German Jordanian University الجامعة الألمانية الأردنية

CS342 Software Engineering

Dr. Ismail Hababeh

German-Jordanian University

Lecture 4

Project Plan

The project plan should contain the following software activities/decisions in the inception (first) phase:

1. Introduction
2. Activities Organization
3. Activities Scheduling
4. Risk Management

1. Introduction

The project plan introduction should contain the following activities/decisions:

- Project overview
- Project deliverables
- Hardware and software resource requirements
- Reference materials
- Definitions and acronyms

2. Organization Activities

The project activities organization should contain the following tasks/decisions:

- Organizational Structure: Internal management, organization chart, relations between project entities.
- Project roles/responsibilities
- Major functions and processes
- Assumptions, dependencies, and constraints
- Monitoring and controlling mechanisms, reporting mechanisms and formats, Information flows, reviews.

3. Activities Scheduling

The project activities scheduling should contain the following tasks/decisions:

- Divide the project into sub-systems, definitions of sub-systems tasks
- Dependencies: precedence relations among functions, activities and tasks
- Resource estimation requirements: personnel, computer units, special hardware, supplementary software.
- Budget and resource allocation: connect costs to functions, activities and tasks.
- Schedule: deadlines, accounting for dependencies, required indicators

4. Risk Management

The project risk management should contain the following tasks/decisions:

- Risk identification
- Risk analysis
- Risk controlling
- Risk monitoring

Additional Project Plan Activities

Additional activities in the first phase:

- Kick-off meeting
- Team building
- Discussions with end-user
- Acceptance testing plan

Additional Project Plan Activities

Additional activities in the rest of project phases:

- Risk monitoring continuity
- Schedule monitoring continuity
- Weekly checkpoint meetings
- Formal review meetings
- Meetings with end-user

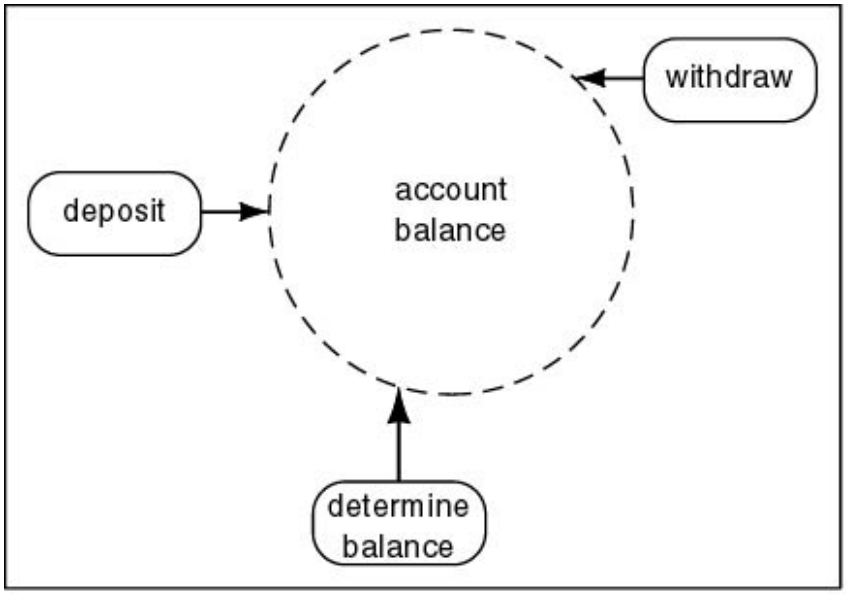
Software Development Prototypes Structured vs. Object-Oriented

- In structural paradigm:

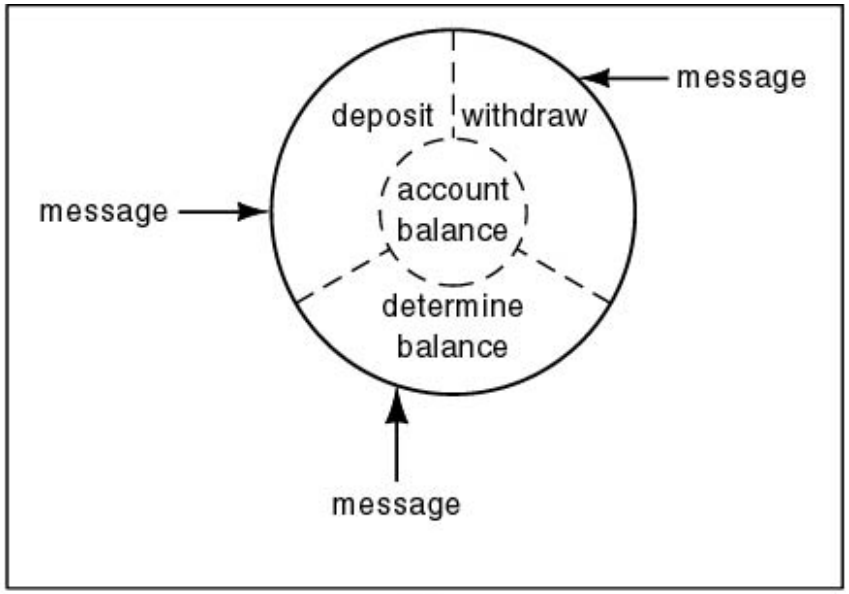
The software product conceptually consists of a single unit (although it is implemented as a set of modules).

- In Object Oriented paradigm:
  - Software is composed of number of units (classes)
  - Lower complexity
  - Better in maintenance and development
  - The way the action is carried out is the responsibility of the object

Software Development Prototypes Structured vs. Object-Oriented



(a)



(b)

Structured vs. Object-Oriented Prototypes

- In the structural paradigm
  - All the modules have details of the implementation of `account_balance`
- In the object-oriented paradigm
  - The solid line around `accountBalance` denotes that outside the object there is no knowledge of how `accountBalance` is implemented

Structural vs. Object-Oriented Workflows

- In the structural prototype:
  - Analysis Workflow:
    - Determine what to do
  - Design Workflow:
    - Determine how to do it
  - Architectural design
    - Determine the modules
    - Detailed design
      Design each module

- In the object-oriented prototype :
  - Analysis
    - Determine what to do
    - Determine the objects
  - Design
    - Determine how to do it
    - Design the objects

Strengths of the Object-Oriented Prototype

1. With information hiding, post-delivery maintenance is safer.
   – The chances of a regression fault are reduced
2. Development is easier
   – This simplifies modeling (a key aspect of the object-oriented paradigm)

Strengths of the Object-Oriented Prototype

3. Well-designed objects are independent units
   - Everything that relates to the real-world item being modeled is in the corresponding object *encapsulation*
   - Communication done by sending *messages*
4. The object-oriented paradigm promotes reuse
   - Objects are independent entities

Can we use the object-prototype to enhance a Structural Product?

- The object-prototype can't be used to enhance a product developed using the structured prototype.
  - Example: Water and oil do not mix
- Exception: if the new part is totally disjoint
  - Example: adding a GUI (graphical user interface)

Software Engineering Development Terminologies

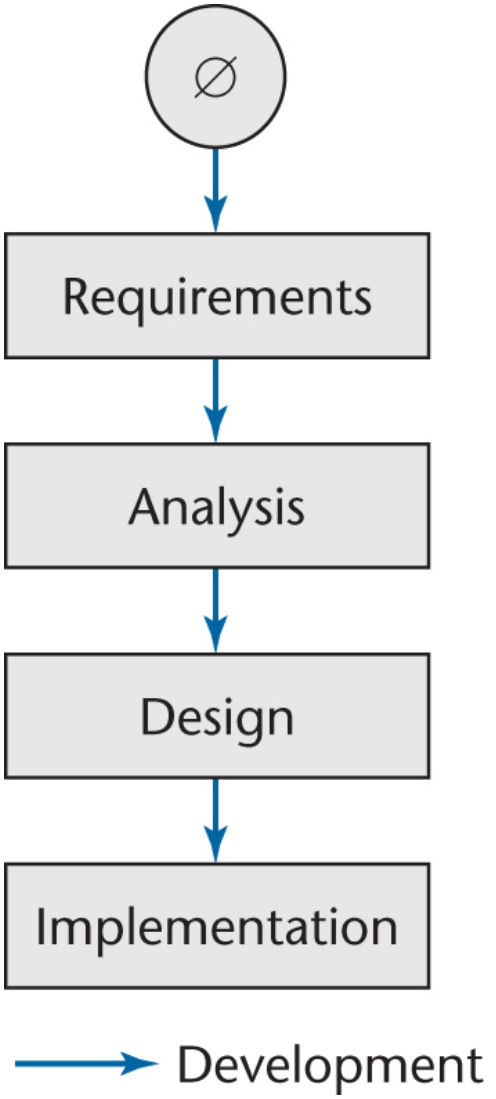
- Program, product: nontrivial piece of software
- System: combination of (software + hardware)
- Methodology, prototype: collection of techniques
- Bug: fault in the software

Object-Oriented Terminologies

- Data component of an object (instance)
  - State variable
  - Field (C++)
  - **Attribute** (generic)
- Action component of an object
  - Member function (C++)
  - **Method** (generic)

Software Development in Theory & Practice

- Software development in theory:
  - Linear (one phase after the other)
  - Starting from scratch
- Software development in practice:
  - In the real world, software development may different
  - The client's requirements may change while the software product is being developed.



Software Development in Practice - Example

Case Study - GJU Registration System

- **Phase 1**: The first version is implemented
- **Phase 2:** A fault is found
  - The product is too slow because of an implementation fault
  - Changes to the implementation are started
- **Phase 3:** A new design is adopted
  - A faster algorithm is used
- **Phase 4:** The requirements may be changed
  - Accuracy should be increased
- **Phase 5:** few years later, these problems carry on