Requirements and Specifications

Week 4

No one is satisfied with his fortune, and everyone is satisfied with his wit.

— Leo Tolstoy, Anna Karenina

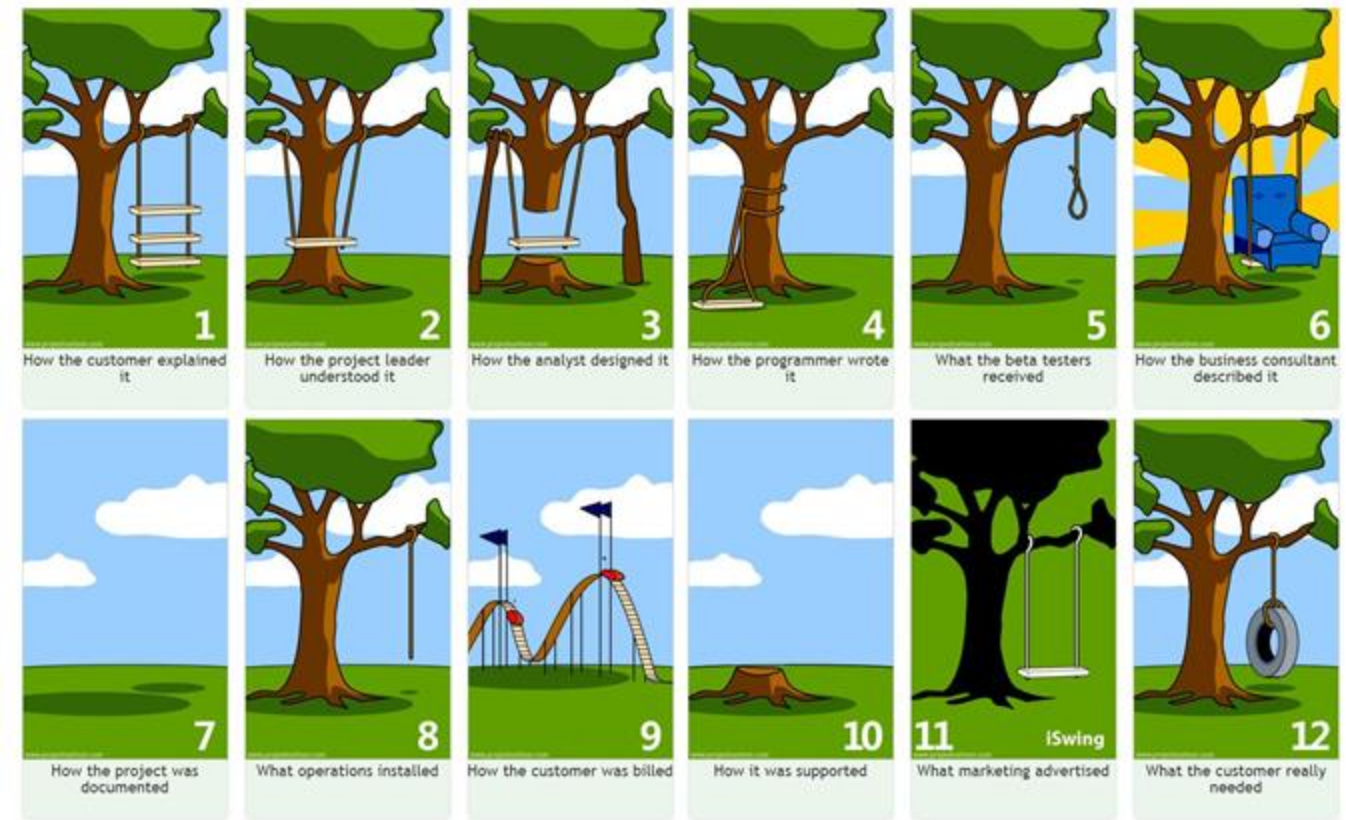
Objectives

- Practice writing user stories
- Concretize the UI and draw a wireframe
- Discuss about the design detail and draw class diagrams

Contents

- Recap on requirements and specifications
- Exercises
  - User stories
  - Wireframe
  - Class diagram

Recap: Difficulty of Capturing Requirements



Source: [Project Management - A Tree Swing Story]

Recap: Requirements Engineering Process

- Eliciting
  - Gathering and helping users to define requirements
  - Various strategies such as interviews, workshops, prototyping
- Analyzing
  - Formularizing the requirements gathered
  - User stories and use cases
- Documenting
  - Formal or semiformal documentation

Recap: User Stories

- Description of a system from the perspective of an user
  - Small scale and easy to use
  - Explain purpose and the value of the feature
- Captures functional requirements
- Focus on who, what, and why
- Used to communicate with users
- Connextra template
  - `As a` <role>, `I can/want` <capability>, `so that` <receive benefit>
  - `As a registered user, I can authenticate against my profile and retrieve my group membership, so that I can access my data and features permitted to the group.`

Recap: User Acceptance Criteria

- User story can be translated into user acceptance criteria
- Behavior-driven development (BDD) approach
- Structure of user acceptance criteria
  - Given: the initial context of the scenario
  - When: the event that triggers the scenario
  - Then: the expected outcome

Recap: User Story Evaluation

- Good user stories should be … ***INVEST***[1]!
  - **I**ndependent
    - Should not depend on others
  - **N**egotiable
    - A story is not a contract, but part of a conversation
  - **V**aluable
    - A story should be valuable to someone
  - **E**stimable
    - Development cost must be estimable
  - **S**mall
    - User stories should be small and cover single thing
  - **T**estable
    - Developers should be able to set acceptance criteria

1. Wake, Bill. *"INVEST in good stories, and SMART tasks."* Retrieved December 13 (2003): 2011.

Recap: Specifications

- Informal specifications
  - Written in natural language
  - Widely used but suffer from various problems
- Formal specifications
  - Mathematics-based techniques
  - Hard to write and understand
- Semi-formal specifications
  - Structured analysis with graph notation

Recap: Class Diagram

- One type of static structure diagram
- Describes the structure of a system
  - Classes
  - Attributes
  - Methods
  - Relationships
- Object-oriented modeling

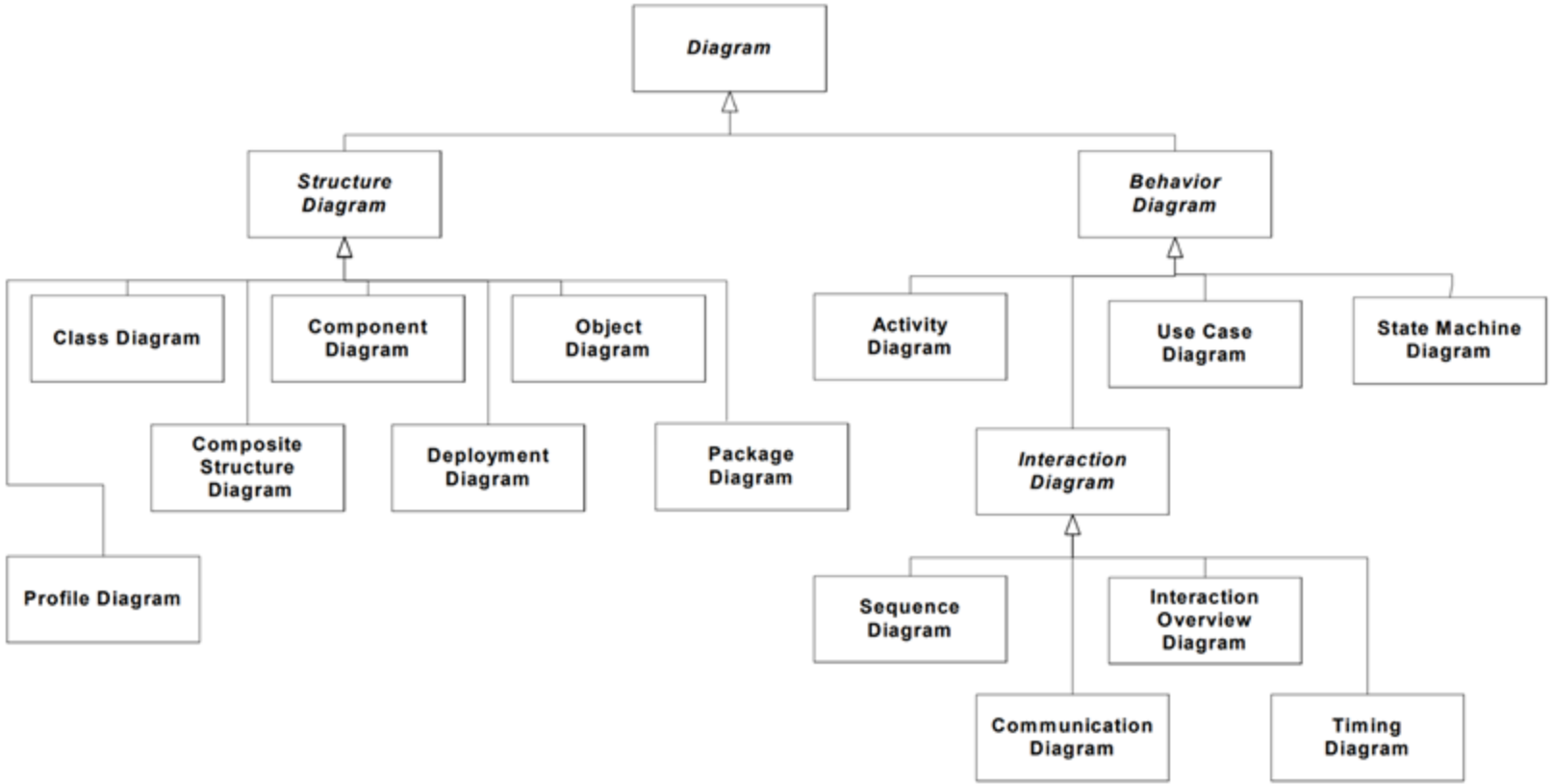
Source: https://en.wikipedia.org/wiki/Class_diagram

Recap: Unified Modeling Language (UML)

- General modeling language
- Standard of visualizing the design
- Two categories of diagrams
  - Structure diagrams: static aspect of the system
  - Behavior diagrams: dynamic aspect of the system

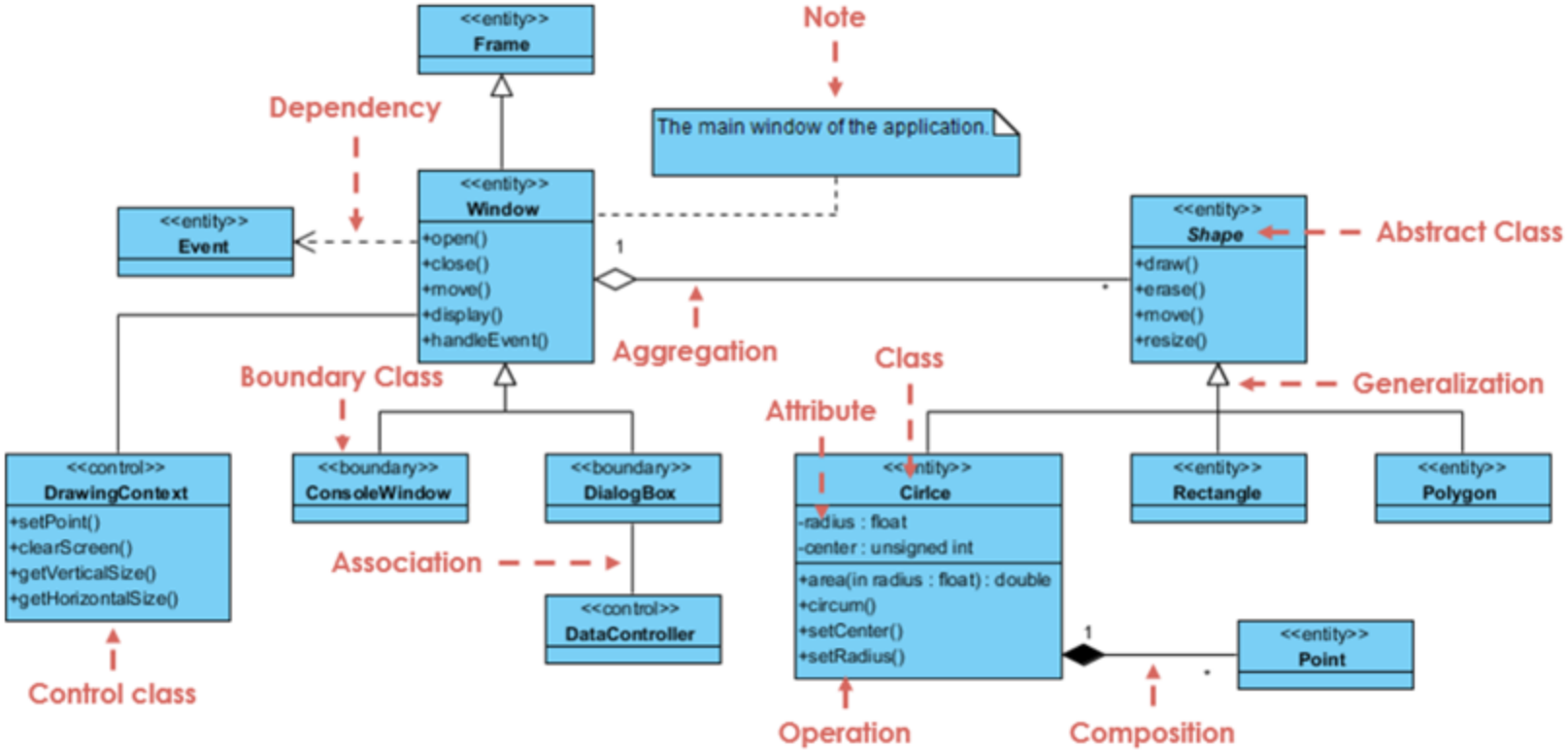
Source: Unified Modeling Language User Guide, The (2 ed.). Addison-Wesley. 2005. p. 496. ISBN 0321267974.

Recap: Diagrams of UML



Source: "OMG Unified Modeling Language (OMG UML), Superstructure. Version 2.4.1". Object Management Group. Retrieved 9 April 2014.

Recap: Class Diagram Example



Source: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/

15 mins

Exercise 1: User Stories

- Write your own **user story** and **user acceptance criteria**
  - One user story and user acceptance criteria per individual
  - If you already have them, you can revise them or add new stories
  - Any story related to your project is okay!
  - Remind ***INVEST***
- Reflect it into the **GitHub Wiki** of your project
  - Create at least 2 user stories per team in this session
  - Include the user stories for all the key features in the wiki
    - 5 (or more) user stories will be used for user acceptance test
  - More user stories will help you clarify on what to build

25 mins

Exercise 2: Wireframe

- Discuss how to configure the UI of the application
- Create a **wireframe**
  - Online tools are available for wireframe
  - Drawing is also possible
    - Keep in mind that you should continually update the wireframe
  - Concrete wireframe will make coding easier
- Reflect it into the **GitHub Wiki** of your project
  - High resolution image with lossless format (e.g., PNG)
  - Manage image files in `docs/images/` directory

~End

Exercise 3: Class Diagram

- Discuss overall structure of your program
- Create **class diagrams**
  - For frontend and backend
  - Online tools are available for class diagram
  - You may start with learning from tutorial
  - Create them as detailed as possible
- Reflect it into the **GitHub Wiki** of your project
  - High resolution image with lossless format (e.g., PNG)
  - Manage image files in `docs/images/` directory

Thank You.
Any Questions?