Design Patterns and Frameworks (DPF) Announcements

Prof. Dr. U. Aßmann
Chair for Software Engineering
Faculty of Computer Science
Technische Universität Dresden
WS 13/14-0.3, 11/16/13

Elements of the Course

- Meeting hour of Prof. Aßmann
  - after the course, or Thursday, 11:00-13:00. Please register with secretary Katrin.Heber@tu-dresden.de, thanks.
- Lecturing
  - Do not miss one, they should give you a short and concise overview of the material
- Reading
- Exercise sheets (Dr. Sebastian Götz)
  - Handed out every 2nd week
  - You have one week to solve them on your own
  - After that, solutions will be explained in the exercise seminars (Große Übungen)
- http://st.inf.tu-dresden.de -> Teaching -> Design Patterns and Frameworks
- http://st.inf.tu-dresden.de/teaching/dpf

Reading Along the Lectures

- Unfortunately, the course is not covered by any book
  - Only about 25-30% is covered by GOF
- You have to read several research papers, available on the internet
  - Marked by “Mandatory Literature (To Be Read)”
  - Secondary Literature or References is non-mandatory, but interesting reading
- Other Literature is not to be read, but also interesting.

Literature (To Be Read)

- During the course, read the following papers, if possible, in sequential order. See also literature web page.
  - Every week, read about 1 paper (3-4h work)
- Start here:
  - A. Tesanovic. What is a pattern? Paper in Design Pattern seminar, IDA, 2001. Available at home page.
  - Brad Appleton. Patterns and Software: Essential Concepts and terminology. http://www.cmcrossroads.com/bradapp/docs/patterns-intro.html Compact introduction into patterns.
- K. Beck, J. Coplien, R. Crocker, L. Dominick, G. Meszaros, F. Paulisch, J. Vlissides. Industrial Experience with Design Patterns Int. Conference on Software Engineering (ICSE) 1996. http://citeseer.ist.pst.edu/beck96industrial.html

Literature (To Be Read)

- [GOF, Gamma] E. Gamma, R. Helm, R. Johnson, J. Vlissides: Design Patterns. Addison-Wesley 1995. Standard book belonging to the shelf of every software engineer.
  - Chapters on Design Patterns in as far as they are handled in the lectures
  - The book is called GOF (Gang of Four), due to the 4 authors
- Alternatively to GOF can be read:
  - Head First Design Patterns. Eric Freeman & Elisabeth Freeman, mit Kathy Sierra & Bert Bates.O'Reilly, 2004, ISBN 978-0-596-00712-6
  - German Translation: Entwurfsmuster von Kopf bis Fuß. Eric Freeman & Elisabeth Freeman, mit Kathy Sierra & Bert Bates. O'Reilly, 2005, ISBN 978-3-89721-421-7
- Alternatively, available at home page. If you have already studied GOF, do not read these. These paper stem from a Design Pattern seminar at Linköpings Universitet, IDA, 2001:
  - A. Tesanovic. What is a pattern?
  - T. Panas. Design Patterns, A Quick Introduction. (on Composite, Visitor)
  - Veaceslav Caisin. Creational Patterns.
  - P. Pop. An overview of the automation of patterns.

Literature (To Be Read)

- D. Riehle, T. Gross. Role Model Based Framework Design and Integration. Proc. 1998 Conf. On Object-oriented Programing Systems, Languages, and Applications (OOPSLA 98) ACM Press, 1998. http://citeseer.ist.pst.edu/riehle98role.html
- D. Bäumer, G. Gryczan, C. Lilienthal, D. Riehle, H. Züllighoven. Framework Development for Large Systems. Communications of the ACM 40(10), Oct. 1997. http://citeseer.ist.pst.edu/bumer97framework.html
- D. Bäumer, D. Riehle, W. Silberski, M. Wulf. Role Object. Conf. On Pattern Languages of Programming (PLOP) 97. http://citeseer.ist.pst.edu/baumer97role.html

Literature (To Be Read)

- W. Pree. Framework Development and Reuse Support. In Visual Object-Oriented Programming, Manning Publishing Co., editors M. M. Burnett and A. Goldberg and T. G. Lewis, Pp, 253-268, 1995. www.softwareresearch.net/publications/J003.pdf
- Or: D. Karlsson. Metapatterns. Paper in Design Pattern seminar, IDA, 2001. Available at home page.
- D. Riehle, H. Züllighoven. A Pattern Language for Tool Construction and Integration Based on the Tools&Materials Metaphor. PLOP I, 1995, Addison-Wesley. http://citeseer.ist.pst.edu/riehle95pattern.html

Secondary Literature

- M. Fowler. Refactoring. Addision-Wesley, 1999.
- D. Riehle, H. Züllighoven, Understanding and Using Patterns in Software Development. Theory and Practice of Object Systems, 1996 http://citeseer.ist.pst.edu/riehle96understanding.html
- D. Garlan, R. Allen, J. Ockerbloom. Architectural mismatch – or why it is so hard to build systems out of existing parts. Int. Conf. On Software Engineering (ICSE 95). http://citeseer.ist.pst.edu/garland95architectural.html
- A. Abel. Design Pattern Relationships and Classification. Paper in Design Pattern seminar, IDA, 2001. Available at home page.
- T. Pop. Multi-Paradigm Design. Paper in Design Pattern seminar, IDA, 2001. Available at home page.

Other Literature

- T. Reenskaug, P. Wold, O. A. Lehne. Working with objects Manning.
  - The OOram Method, introducing role-based design, role models and many other things. A wisdom book for design. Out of print. Preversion available on the internet at http://heim.ifi.uio.no/~trygver/documents/book11d.pdf
- K. Beck. Extreme Programming. Addison-Wesley.
- H. Allert, P. Dolog, W. Nejdl, W. Siberski, F. Steimann. *Role-Oriented Models for Hypermedia Construction – Conceptual Modelling for the Semantic Web.* citeseer.org.

Please, Please Be Aware – There Will Be Pain!

- **This course is a research-oriented course**
- **It treats rather advanced material, the most recent sugar sweets of object-oriented software engineering**
- **There is no book on all of that**
  - GOF covers only about 25-30%
  - Please, collaborate! Read the articles, ask questions!
  - Do the exercise sheets
- Warning: **The oral exams can only be done if you have visited all lectures and solved all exercise sheets**
  - **The GOF Book alone is not sufficient**
- **Learn continously!**
- **Be aware: you have not yet seen larger systems**
  - You will see one small system in the labs (< 100KLOC)
  - Middle-size systems start over 100KLOC

Learning Java with the Praktomat

- In our basic course on software technology, we have published a web-based self-learning system for Java
  - into which you can enter Java programs
  - which tests style and syntax of the programs
  - and runs a test suite against your program
- The praktomat gives you feedback about your programming abilities in Java
- The Praktomat is an opportunity for you, please use it!
- Students without ZIH-account, please register by emailing to Sebastian.Richly@tu-dresden.de

http://praktomat.inf.tu-dresden.de/

The Positive Side

- If you follow carefully, you will discover an exciting world of beauty in software
- If you know all the patterns of the course, you will be a much better software engineer than the standard programmer
  - Most of the work has been discovered in the last 8-10 years, and is unknown to the programmers
  - Your language and communication will improve
- You will also be a much better software architect,
  - because patterns are good design knowledge
- and manager,
  - because patterns and frameworks teach you how to master large systems and product lines in your company
- Confession: If I myself had knewn all these patterns in 1998, my life would have been much easier
- The gain is worthwhile the pain!

Running Also in WS Term

- Academic Skills for Computer Scientists (3/1/0 SWS, 6cp)
  - In English
  - http://st.inf.tu-dresden.de/teaching/asics
  - Preparation of your master's thesis
- Future-Proof Software Systems (Dr. Frank Furrer, Senior Consultant and EU-Reviewer)
  - Architecture that is stable and can be evolved
  - In English
  - http://st.inf.tu-dresden.de/teaching/fpss
- Softwarewerkzeuge
  - http://st.inf.tu-dresden.de/teaching/sew
  - Metamodeling, technical spaces, model mappings, traceability
  - On the transition to English; on demand, I will lecture in English, but many slides are still in German

Design Patterns and Frameworks (DPF) Goals

Prof. Dr. U. Aßmann
Chair for Software Engineering
Faculty of Informatics
Dresden University of Technology

Main Goals

- Know several different kinds of patterns
  - Basic kinds of incentives for design patterns
- Explain patterns for variability, extensibility of systems
- Understand frameworks and product lines better
- Explain systematic structures for systems with 100KLOC
  - Layered frameworks
  - Facets
- Understand a different way of object-oriented design
  - Role-based design

Standard Problems to Be Solved By Design Patterns

- Variability: Exchanging parts easily
  - Static and dynamic
  - Variation, complex parameterization
  - For product lines, framework-based development
- Extensibility
  - Software must change and evolve
- Connections (Glueing, bridging, adapting)
  - Overcoming architectural mismatches
  - Coupling software that was not built for each other
- Representation of complex objects
  - Not fitting into one physical location
- Others Problems:
  - Optimization: making things more efficient
  - Structuring of interactive applications

Goal: Variability Patterns

- Variability (Variation, Exchange, Parameterization)
  - Expressing commonality and variability
  - We fix a common part (a framework) and parameterize it at variation points (variability)
  - Framework instantiation patterns describe variation of frameworks
- Understanding Templates and Hooks
  - Template Method vs Template Class, Dimensional Class Hierarchy, Bridge
  - Understanding creational patterns as variability patterns
  - Factory Method, Factory Class, Builder
- Variability patterns for frameworks
- Variability concerns
  - Exchange of communication, Dynamic call
  - Exchange of policy
  - Exchange of material in data-based applications

Goal: Extensibility Patterns

- Extensibility
  - For new, unforeseen product variants
  - For evolution
  - For dynamic change
- Understanding extensibility patterns
  - ObjectRecursion vs TemplateMethod, Objectifier (and Strategy)
  - Decorator vs Proxy vs Composite vs ChainOfResponsibility
  - Visitor, Observer (EventBridge)
- Parallel class hierarchies as implementation of facets
  - Understand facets as non-partitioned subset hierarchies
  - Layered frameworks as a means to structure large systems, based on facets
- Template/Hook Extension:
  - Code skeletons are *extended* at *hooks*
  - Frameworks can have hooks that can be extended (beyond variation)
- Framework extension patterns

Goal: Glueing Patterns for Overcoming Architectural Mismatches

- Glue patterns
  - Understand architectural mismatch
  - Understand patterns that bridge architectural mismatch
- Adaptation, bridging, connections
  - Of communication protocols
  - Between heterogeneous components (different representations, different locations, different control flow structure)
- Anonymous communication
  - For exchange of commmunicators
- Scalable communication
  - At runtime, in distributed systems

Goal: A Basic Tool: Role Modelling

- For all of that, a basic tool set is role modelling
  - Which roles does an object play in the application?
- It tells how design patterns occur in applications
  - Invented by Reenskaug. Summarized in the book “Working with Objects”, 1995
- Role-model based design
  - Why design patterns are role models of class diagrams
  - Understand the difference between roles and objects, role types and classes
  - Understand role mapping to classes
    - How roles can be implemented
  - Understand role model composition
  - Understand composite design patterns as composition of role models

Goal: Frameworks Pattern

- Understand variabilities in frameworks
  - Introducing different types of hooks for frameworks and components (TH patterns)
  - Understanding framework variability patterns
- Studying extensible framework hook patterns
  - Role Object pattern
  - Layered frameworks, implemented by Role Object
- Patterns document frameworks
  - Patterns play an important role on how a framework is instantiated
  - Whitebox, blackbox, layered, T&H framework

Goal: Structuring Interactive Applications with Tools&Materials

- Understand the central metaphors of the Tools-and-Materials architectural style for the construction of interactive applications
  - Know an example of a pattern language
- Interactive applications can be pretty complex
- TAM (tools-and-materials, Werkzeug-Automat-Material, WAM) is a *pattern language for interactive applications*
- Nice metaphors that help thinking, constructing, maintaining interactive applications

Overview of the Course

Refactoring

Framework Backward Compatibility

Variability-Based Design

Part III: Refactoring and Symmetries

Contracts

Documentation

Frameworks with components/models

Part II: Frameworking

Eclipse

San Francisco

SAP

Tools & Materials

Layered Frameworks

Part I: Patterns and Frameworks

Metapatterns and Framework patterns

Role Models

Composite Patterns

Part 0: Basic Patterns

Variability Patterns

Extensibility Patterns

Connection Patterns

Intro

Employment and Usage

The End

Prof. Uwe Aßmann, Design Patterns and Frameworks