

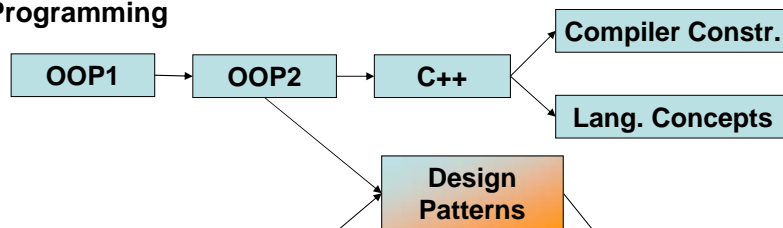
Design Patterns: Between Programming and Software Design

Dominik Gruntz & Christoph Denzler
University of Applied Sciences Northwestern Switzerland
Institute for Mobile and Distributed Systems

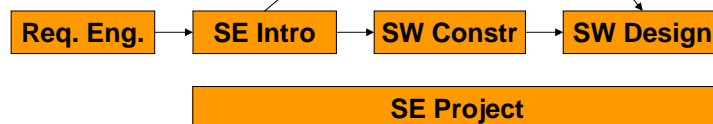


Computer Science Curriculum

■ Programming



■ Software Engineering



Computer Science Curriculum

■ Programming

- Precise Discipline
- Tools (compilers, runtime) are not forgiving errors
- Focus: correct programs

■ Software Engineering

- Soft Discipline
- There hardly exists "the right solution"
- Focus: balance between different forces

Ways to Teach a Design Patterns Course

■ Programming Focus

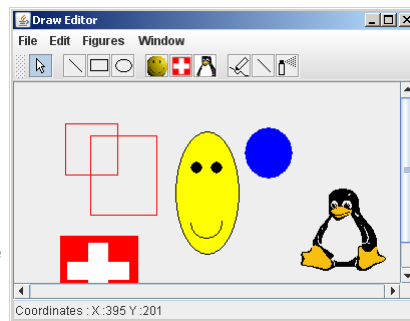
- Content: Set of Patterns (GoF, Grand, Head First)
- Focus: Implementation details, programming skills
- Scope
 - Pro: Best fitting examples can be chosen
 - Con: Patterns are seen as UML diagrams rather than problem/solution pairs

■ Software Engineering & Design Focus

- Content: General Design Principles
- Focus: Patterns as applications of higher-level concepts
- Scope
 - Pro: Design Patterns are presented from a design point of view
 - Con: Students may be overwhelmed by general design rules
 - Con: Students do not learn to effectively apply the patterns

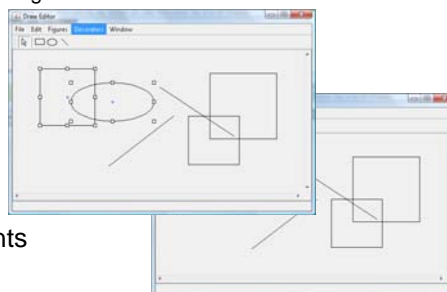
A “hands-on” Design Patterns Course

- **Course based on a Graphics Editor**
 - Focus on *problems* (editor features) rather than *solutions* (design patterns)
 - Implementation rather than discussion only
- **Assignments**
 - Each assignment adds a new feature to the editor
 - Knowledge consolidation
 - Effort is rewarded with a running application
 - Deeper understanding of how design patterns integrate into larger systems



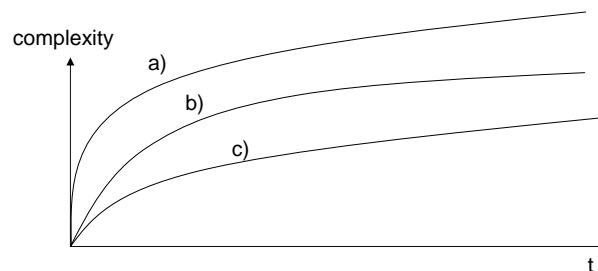
Editor Features & Patterns Used

- Single Document - Multiple View Editor (Observer)
Views have to be consistent
- Tool Bar (State / Abstract Factory)
Tools control the construction of new figures
- Group/Ungroup
(Composite)
- Undo/Redo
(Command)
- Clipboard, Cut/Copy/Paste
(Prototype)
- Generic Figure Enhancements
(Decorator)
- Figure Grid (Strategy)



Workload for assignments

- Provide or Build the Framework?
- Which parts should be provided at the beginning?
 - a) Requirements only
 - b) Simple but running editor with a set of requirements for extensions
 - c) GUI and interfaces for the main entities



Provided Interfaces

- Advantages
 - Students can concentrate on application of the design patterns
 - Test cases inspecting implementation details can be provided
 - Teamwork is encouraged
 - Exchangeability of figure implementations
 - Effort for assignment support / mentoring is reduced
 - Teacher can give concrete feedback on malfunctions (due to his/her experience)
 - Correction and annotation of assignments is easier
 - System and its interfaces may be used as a context for exam questions

Provided Interfaces

■ Disadvantage:

Anticipate challenging questions about where and how patterns are applied

- Strategy: Design of strategy interface is given
- Composite: Decision between transparent / safe approach is taken
- Observer: Roles are given (Model & Figures are observables)

■ Experience

- The correct wiring is still challenging
- Give talented students the possibility to extend the editor

Beyond Design Patterns

■ Effective Use of Design Knowledge

- Project based learning
 - Software Engineering project over 3 semesters
 - 180h per Student per Semester
 - Usually 6-8 students per group => 3500h

■ Software Design Course

- Focus: what is "good" design
- Teaching with plausible examples, practice it on a complex system
 - Topics become intertwined
 - New questions arise automatically:
 - Compatibility with older versions
 - System wide exception handling
 - Design Refactoring

Considerations / Advise

■ Demanding Assignments

- Editor extensions are orthogonal, i.e. undo/redo can be implemented without grouping feature
- Interleaved time schedule
- Solution fragments are provided (but not complete solutions, that would demotivate the students)

■ Only *one* Playground (can become boring)

- Concentration on interesting parts
- Deliver a visible success each time

■ Plagiarism

- We ignore it
- Exam questions are based on assignment knowledge

Summary

■ Knowledge Consolidation

- The patterns are applied in a concrete context
- students gains a feeling about "good" design
- Broader understanding of how design patterns integrate into larger systems
- Students discuss their different solutions

■ Motivation

- The efforts are rewarded with a small application!
- Excellent students have enough room for enhancements

■ Feedback

- Student feedback is positive