful modeling language, you can start with the development of simple models or model complex systems in great detail. If the basic functionality of UML is not sufficient, you can extend it through the use of stereotypes.

- UML builds upon widely used and proven approaches. UML was not devised in an ivory tower but was developed mainly from real-world problems and existing modeling languages. This guarantees usability and real-life functionality.
- UML is widely supported.
- UML-based bids for software systems can be compared much more easily.

At the time this book was wrote, standardization of UML was not fully completed. However, subsequent changes to UML 2.0 will not affect the simplified approach that our book presents.

Our book was written for computer scientists and for people involved in the development process of IT systems, such as analysts, decision makers, users, and experts. It shows how, with UML, simple models of business processes and specification models can be created and read with little effort. Our experience with projects showed that:

- Often only components of a model are created.
- Most of the time the entire system is *not* modeled.
- Very little time is spent on training in modeling language and methodology.
- In short: modeling is not given much priority.

Certainly a few projects use the complete UML model appropriately. However, the bulk of all projects use UML or other modeling languages, modeling tools, and modeling methods to only a small degree, if at all. While enthusiasm and motivation are strong at the beginning of the project, modeling and documentation of the modeling results are often the first to fall victim to the increasing time pressure as the deadline approaches.

Unfortunately, we cannot change that. Considering these circumstances, we have tried to portray a much-simplified picture of UML, in order to make it possible to use UML more efficiently and appropriately with only a small investment of time.

Experience shows that mastering only a few elements of UML leads to better results than the superficial knowledge of many UML elements. So we have selected some of these elements for you—subjectively, of course. We have not even mentioned many elements of UML, and explain others in a very simplified manner. Even though that is not always how it was originally intended, it does reflect our practical experience.With its changes and extensions, UML 2.0 supports the modeling of business processes much better. The increased size of UML 2.0's vocabulary shows that it's no longer sufficient to define only certain elements of the language, but also necessary to define the use of UML in specialized fields, such as, system integration or data warehousing.

In this book we have emphasized these aspects through the use of profiles, while defining the language elements (vocabulary/terminology) of UML 2.0. We will refer to these profiles at the appropriate places throughout this book.

We have structured this book so that it can be read while working on a project. We begin with the modeling of a business system and its business processes in Chapter 3, *Modeling Business Systems*. Then we go on to specify an IT system that is to be embedded into the business system (Chapter 4,*Modeling IT Systems*) and lastly describe the integration of the IT system into its environment (Chapter 5, *Modeling for System Integration*). These three chapters are independent entities. You can read only those chapters that you require for your project. But in any case, you should first read Chapter 2, *Basic Principles and Background*, where we introduce the case study. This chapter also explains several basic terms and concepts that the following chapters build on.

We use a case study throughout this book to convey the theoretical knowledge about UML. This case study serves the sole purpose of illustrating UML, since much can be explained and understood through the use of examples rather than abstract definitions. The reader is supposed to get a 'feel' forUML. Of course it is not possible to use every diagram and every piece of documentation of the entire case study: that would have been well beyond the scope of this book and is not necessary for the comprehension of UML. The case study—passenger services at the fictional UML Airport—does not always accurately represent real passenger services: we have simplified it in parts. However, this does not have a negative effect on understanding UML.

This book does not assume any prior knowledge of UML or object-oriented programming. However, a basic understanding of the development of IT systems is expected.

At this point, we especially want to thank the friendly employees of Unique Zurich Airport, who helped us with patience and competence to understand the technical details of our case study. We thank everyone who helped with the creation and revision of this book. In particular, we want to thank the editor of our German edition, Judith Stevens-Lemoine, and her team at Galileo for their competent and friendly attention. We also found helpful the critical comments and suggestions from our readers, and our colleagues at *Integratio* and *KnowGravity*.

Source : http://sourcemaking.com/uml/introduction