



Aalto University
School of Science

Program design

CS-C2120, Programming studio 2

CS-C2105, Programming studio A

13.1.2021

Program design aspects

- Conceptual analysis of the problem domain.
- Identifying initial classes / objects.
 - And initial methods / instance variables
- Designing data structures / collections to be used.

Program design aspects, cont.

- Designing user interface
- Designing general program logic
- Designing access to external data (files, data bases, network access, ...)
- Designing data structures and algorithms

Aspects not covered here

- Software architecture
- Requirements engineering
- Choice of tools / libraries / technologies
- Usability points of view
- Choice of *efficient* data structures and algorithms
- Technical linking to data bases and network resources

CS-E4950
Software
Architecture

CS-C3150
Software
Engineering

CS-C3120
Human-Computer
Interaction

CS-A1140
Data Structures
and Algorithms

CS-A1150
Data Bases

Some advice

- Designing is highly important
 - There is no "right" and unique best design
 - Rather some designs are better or worse related to different criteria, like
 - Clarity, complexity, cohesion, coupling, performance, etc.
 - Design skills improve with experience
 - When you have to modify your program structure, consider why this is needed and what failed in your initial design
-

Program design approaches

- Top – Down
 - Focus first on high-level design
 - Proceed in refining actions
 - Bottom – Up
 - Focus on identifying generic "tools"
 - Build bigger things by using these
 - Support code reuse
 - Both together
 - Practical approach
-

Conceptual analysis

- Start with a verbal description of the project goal
 - Can be free form, but gives a comprehensive enough description of what functionality should be available when the project is ready.
 - Proceed with Noun and Verb analysis

Noun method

- **Goal:** Identifying initial class structure
 - **Process:**
 - Identify all different nouns in the verbal description
 - List them separately
 - Add clarifications in parenthesis, if needed
 - Cluster related nouns as separate groups and give clusters a title
 - Remove overlapping / redundant terms
 - Consider relations of terms and identify potential abstractions
 - Revise correspondingly
 - Identify initial classes, their relations and instance variables
-

Verb analysis

- **Goal:** Identify methods for classes
 - **Process:**
 - Identify all different verbs in the verbal description
 - List them separately
 - Add clarifications in parenthesis, if needed
 - Cluster related verbs as separate groups
 - Remove overlapping / redundant terms
 - Identify how actions are related to initial classes
 - What information and parameters are needed?
 - Revise the classes and methods, as appropriate
-

User stories

- Write together with the customer a number of short descriptions of activities
- Test your design, whether these stories make sense in it.

Let us consider an example

- Rogue game
 - You can access the text description in MyCourses (Lecture materials)

Summary 1

- Noun analysis
 - Identify nouns in the description
 - Combine synonyms
 - Cluster nouns which are related to each other => identify abstractions
 - Form initial classes
 - Revise class structure, where needed

Summary 2

- Verb analysis
 - Identify verbs in the description
 - Combine verbs with the initial classes
 - Identify different types of verbs (action types)
 - Helps planning methods
 - Identify target of actions
 - possible method parameters

Next week

- CRC cards
- User stories
- UML