CS 4500
Software Development

[Overview of UML and Pair Programming]

Ferdinand Vesely

September 13, 2019

Recall

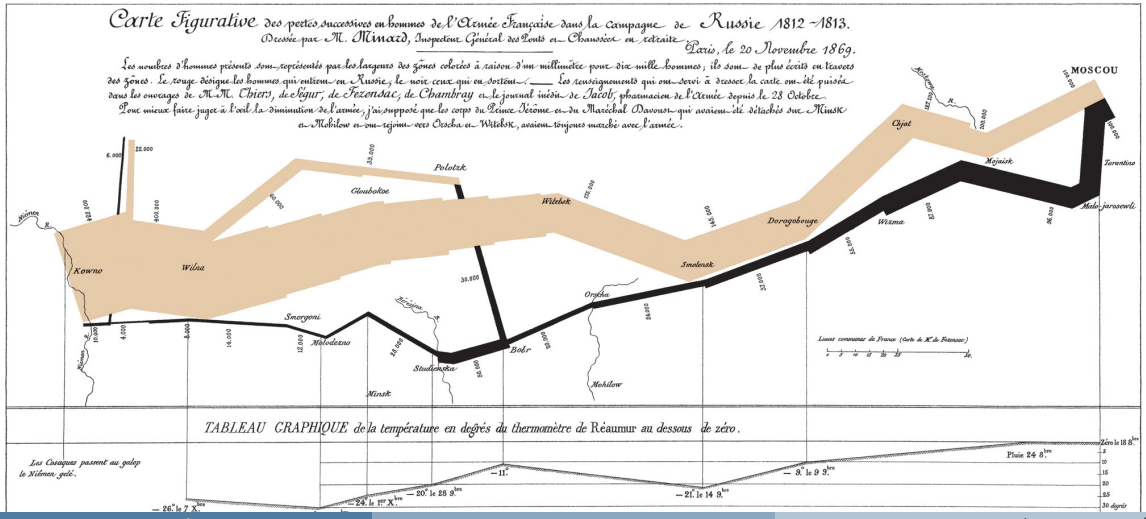
1. **Analyze the problem/requirements/scope**
2. **Gather use cases**
3. **Identify components and interactions**
4. Plan and build a minimal prototype
5. Iteratively refine the prototype / add use cases

Communication

- Emphasized in this course
- Shared understanding
- Common language to communicate ideas

Visual

- Faster to process
- Efficient
- Helps focus process



Unified Modelling Language

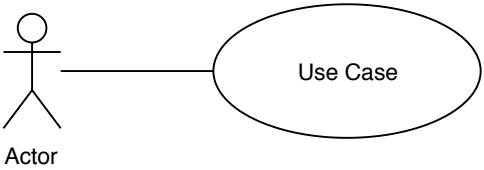
- standardized (OMG, ISO), general-purpose *visual* modelling language
- developed in late 1990s
- overseen by the Object Management Group
- huge – spec is over 700 pages
- associated with object-oriented methods
- latest standard 2.5.1 from 2017

UML Diagrams

13 diagram types across 3 categories

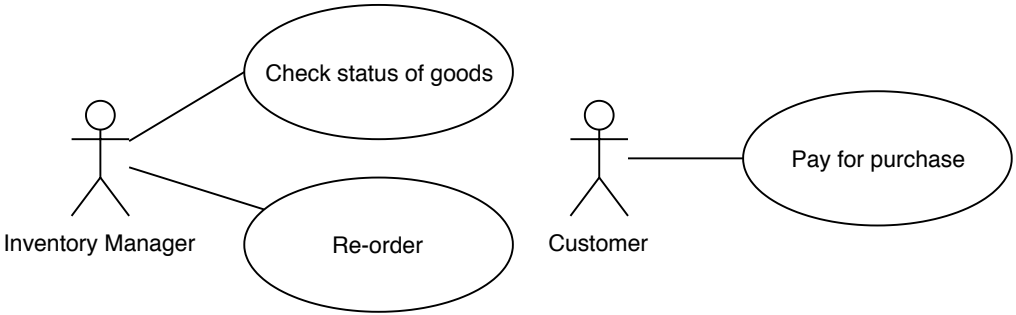
1. Structural
   - models structure of systems
2. Behavioral
   - express behavior of systems
3. Interaction
   - express data and control flow

Use Case Diagrams



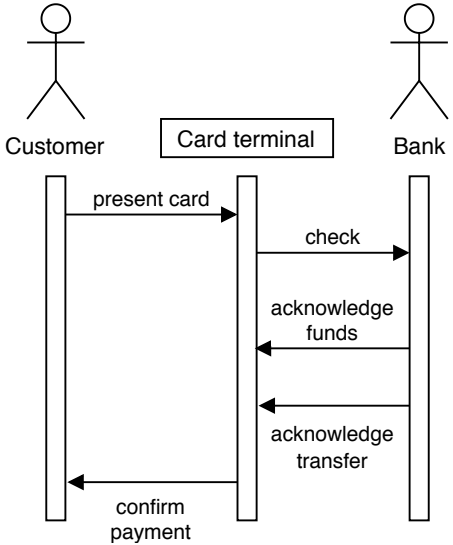
Actor – person or other system

Use Case Diagrams



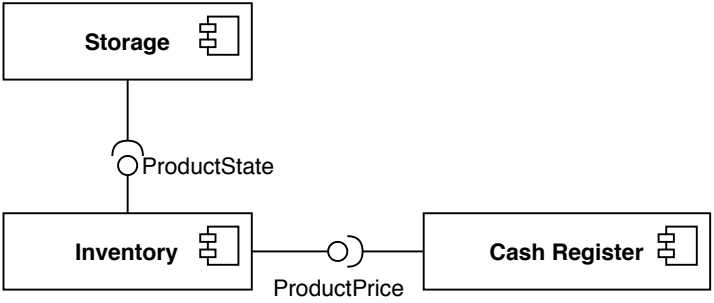
Sequence Diagrams

- models flow of data or control between components, subsystems, actors



Component Diagrams

- models components and their interfaces



Other Diagram Types

- Structural:
  - class diagram, package diagram, object diagram, composite structure diagram, deployment diagram
- Behavioral:
  - activity diagram, state machine diagram
- Interaction:
  - communication diagram, interaction overview, timing diagram

UML Summary

- Standard language – aids communication
- If you are using diagrams as part of design, might as well take inspiration or use standardized ones[1].->Not required though
- Plenty of tutorials available
- Interesting use: code generation from diagrams

Pair Programming (Recap)

- 2 programmers, one keyboard / pencil / mouse
- jointly produce one artifact – code, design
- working “as one”

Pair Programming

1. driver: controls the keyboard
   - writes code or draws up design
2. observer (or navigator)
   - continuous and *active* observation of driver's work
   - watches for defects
   - alternatives
   - looking up resources
   - considering the bigger picture

Pair Programming

- Partners deliberately switch roles periodically
- Wholly share ownership of the work
- In general, people have unequal skills
- Goal: use each others strengths and learn from each other.

Pair Programming – Why?

- Two pairs of eyes better than one.
  - design and programming is a thinking activity
- Overall, a slight increase in cost/time is offset by less buggy in code
- At any moment, (at least) two people are well familiar with the code base
  - better than having a single start developer with exclusive knowledge
  - avoids problems with staff turnaround