Rational Rose Overview

Modern software systems:

Modern software systems are getting more complex. It is difficult to be familiar with all parts of them. To build a complex system, we begin by looking at the big picture without getting caught up in the details. Developing complex system can be generalized into three big steps:

- abstract different views of the system, building models using precise notation
- verify the models to satisfy the requirement of the system
- gradually add details to transform models into an implementation

Models:

Building models is an ideal way to portray the abstraction of complex problems by filtering out nonessential details. It promotes better understanding of requirements by visualizing abstract ideas. It leads to a cleaner designs and more maintainable systems.

Visual Models:

Visual modeling captures business processes to a graphical representation by defining the software system requirements from the user's perspective. This streamlines the design and development process. It also captures the logical software architecture independent of the programming language will be used. It promotes reusing parts of the application by creating components.

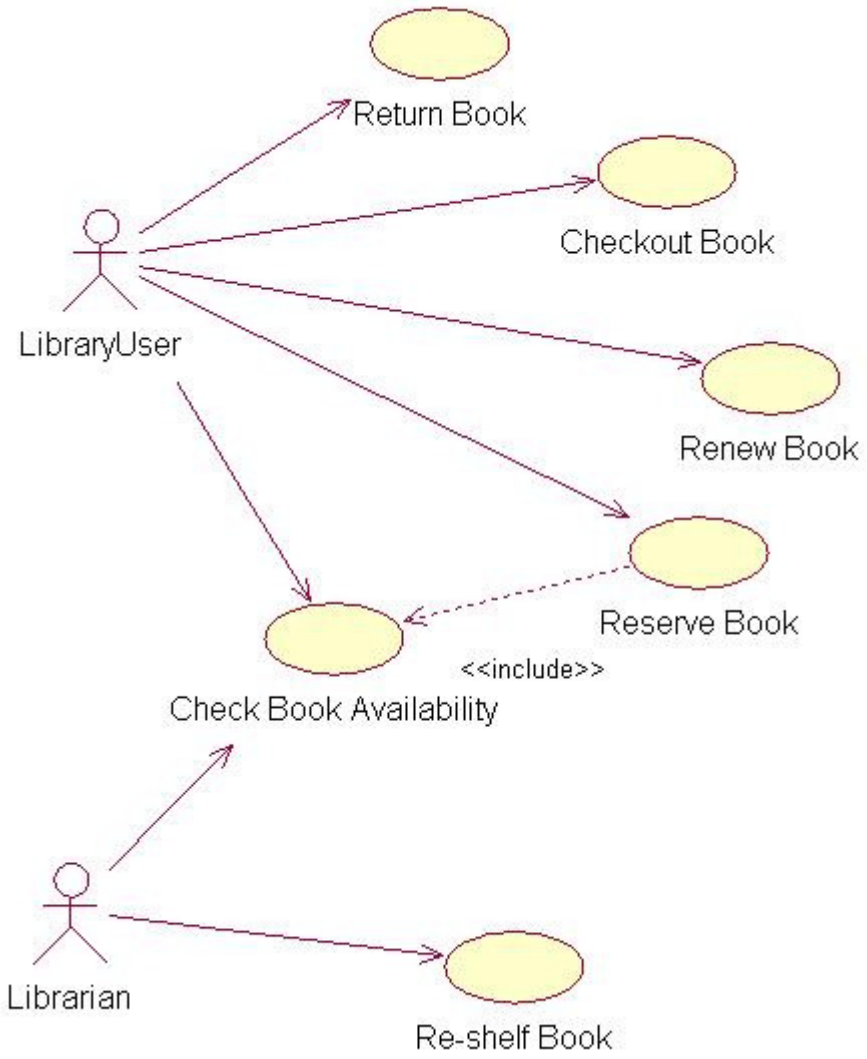

Fig. 1 A visual model depicting a library transaction system

Modeling with Rational Rose:

Rational Rose is a visual modeling software solution that can graphically depict an overview of the behavior of a system. It shows objects interaction and links between them, life history of a given class, and workflow of a business process. Rose supports UML – a communication standard. Using UML, a common vocabulary is used and thus miscommunication is minimized.

Rose captures the following system architecture:

- logical architecture: classes and relationships that represent the key abstractions
- component architecture: actual software module organization

- deployment architecture: configuration of run-time processing elements and their processes

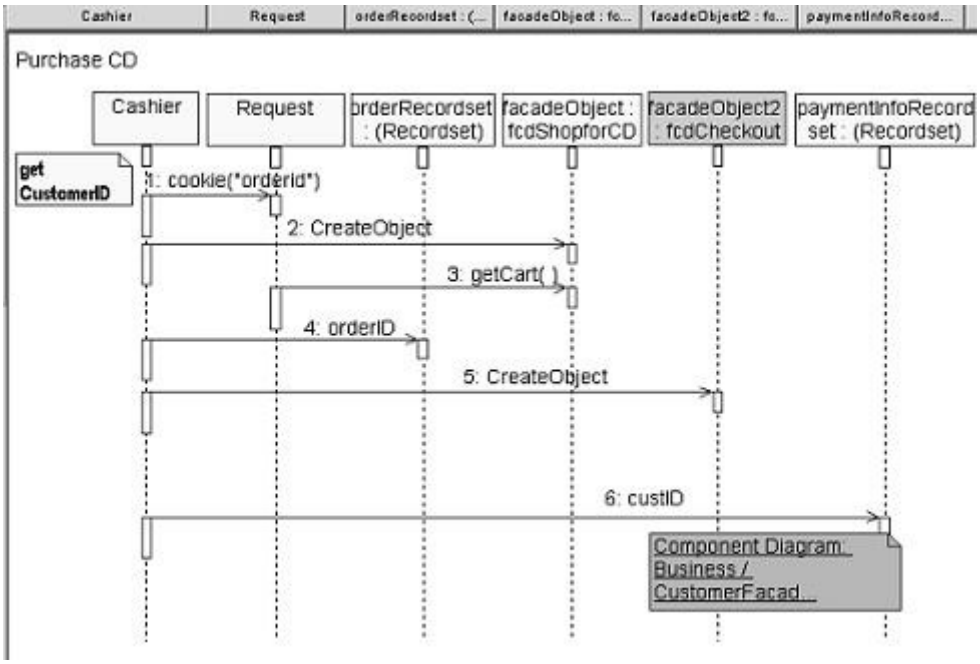

Fig. 2 Rose shows the interaction between objects

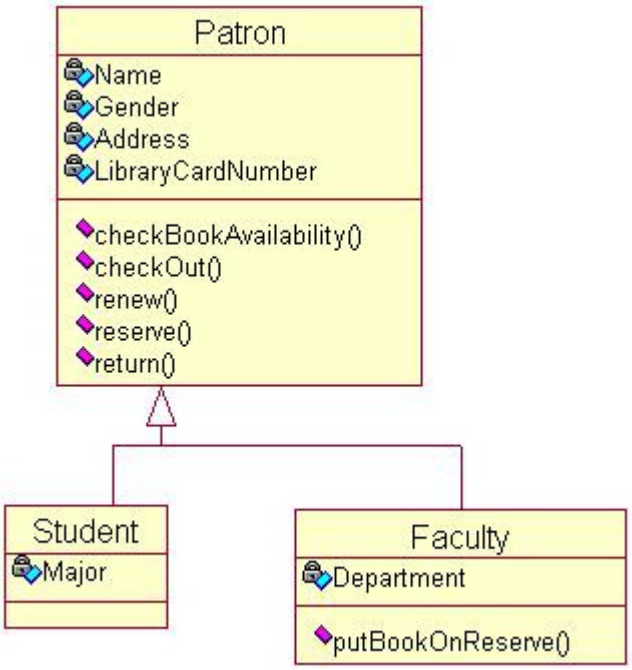

Fig. 3 Describing classes with UML