upcomillas es

upcomillas es

Advanced Computing Tools for Applied Research

Chapter 1. Introduction to software engineering

Jaime Boal Martín-Larrauri
Rafael Palacios Hielscher

Academic year 2014/2015

1

Introduction

Types of software developers

- Many people write programs
  - Amateur
    - Business people ➡ Spreadsheets
    - Scientists and engineers ➡ Simulation and data analysis tools
    - Hobbyists…
  - Professional
    - Application suites (productivity software, graphic design, IDEs…)
    - Mobile and web app developers
    - Video game developers…

What are the fundamental differences between amateur and professional software?

Amateur vs. professional software development

	Amateur software	Professional software
Number of developers	Single (or a very reduced group)	Team
Number of end-users	Very limited	Many (potentially millions)
Documentation	Optional (though recommended)	Essential

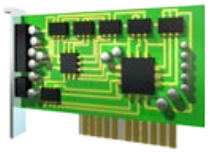
- Software is more than code
  - Program design documentation, user guides…
- **Software engineering** intends to support **professional** software development

What kind of software should you be developing? Why?

What makes good software?

Types of applied research applications

- Industrial
  - Embedded, real-time, safety-critical
  - *E.g.* Data collection and control systems

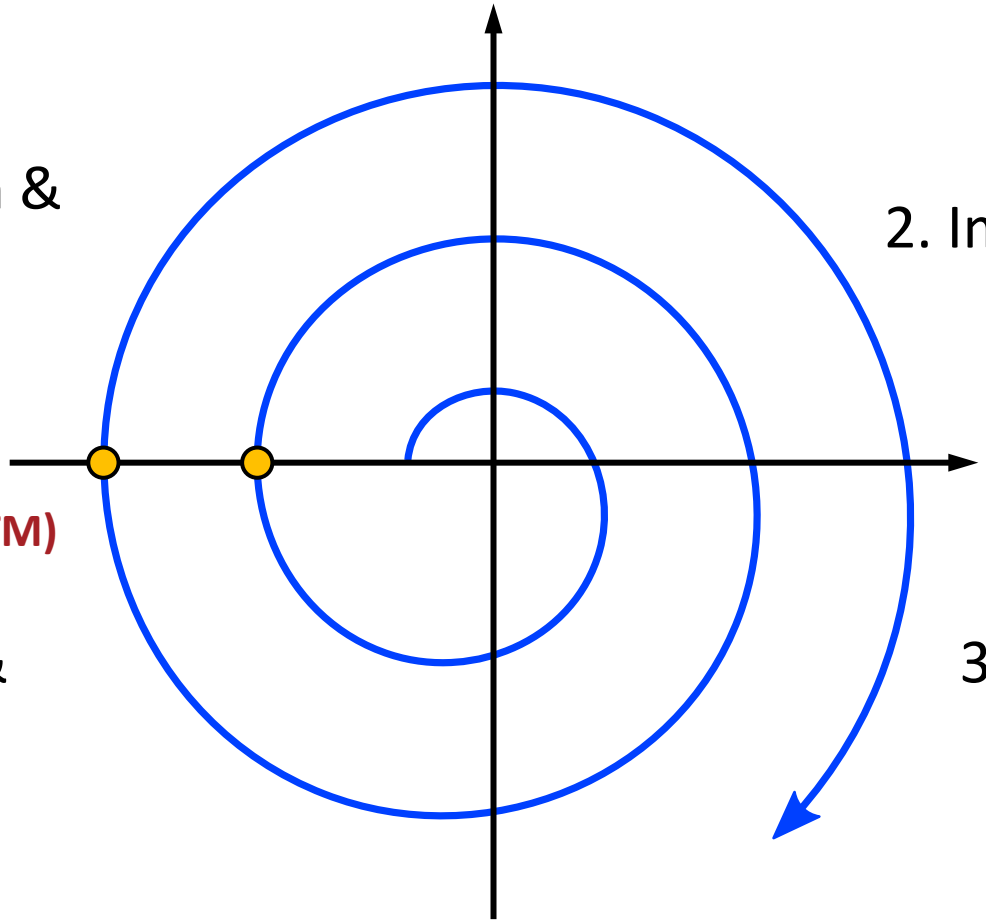


- Interactive transaction-based
  - Databases, distributed access, communications, HMI
  - *E.g.* Web applications

- Scientific and technical
  - Simulation, optimization, planning
  - *E.g.* Modeling tools

You typically need to combine elements from all of them

Spiral software development



2

System modeling

Unified Modeling Language (UML)

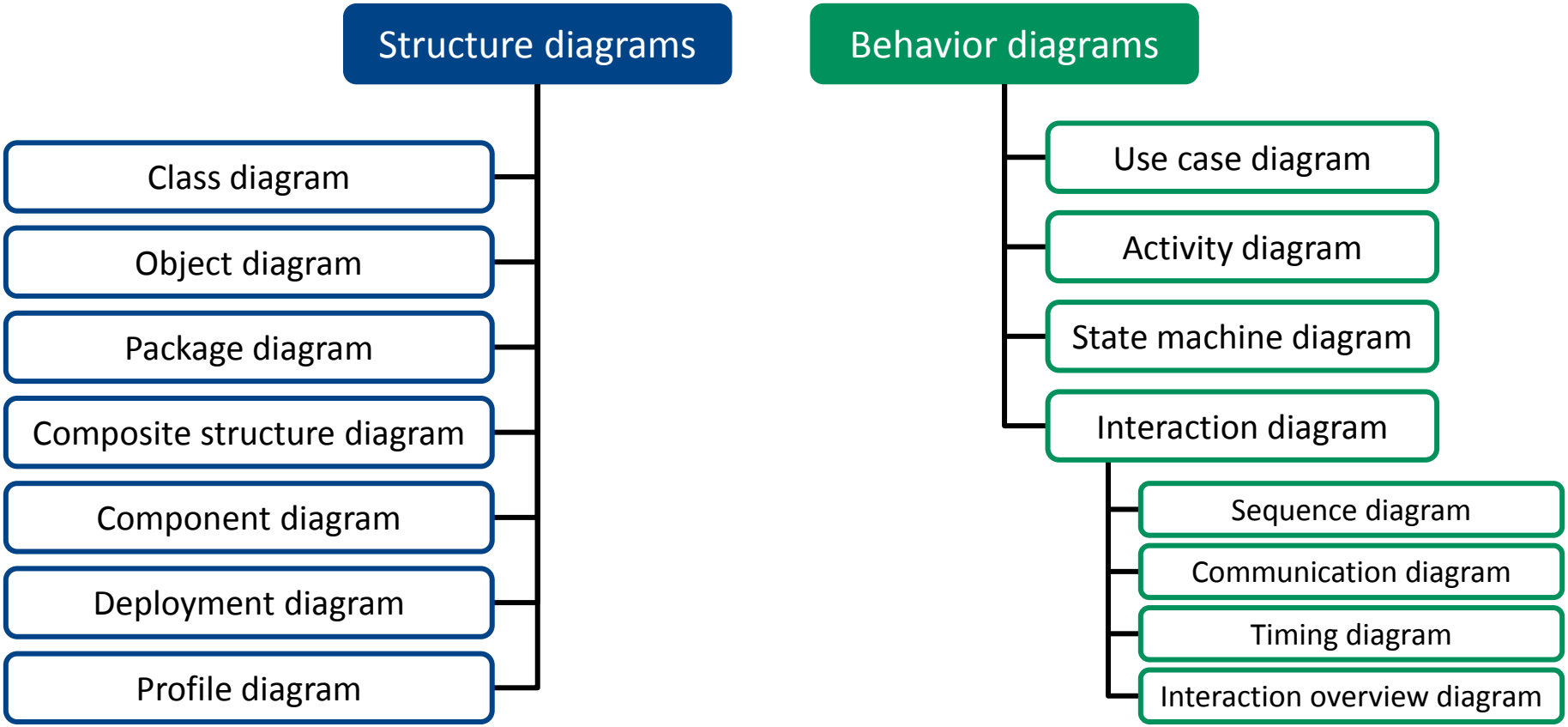
- Start coding right away is a bad idea
  - In the long run you need to rewrite most of it

DESIGN BEFORE CODING!

- UML is a set of universally standard diagram types to model software systems from different perspectives

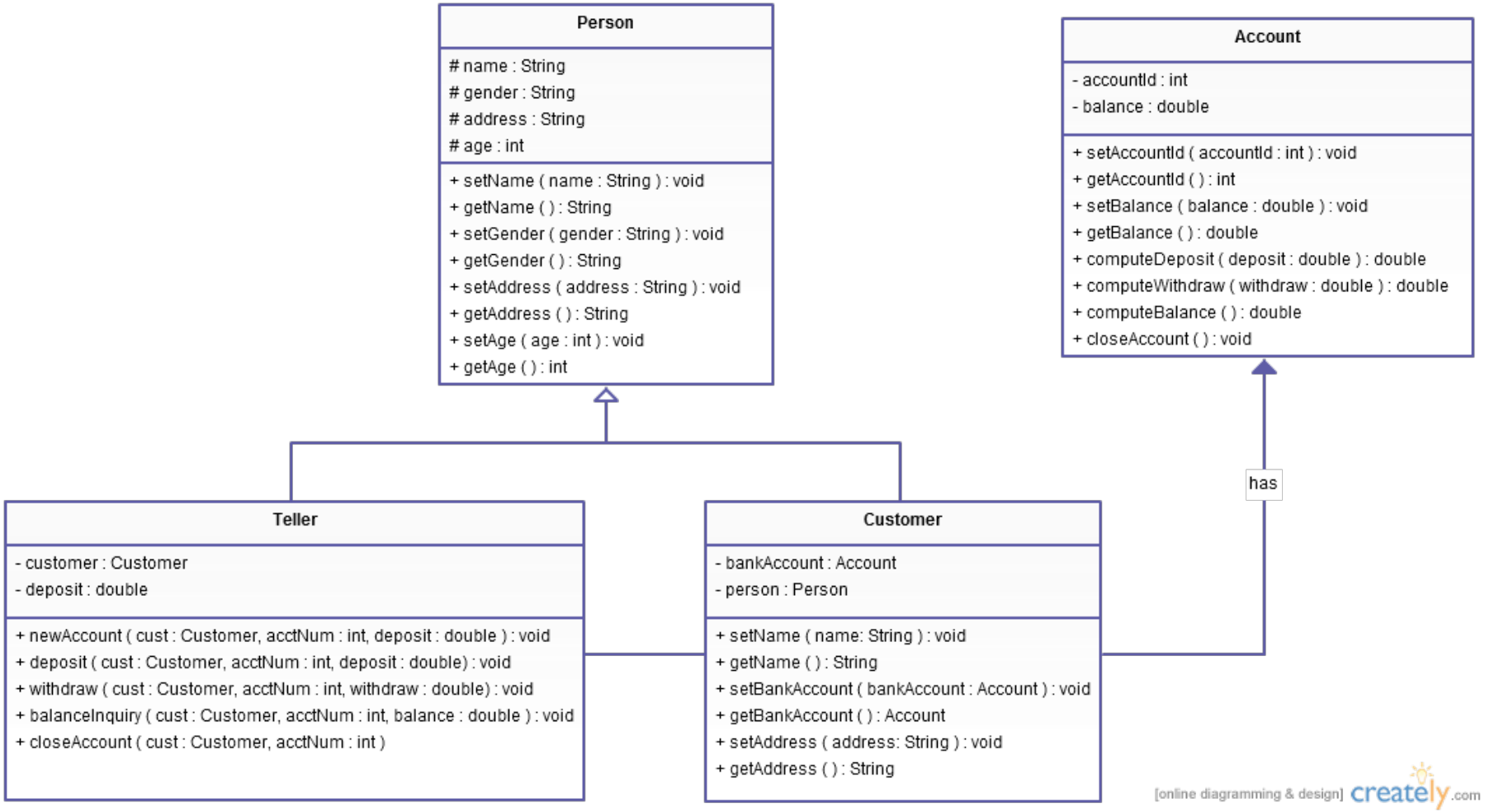
- These diagrams are used throughout the development process
  - Facilitate discussion during requirements elicitation
  - A detailed description makes implementation easier
  - Technical documentation when the project is over
  - ➡ They need to be kept up-to-date!

UML diagrams

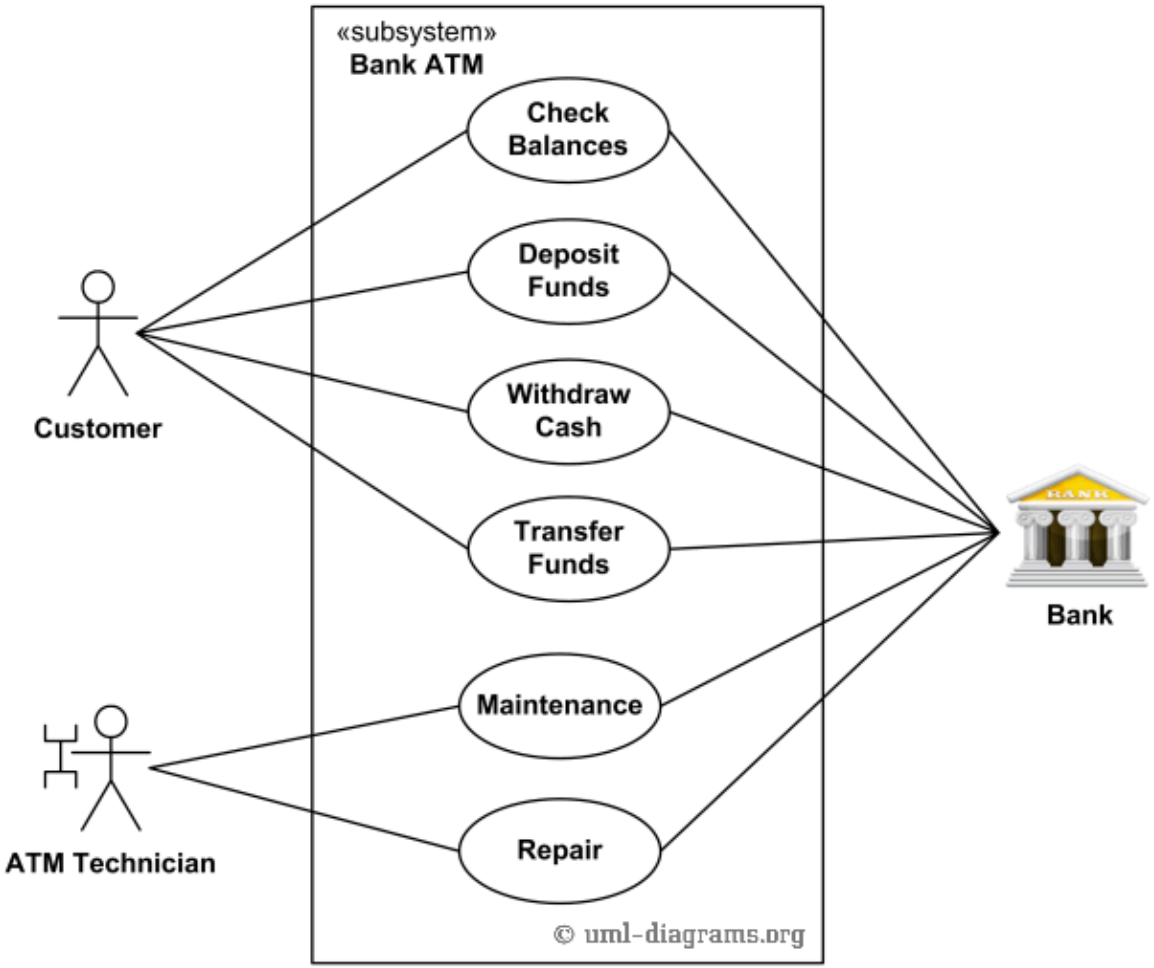


- Check http://www.uml-diagrams.org for details

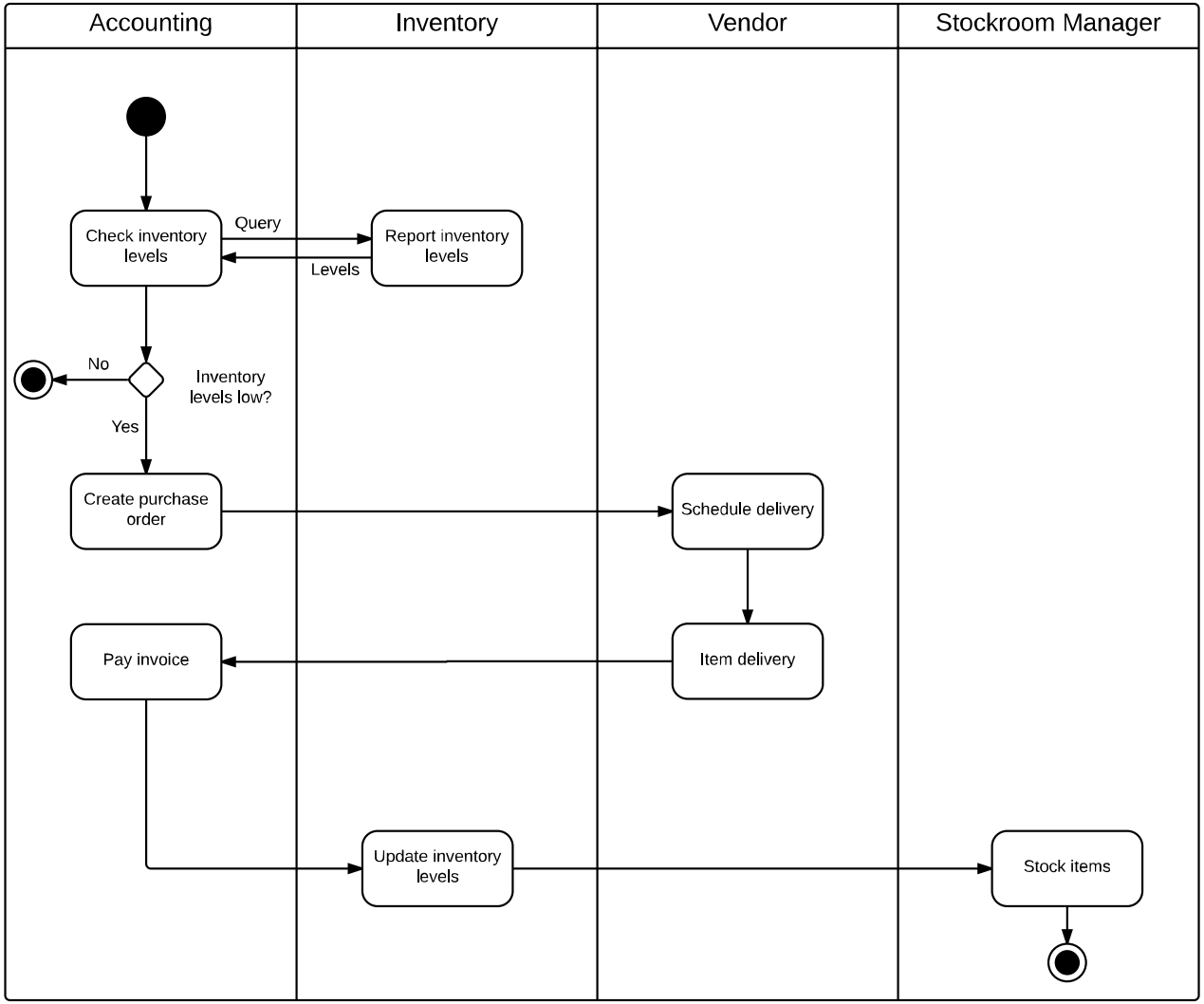
Structure diagrams | Class diagram



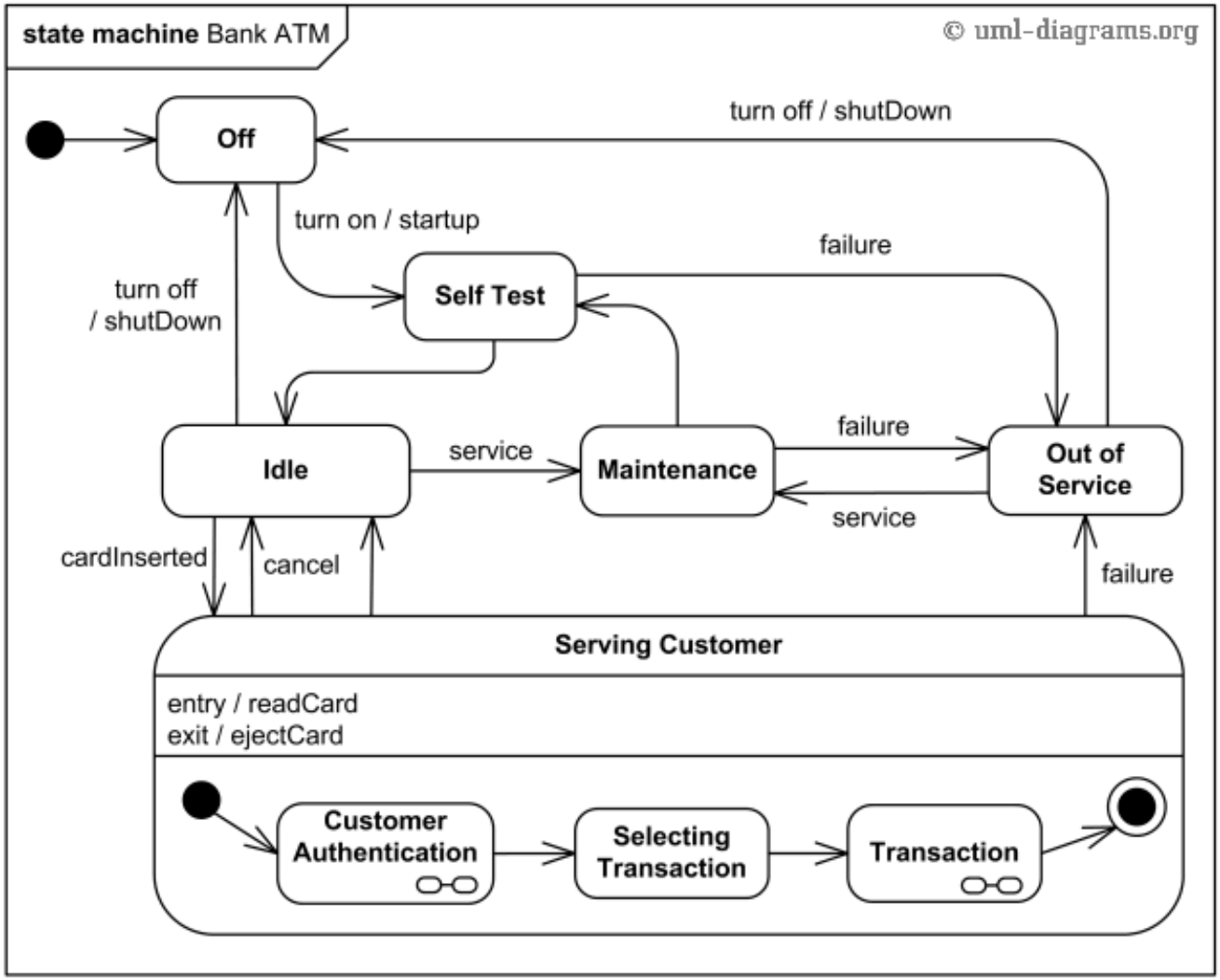
Behavior diagrams | Use case diagram



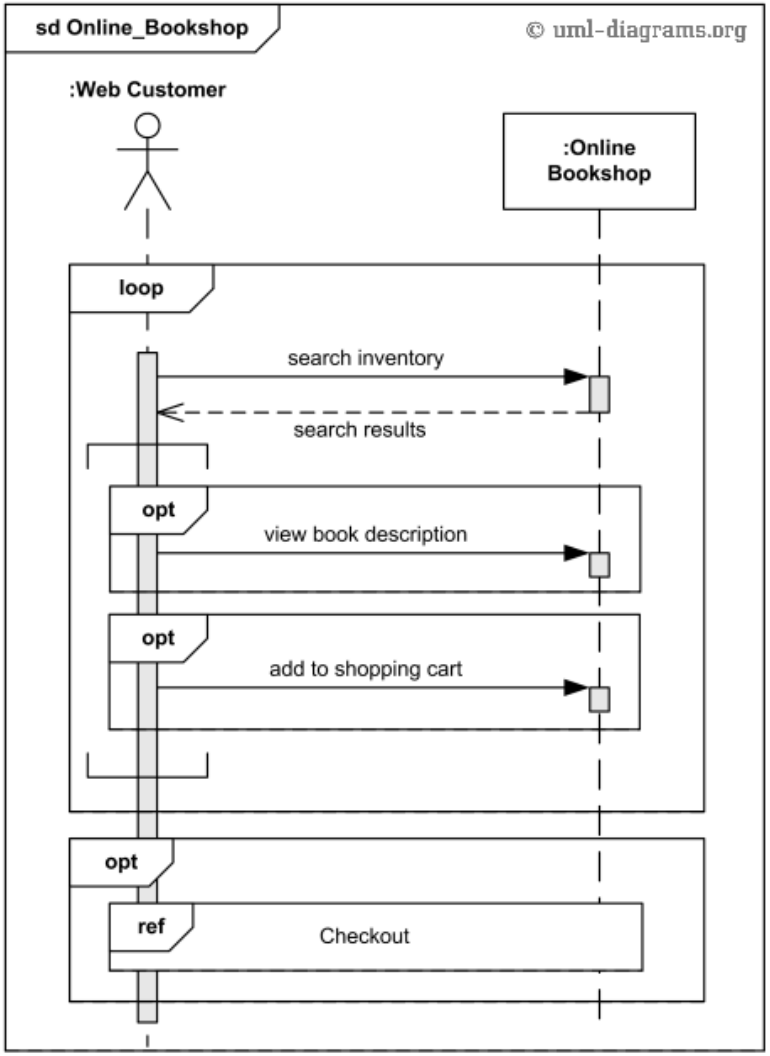
Behavior diagrams | Activity diagram



Behavior diagrams | State machine diagram



Behavior diagrams | Sequence diagram



Some UML tools

- Browser

- Desktop

Takeaways

- Software is much more than code
  - Remember what makes good software
- Software development should be incremental
- Not all applications are the same
  - Always bear in mind your particular requirements
- Modeling your tool before you start coding is worth the effort
  - Prevents misunderstandings of the specification
  - Improves communication within the development team