Seminar on

Design Patterns and Architectural Solutions

„Discover, understand, use and construct design patterns to provide solutions to architectural problems“

Walter Kriha

Typical Beginner Problem in Software Development:

“ I know how to create classes from use case diagrams. But when do I create classes? How do I connect classes? What creates a “flow” through my application? How do I create an architecture from my classes? What turns my classes into a solution? And that is not all: How do I make my application more flexible, customizable, faster etc.?”

Architectural Problems

- How to extend a component or framework without breaking clients?
- How to make processing context dependent?
- How to achieve multi-tenant software (mandantenfähigkeit)?
- How to achieve isolation of processing within an application, e.g. a browser?
- How to change behavior of objects during runtime?
- How to simulate things or behavior?
- How to deal with large numbers of objects in high-speed applications?
- How to design 3D-games and game frameworks?
- How to make software extensible?
- How to achieve performance?
- How to manage resources in programs?
- How to de-couple components and processing?

see http://www.kriha.de/krihaorg/designpatterns.html for more information

Patterns to the Rescue!

- Design Patterns are proven ways to solve problems – they are NOT „new“ in most cases
- Design Patterns are „canned“ experience. They allow newbies to get architectural and problem solving compentence in a short time
- Design Patterns are a means to communicate design and implementation information in teams
- Design Patterns are not only found in software – originally they come from architecture (buildings not code)
- Design Patterns are a semi-formal way to shape architectures
- You wont get a job without knowing them. And if you do, you probably wouldn‘t want to work for such a company anyway.

Example: Adapter

Context: Developers look for this pattern when they need to integrate different pieces of hardware or software
Problem: Devices have different forms (interfaces) and do not fit to each other
Forces: Number one „force“ is that we are unable or unwilling to change the existing devices, e.g. to change all of them to use the same interface.
Solution: Use an intermediate element that looks right for both devices.
Structure: The intermediate has a two-face structure that looks different on both sides. Each side fits to one device.
Participants and Responsibilities: Plugs and jacks of different countries are now compatible. The adapter is responsible for interface adjustment. It is not allowed to change implementation details (e.g. voltage)
Strategies: The adapter should be small but rugged. No dangerous components allowed,
Consequences: Users are forced to carry the adapter with them. The adapter might be expensive and needs to be changed when country standards change.

But most important: the NAME – to remember and communicate the principle!

Where do patterns come from?

- Design Patterns are usually FOUND, not invented. Most crafts have a rich knowledge of practical and successful ways to design things (solve problems)
- Design Patterns are not only found in software – originally they come from architecture (buildings not code)

See Resources for a short paper on Christopher Alexander, the inventor of architecture patterns and their language.

Why do we need design patterns?

• Design Patterns are a semi-formal way to shape architectures and solve partial design problems.

•Design Patterns have been the single most successful „invention“ in software technology EVER. Till today higly formal methods have not found their way into practical software engineering. Design Patterns did.

•Design Patterns help beginners to acquire real design and architecture skills easily.

•Design patterns let teams FIND and COMMUNICATE solutions quickly.

•Design patterns take the fear out of complex software problems. Suddenly you know several ways to do things and you can compare the forces with proposed patterns and their consequences.

•You wont get a job without knowing them. And if you do, you probably wouldn‘t want to work for such a company anyway.

Goals for this seminar

- Learn how to become „design pattern aware“ – discover patterns used in different software
- Learn how to understand design patterns
- Learn how to use design patterns (in combination) to achieve large-scale architectural solutions
- Learn how to construct new design patterns, possibly in areas which have not been covered yet by existing patterns
- Learn how to provide complete solutions to general architectural problems

Don‘t worry – we will start with simple patterns and work our way up to more complex architectural patterns and frameworks

Roadmap

1. Organizational (exams etc.)
2. Introduction to the concept of design patterns
3. Some history on patterns: Christopher Alexanders pattern language etc.
4. A note on the importance of communication for design patterns
5. Assessment: what do you already know about design patterns (collecting patterns)
6. Patterns and form: we need to standardize on a simpler representation of patterns
7. What are your interests: find possible areas to work on.
8. Design Patterns in action: how to use them for re-factoring an initial design to perform better and become extensible.
9. Resources: Hints and tips for beginners and advanced students.

Feel free to bring your design problems for discussion. There is a lot to learn from problems and mistakes.

Organization and Duties

A presentation of a design pattern (or several collaborating patterns)– and its use and context is expected DURING the seminar so that all participants can profit from it. It is more important to share something that is not complete and „perfect“ with a team early on than to deliver a paper much later.

The presentation must follow the design pattern template form (roughly) and also show the application of the design pattern in one open source project (from www.sourceforge.net, www.objectweb.org or elsewhere.

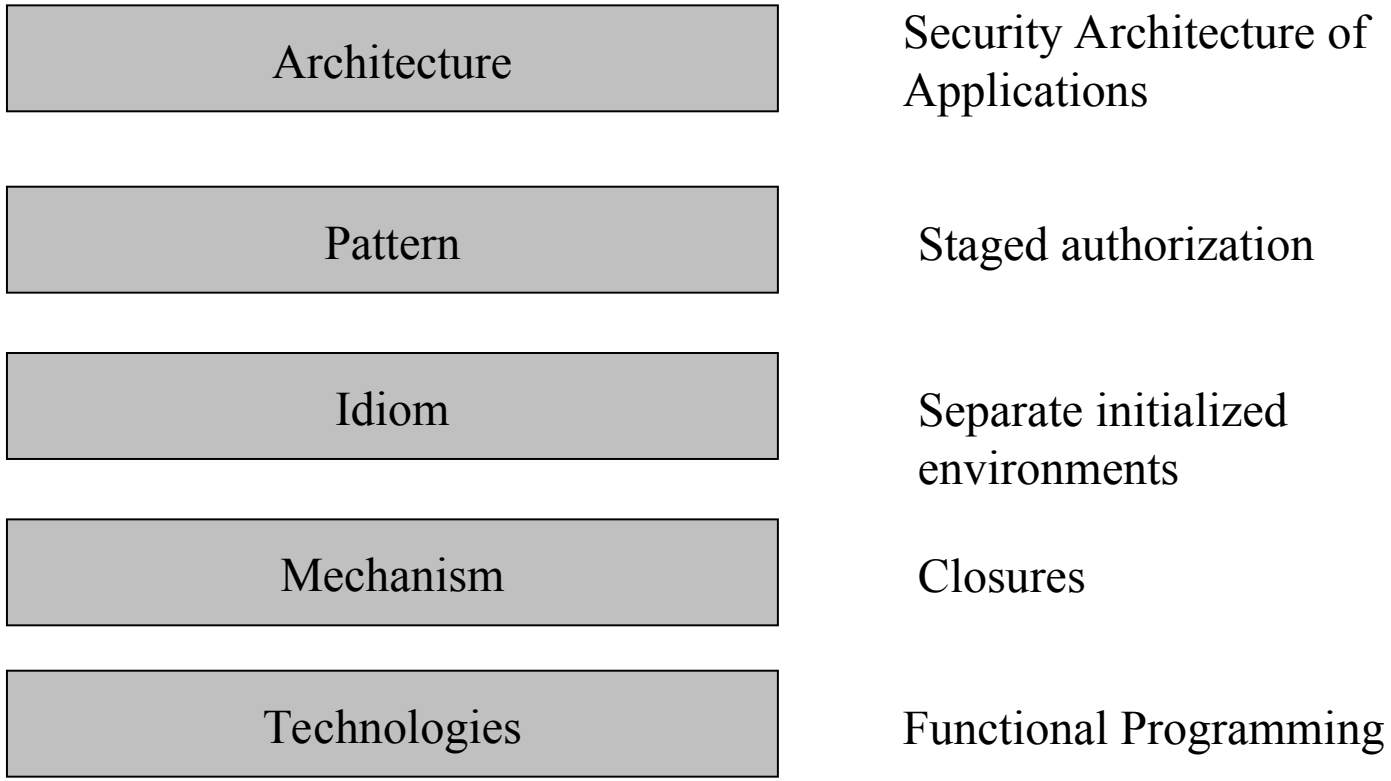
Patterns systems (applications of several patterns) need to show the SOLUTION provided by the system. Use examples from open source or create your own.

Lively participation in discussions about architectural decisions are expected as well.

No special programming language is expected/required. If you detect a design pattern in a different area (e.g. graphic design, rendering etc.) – just bring it in.

Your results will become part of the design pattern or solution catalogs at http://www.kriha.de/krihaorg/designpatterns.html

The position of patterns in the design space



For closures see: Java theory and practice: The closures debate (http://www.ibm.com/developerworks/java/library/j-jtp04247.html). But don‘t get too religious about something being an idiom or a pattern…

Software Pattern Areas

- The „classic“ patterns for beginners
- XML patterns (processing and schema patterns etc.)
- Security Patterns
- Event handling patterns (from simple observer to publish/subscribe and message bus/topics)
- Patterns for distributed computing (concurreny and distribution)
- Possible patterns in the graphic rendering process
- Enterprise Architectural pattern
- Building frameworks with patterns
- Development patterns: JUnit
- Metaobject patterns (reflection, dispatch)
- Modeling Patterns, Business Analysis Patterns (Fowler)
- Data Access Patterns
- Generative Patterns
- technical pattern: real-time constraint checking (like xml editor with schema)
- Deep Learning Patterns (Charles E. Perez)
- Serverless Patterns
- Project Management Patterns

see http://www.kriha.de/krihaorg/designpatterns.html for more information

Improving Design and Performance with Patterns

Learn how to use patterns to improve an existing design incrementally for better performance and extensibility.

A real development case will show us how this works. We will see the initial design, discuss its deficiencies, see the new design with its new patterns and what they do and also get an impression on design for performance by using pooling, threading and careful representation of heavy objects.

Eclipse has turned into a treasure box for patterns that both provide flexibility and performance. (see the extending eclipse book by Gamma and Beck)

Scalability Patterns on highscalability.com

Example: Designing Distributed Systems,
PATTERNS AND PARADIGMS FOR SCALABLE, RELIABLE SERVICES
Brendan Burns

Pattern Presentation Form

- **Context**: The context describes the situation, a developer is in, when he chooses to apply this pattern.
- **Problem**: The specific problem that needs to be solved.
- **Forces**: Considerations (controversial) that must be taken into account when choosing a solution to a problem.
- **Solution**: The proposed solution to the problem.
- **Structure**: The structure section gives a graphical overview about the proposed solution.
- **Participants and Responsibilities:** components involved in the solution and their functions.
- **Strategies:** abstract discussion of implementation details.
- **Consequences**: The situation after the pattern
- **Source Code Example** and Application of the pattern in an open source project.

Roughly taken from Philipp Schill, Eclipse as client container for J2EE applications, HDM 2005. Schill extends a pattern form from Gerard Meszaros and Jim Doble. Last point added by W.Kriha.

Pattern Systems

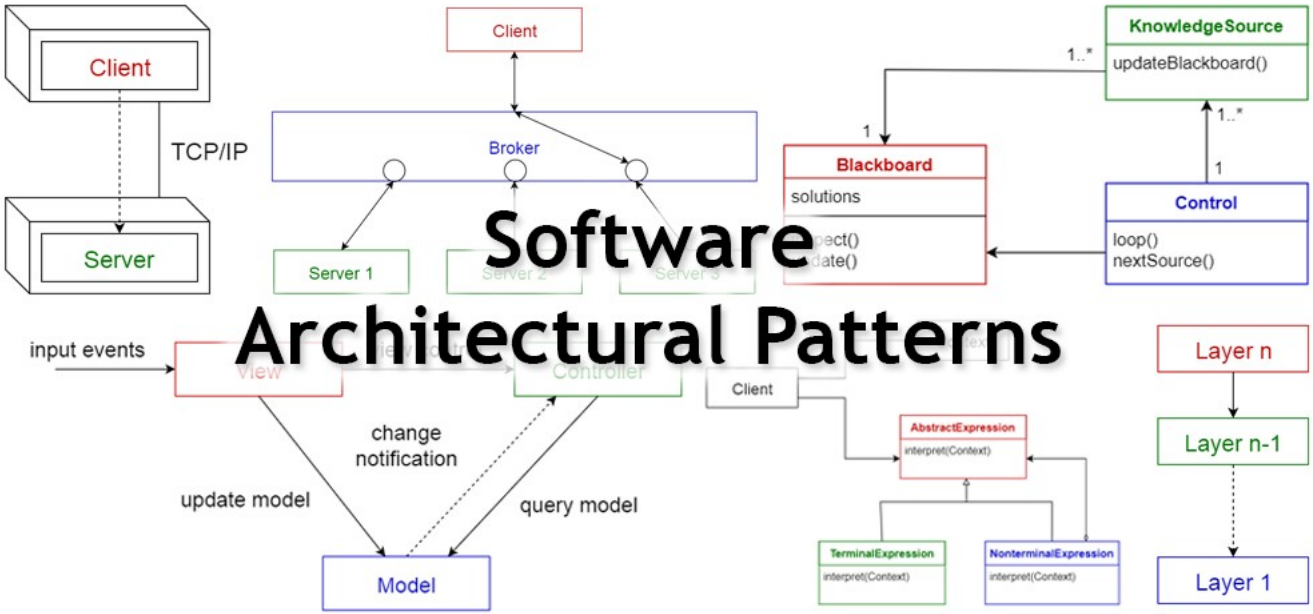
- Complex architectural solutions require patterns to collaborate.
- Here the interaction of patterns and the intended effect ist the most important thing.

See resources: „A pattern system“

Pattern Types

- **Behavioral**: mostly geared towards runtime flexibility and complex interactions (e.g. proxy)
- **Creational**: control over the lifecycle of objects – indirectly determining behavior of applications and providing extensibility of architectures (e.g. factory)
- **Structural**: determining the main structural units of an architecture (e.g. composite object)
- **Architectural** patterns: patterns important enough to define the basis of a complete architecture (e.g. Model-View-Controller, blackboard)
- **Class** based: a pattern working through classes
- **Instance** based: a pattern working through instances

See resources: „A pattern system“ and the GOF book or „Head first design patterns“



From: https://towardsdatascience.com/10-common-software-architectural-patterns-in-a-nutshell-a0b47a1e9013

Game Patterns

Technical

- Speed-Patterns
- Data-Driven
- Compiled/dynamic
- Game-Loop
- ...

Narrative

- Boss-Enemy
- Portals/Doors
- Power-Extension:Find stuff
- Groups/clans

Programming-Patterns, Robert-Nystrom,
gibt's auch kostenlos unter
http://gameprogrammingpatterns.com/contents.html
Mark Rosenfeld, The Planet Construction Kit

Example: From Observer to EAI

The observer pattern is used in Mode-View-Controller architectures as well as many others. It evolves frequently into public-and-subscribe patterns based on message oriented middleware. Finally it can evolve into Enterprise-Integration Patterns used in Enterprise Integration and Enterprise Service Bus Technology.

See Resources: IBM paper on observer, Gregor Hohpe on EAI Patterns.

High Scalability, High Availability, and High Stability Back-end Design Patterns

An updated and curated list of selected readings to illustrate High Scalability, High Availability, and High Stability Back-end Design Patterns. Concepts are explained in the articles of notable engineers (Werner Vogels, James Hamilton, Jeff Atwood, Martin Fowler, Robert C. Martin, Tom White, Martin Kleppmann) and high quality reference sources (highscalability.com, infoq.com, official engineering blogs, etc). Case studies are taken from battle-tested systems those are serving millions to billions of users (Netflix, Alibaba, Flipkart, LINE, Spotify, etc). By Benny (Quoc-Binh) Nguyen, 2018

https://github.com/binhnguyennus/awesome-scalability

Beyond Patterns: Components

Developers need to understand patterns and construct them from scratch again and again. Is there a way to build patterns as components – ready to use? This requires advanced customization abilities beyond what many OO languages provide currently. A central mechanism for customization is based on functional languages: the higher-order function (delegate, agent).

See Resources: Arnout and Meyer, Arnout thesis

Source Example: http://se.ethz.ch (visitor with closures)

Non-Software Patterns

- Social patterns (development, organisation, anti-patterns)
- Narrative patterns (how to tell a story)
- Movie patterns (the patterns used in movies to convey meaning)
- Usability patterns: what makes things usable (car design)
- Network patterns (what makes networks and devices fast and flexible)
- Psychological Patterns (anti-patterns, mistakes)
- Political patterns (CYA, etc.)

See resources: „Anti-patterns“, security patterns, Dörner, etc..

Creativity Patterns

Emiliy Buder, David Mamet Reveals Why Movies Don't Need Dialogue and More No-Nonsense Screenwriting Lessons

Christopher Alexander, Generative Sequences

See resources: „Anti-patterns“, security patterns, Dörner, etc..

Psychological Patterns (1)

- Die Logik des Misslingens – Wie wir uns selber aufs Kreuz legen, Dieter Dörner
- Psychology of Security – what makes us mis-judge risks and costs. How to manipulate people through risk statements and actions
- The political psychology of terror alarms – how to manipulate people into a constant state of fear and use it. (Today is „yellow“ danger for terrorist attacks). Philip G. Zimbardo, http://www.apa.org/about/division/terrorism.html. Shows how - based on the behavioral patterns mentioned by Schneier - politicians successfully manipulate people through the media.
- Spy the Lie: Former CIA Officers Teach You How to Detect Deception Paperback – July 16, 2013 by Philip Houston et.al.
- Daniel Kahneman, Thinking fast and slow

See resources: Dörner, Schneier, Zimbardo

Psychological Patterns (2): Cold Reading

November 14, 2007 The Sham of Criminal Profiling

Malcolm Gladwell makes a convincing case that criminal profiling is nothing more than a "cold reading" magic trick. A few years ago, Alison went back to the case of the teacher who was murdered on the roof of her building in the Bronx. He wanted to know why, if the F.B.I.'s approach to criminal profiling was based on such simplistic psychology, it continues to have such a sterling reputation. The answer, he suspected, lay in the way the profiles were written, and, sure enough, when he broke down the rooftop-killer analysis, sentence by sentence, he found that it was so full of unverifiable and contradictory and ambiguous language that it could support virtually any interpretation.

Astrologers and psychics have known these tricks for years. The magician Ian Rowland, in his classic "The Full Facts Book of Cold Reading," itemizes them one by one, in what could easily serve as a manual for the beginner profiler. First is the Rainbow Ruse -- the "statement which credits the client with both a personality trait and its opposite." ("I would say that on the whole you can be rather a quiet, self effacing type, but when the circumstances are right, you can be quite the life and soul of the party if the mood strikes you.")

From Schneiers Cryptogram newsletter. Go and check out the Astro Channel – can you detect the patterns of cold reading?

Cold Reading Patterns and their names

„They had been at it for almost six hours. The best minds in the F.B.I. had given the Wichita detectives a blueprint for their investigation. Look for an American male with a possible connection to the military. His I.Q. will be above 105. He will like to masturbate, and will be aloof and selfish in bed. He will drive a decent car. He will be a "now" person. He won't be comfortable with women. But he may have women friends. He will be a lone wolf. But he will be able to function in social settings. He won't be unmemorable. But he will be unknowable. He will be either never married, divorced, or married, and if he was or is married his wife will be younger or older. He may or may not live in a rental, and might be lower class, upper lower class, lower middle class or middle class. And he will be crazy like a fox, as opposed to being mental. If you're keeping score, that's a **Jacques Statement**, two **Barnum Statements**, four **Rainbow Ruses**, a **Good Chance Guess**, two predictions that aren't really predictions because they could never be verified -- and nothing even close to the salient fact that BTK was a pillar of his community, the president of his church and the married father of two.“ Posted on November 14, 2007 at 06:47 AM by Bruce Schneier (Cryptogram)

See why the names of patterns are so important?

Organizational Patterns

Project Management Patterns:

- The Deadline: A Novel About Project Management von Tom DeMarco von B&T
- Peopleware, Tom DeMarco, Tim Listner
- Fred Brooks, The mystical man-month
- Ed Yourdon, Deathmarch projects
- Design Patterns for Managing Up KATE MATSUDAIRA, Four challenging work situations and how to handle them

Organizational patterns

- The peters principle

The embarrassing, sad and dangerous truth behind projects and their (incompetent) management. Or dig into organizational sociology for amazing finding on how organizations act, find their goals etc. Ever wondered what VDK will do in the future?

Political Patterns (1)

•the icebreaker (foot-in-the-door)- get a foot in the door and wait for opportunities to push the door further open

•the adjusting screw (Stellschraube): get a quantitive mechanism in place which you can adjust piece by piece (mehrwertsteuer, LKW-Maut)

•the god on a stool (götze): pick one factor of many and turn it into a god. Stick to it by all means for a couple of years and then let it drop quickly. replace it with a different god (often used in economics, e.g. in outsourcing, shareholder value etc.)

•mc-kinsey pattern: divide and rule by manipulating your underlings (also called pit-bull education) Some useful mobbing included

•late-moment-blitzkrieg: that is how lobbyists change laws at the last minute of their preparation. Gives others no chance to respond or react (like members of parliament) and avoids extended press coverage of lobbyist-reasons and arguments (which always sound somehow lame and egoistic). frequently used by members and organizations in the health care professions. Lately successfully applied at so called health reforms

•A beautiful noise: a blunt pattern: replace the word tax raise with reform.

•berlusconi: always make sure that in a discussion the guy representing your position gets the last word to the public. Easier if you control the media.

•The long way to war: e.g. to get germans agree to war again. Uses successfully :The good cause (humanitarian aid), the-others-are-calling-us (or we can't help it pattern) and ends with: higher forces, we cannot withdraw now, approved by parliament etc.

Political Patterns (2)

•Feedback loop, Double goal: uses successfully also (circular argument): We need to fight terrorism in XYZ, this causes dangers of terrorism in germany,we need to stock up on weapons and spend more on the army.
•we need more internal control of our citizens.

•A complex strategy/political architecture: a combination of patterns to achieve various goals

•Blitzkrieg plus hidden baggage: Republican Cong. F. James Sensenbrenner of Wisconsin did just that. In
•February 2005, he attached the Real ID Act to a defense appropriations bill. No one was willing to risk not supporting the troops by holding up the bill, and it became law. No hearings. No floor debate. With nary a whisper, the United States had a national ID.

•security theater: how to create the illusion of security in people. Often combined with getting a lot of money for it.
•Baby RFIDs, tamper proof packaging, Legal protection through sec. Theater, Law and economic effects on everyday life.
•NASA: astronaut scandal and screening demands. FedEx refuses to ship empty containers: security theater at its finest.
•http://putative.typepad.com/putative/2007/01/fedex_refuses_s.html . Sir Ken Macdonald -- the UK's "director of public prosecutions" - has spoken out against the "war on terror" rhetoric: http://politics.guardian.co.uk/terrorism/story/0,,1997247,00.html

•moral disguise (http://news.google.de/news/url?sa=t&ct=de/0-0-0&fp=45dd69f2146611b4&ei=qpndRabHOrPywQHUvaToDA&url=http%3A//www.handelsblatt.com/news/Politik/Deutschland/_pv/_p/200050/_t/ft/_b/1228952/default.aspx/bischof-attackiert-von-der-leyen.html&cid=1103987009)
•child care discussion

Political Patterns (3)

- Leo Löwenthal, Falsche Propheten (1949). Demagogie und Populismus Pattern,

  https://www.koellerer.net/2018/11/17/leo-loewenthal-falsche-propheten-studien-zur-faschistischen-agitation/

City Patterns

Scaling: The surprising mathematics of life and civilization
By Geoffrey West, Distinguished Professor and Past President, Santa Fe Institute
https://medium.com/sfi-30-foundations-frontiers/scaling-the-surprising-mathematics-of-life-and-civilization-49ee18640a8

Cities Are Innovative Because They Contain More Ideas to Steal, Jun 12, 2013 By Eric Jaffe,
https://www.citylab.com/life/2013/06/cities-are-innovative-because-they-have-more-ideas-steal/5883/

Bijlmer City (Amsterdam) https://99percentinvisible.org/episode/bijlmer-city-future-part-1/ and:
https://99percentinvisible.org/episode/blood-sweat-tears-city-future-part-2/

Cities: Engines of Innovation
Most of humanity now lives in a metropolis. That simple fact helps to fuel our continued success as a species
By Edward Glaeser https://www.scientificamerican.com/article/engines-of-innovation/

A Physicist solves the City,
Jonah Lehrer, http://www.nytimes.com/2010/12/19/magazine/19Urban_West-t.html

Bio-Computing Patterns

Everything in biological systems depends on everything else; all variables are global, all methods are public, and all definitions are recursive. It's all spaghetti code- there is nothing like the separation of functionality that we impose on our designed artefacts for the sake of our limited minds. A signal needed to be sent from one place to another, and the brain has the machinery for endocrine functions already, so why not have it handle it? (Maxander, https://news.ycombinator.com/item?id=16589702)

Brainless Embryos Suggest Bioelectricity Guides Growth. Researchers are building a case that long before the nervous system works, the brain sends crucial bioelectric signals to guide the growth of embryonic tissues. (https://www.quantamagazine.org/brainless-embryos-suggest-bioelectricity-guides-growth-20180313/)

What are the patterns biology uses to design life? Are design and architecture patterns universal? Software patterns divide and conquer complex problems by compartmentalization, abstraction, isolation etc. Is this universal? See also: Biologists Home in on Turing Patterns, https://www.quantamagazine.org/biologists-home-in-on-turing-patterns-20130325/

Anti-Patterns

- Big Ball of Mud
- Golden Hammer
- Analysis/Paralysis
- Linear Thinking
- The Guru

…..

Design “Types”

“design-types.net describes a system of 16 design types characterizing developers in the aforementioned way. Each design type links back here to those principles each type favors or disregards. There is a questionnaire for testing yourself, a questionnaire for assessing colleagues and some statistics about those who already took part. “

http://www.principles-wiki.net/
http://www.design-types.net/index.html

Patterns: A Critical View

- Patterns are not the real thing: abstractions and concepts are needed
- Patterns hide defects in languages
- Patterns lead to fixing problems without thinking about the problem first (the “why” tends to get lost)
- Patterns just move problems to different spots (e.g. flexibility patterns create the need for meta-data)

http://blog.ircmaxell.com/2013/09/beyond-design-patterns.html

What of Patterns and Pattern Languages?

- The Fundamental Process needs some idea of what is being built:
  - e.g. for a fireplace you need a firebox, a fireback, splayed sides, a hearth, a throat, a smoke shelf, and a chimney
- What you are building has a cultural component because of how cultures have come to live:
  - tea for an Englishman involves sitting on chairs
  - tea for an Indian involves sitting on the floor
- Therefore one needs a set of generic centers
- These generic centers form the pattern language for the project

The essence of it is that the generic centers must unfold from the culture.

What of Patterns and Pattern Languages?

There was always one great difficulty with the theory of pattern languages, and with the languages my colleagues and I, and others, published. Where did the patterns come from?

Much of our early work implicitly made use of the idea that good patterns were to be derived, somehow, from existing culture thus ensuring a relation to the subtleties of culture variation, and preserving things that were good and important, which had been swept aside in the onrush of techno-civilization. But there was always hanging over this process, a sword of Damocles. If—as a procedure—one takes the patterns from existing culture, then one merely reiterates what is being built. That is not necessarily *good.*

The unfolding process takes existing cultural patterns and moves the culture forward.

Christopher Alexander, The Nature of Order,
https://www.dreamsongs.com/Files/NatureOfOrder.pdf

Tasks for the next session

1. Read the first 80 pages of „Design Patterns“ by Erich Gamma, John Vlissides at al.
2. or: Read the introduction from James W.Cooper, The Design Patterns Java Companion, a FREE book on java design patterns. http://www.patterndepot.com/put/8/JavaPatterns.htm . Don‘t forget to download the java examples!
3. Or read the introduction to „Head First Design Patterns“ – the new book by Oreilly. I was told it is the most accessible of all pattern books.
4. Prepare for a general discussion and clarification session on patterns (next time)
5. Start thinking about which architectural problem you want to solve and which pattern or pattern system you would like to present to us as a solution.

In any case: Prepare yourself for the fact that from now on any discussion at HDM regarding architecture, design or implementation of software artefacts will be pattern based – as it is in many good software companies. You will be unable to work in better teams without knowing your patterns.

Resources (1)

Erich Gamma et. all. , Design Patterns (The „classic“. Beginners should read at least the first 80 pages for an introduction. The rest is a collection of the most-used patterns. No architectural or composite design patterns.

- Pattern Language of Program Design (PLOP) book 1-3. Book 2 has a collection of patterns including distributed systems. Every book very good. Book 3 covers architectural patterns like „bureacracy“ which are actually composite patterns (patterns working together – not to confuse with the single composite object pattern)
- The siemens book on „A pattern system“
- Idioms are „little“ patterns: see Jim Coplien on C++ idioms – one of the very sources of the pattern movement.

Resources (2)

- www.c2.org , the famous pattern „wiki“ of the two Cunninghams – At least check it out.
- Portland pattern repository – one of the most important platforms for patterns and their discussions
- Euro-Plop: a yearly conference for pattern afficionados.
- Patterns for distributed computing, Buschmann et.al.
- http://www.meurrens.org/ip-Links/java/designPatterns/ Java design patterns but also a good introduction and link collection.
- Implement a Data Access Object pattern framework
  http://click.idg.email-publisher.com/maaah3RaaRlW0a9JUqkb/
- Design patterns let you cache SOAP services and improve performance http://click.idg.email-publisher.com/maaaisaaaRqJza9JUqkb/
- http://www.enteract.com/~bradapp/docs/patterns-intro.html A wonderful introduction with lots of links.
- http://www.security-patterns.de patterns related to all kinds of security problems. Like a portal.

Resources (3) From observer to EAI, ESB

- http://www6.software.ibm.com/developerworks/education/j-delivery/index.html covers all kinds of event delivery patterns from observer to message oriented middleware. You need to register at www.ibm.com/developerworks first - something you should do anyway to get their newsletters and excellent articles

- Gregor Hohpe, http://www.enterpriseintegrationpatterns.com/ Enterprise Integration Patterns (very important patterns for EAI problems in large enterprises (asynchronous messaging etc.) Very good articles at his homepage.

  http://www.enterpriseintegrationpatterns.com/docs/jaoo_hohpeg_enterprise/integrationpatterns.pdf

- Patterns of Distributed Systems, https://martinfowler.com/articles/patterns-of-distributed-systems/
-

Resources (4)

- Martin Fowler, Patterns of Enterprise Application Architecture. Very good for large scale design. Find a draft online at: http://www.martinfowler.com/isa/ (also good stuff on distributed computing patterns)

- IBM e-business patterns (also for less-technical people like CEOs etc.) http://www-106.ibm.com/developerworks/patterns/ IBM tries to categorize the endless ways to build e-applications into few business, application and runtime patterns.

- James W.Cooper, The Design Patterns Java Companion, a FREE book on java design patterns from a true IBM Watson guy. http://www.patterndepot.com/put/8/JavaPatterns.htm . Don‘t forget to download the java examples!
- Game Programming Patterns von Robert Nystorm

Resources (5)

- ::: A taste of "Bitter Java" :::

Design patterns are important to software development as witnessed by the amount of coverage they get in the technical trade press, but as useful as they are in the development proceess, design patterns solve only half the puzzle. Antipatterns -- which describe "a commonly occurring solution to a problem that generates decidedly negative consequences" -- seek to address the other half by showing Java programmers how they can be used to avoid common Java traps. In this article, antipatterns expert and noted author of Bitter Java, Bruce Tate, demonstrates how and why antipatterns are a necessary and complementary companion to design patterns. He provides concrete examples of antipatterns -- the Round-Tripping antipattern and the Magic Servlet antipattern -- and describes how to apply the knowledge to improve your programs and your development process.
http://www-106.ibm.com/developerworks/library/j-bitterjava/?n-j-3212
BTW: Bitter java itself can be downloaded for free!

Resources (6)

- Assorted links to J2EE patterns: http://www.javaworld.com/javaworld/jw-06-2002/jw-0607-j2eepattern.html and http://www.javaworld.com/javaworld/jw-01-2002/jw-0111-facade.html
- The Publisher-Subscriber pattern reduces object dependencies for flexible UI design http://www.javaworld.com/javaworld/jw-11-2001/jw-1109-subscriber.html?
- Improve your application's workflow with the Dispatcher design pattern and XSL http://www.javaworld.com/javaworld/jw-10-2001/jw-1019-dispatcher.html
- Server Component Patterns, Markus Völter et.al. Describes the patterns behind .NET and J2EE container technology.
- Stahl, Völter, Modellgetriebene Software-Entwicklung. Design patterns für MDD. There is a public paper in english that covers most MDD patterns.
- AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis

Resources (7)

- The current buzzword is "software product line" (see http://www.sei.cmu.edu/SPLC2/) though "domain specific language" also seems to be a current term.
- Component-Based Product Line Engineering with UML by Colin Atkinson, Joachim Bayer, Christian Bunse, Erik Kamsties, Oliver Laitenberger, Roland Laqua, Dirk Muthig, Barbara Paech, Jurgen Wust, Jorg Zettel.
- The bible of generative computing: Eisenecker/Cernecky: Generative Computing (covers all kinds of „meta“ programming like aspect-oriented, generative etc.) Really good if sometimes a bit dry.
- The bible of product lines: Paul Clements, Linda Northrop, Software Product lines: Practices and Patterns
- For the up-and-coming realtime developments: go to ilogix.com for patterns.

Resources (8)

- David C. Hay, Data Model Patterns. Software developers usually underestimate the consequences of bad data models. Hay shows patterns for flexibility that allow new and extended uses without major data reorganization. Why shouldn‘t there be patterns in modelling as well? Do we have UML patterns already?

- Data Access Patterns, Clifton Nock.

- How to find (recover) patterns and an introduction to product lines: Ilka Philippow et.al., Design Pattern Recovery in Architectures for Supporting Product Line Development and Application

- On Pattern Languages and Christopher Alexander: Martin Gorlt, Christopher Alexanders Muster Sprache, www.kriha.de/krihaorg/dload/uni/designpatterns/gorlt_pattern_language.pdf

Resources (9)

- Dieter Dörner, die logik des Misslingens – Anti-Patterns im täglichen Verhalten. Exzellente Darstellung wie wir uns selber aufs Kreuz legen.

- Bruce Schneier, The Psychology of Security – Draft Feb. 28 2007, see www.schneier.com. On basic patterns of risk management (and mis-management). How humans mis-judge risks and costs due to their evolutionary history.

- Fred Spiessens, Patterns of safe collaboration. Thesis 2007. Develops patterns for secure software components like caretaker. Shows how to model and model-check those patterns and complete designs (SKOLL, SKOLLAR) http://www.info.ucl.ac.be/~fsp/fsp_thesis.pdf

- Daniel Kahneman, Thinking Fast and Slow. Watch your own instincts!

- Workflow Patterns, Aalst et al., http://is.tm.tue.nl/research/patterns/patterns.htm

- Pattern-Oriented Software Arch., Vol. 2-5, Buschmann et.al. 2007

- Agile Anti Patterns: https://www.infoq.com/presentations/agile-anti-patterns-mitigate/?utm_source=email&utm_medium=editorial&utm_campaign=SpecialNL&utm_content=05072020&forceSponsorshipId=2016

Resources (10): Tutorials

Patterns-based design and development for architects, Part 1: Using design patterns (Architecture)

Learn the design problems that can arise while architecting a system, and how you can use design patterns to solve these problems and improve your design.

http://www.ibm.com/developerworks/edu/ar-dw-ar-designpat1.html

Using model-driven development and pattern-based engineering to design SOA: Part 2. Patterns-based engineering

25 Sep 2007
http://www.ibm.com/developerworks/edu/dw-rt-umlprofiles2.html

From the IBM developerworks newsletter.

Resources (11) PHP/AJAX

-Five common PHP/AJAX Patterns

http://www.ibm.com/developerworks/library/os-php-designptrns/

http://www.ibm.com/developerworks/library/x-ajaxxml2/

Resources (11) Repositories

on http://sourcemaking.com/design-patterns-and-tips 100 of the most popular software-development patterns (including Java-Sourcecode examples and excellent diagrams) – thanks to Marc Seeger for pointing me to this repository

Resources (12) Beyond Patterns

Bertrand Meyer, Software Architecture vs. Functional Languages, in: Beautiful Architecture, by Spinelli and Gousios

Arnout and Meyer with an oberserver pattern replacement

Multicore-Patterns – how to deal with massively parallel systems

Service Design Patterns (new book available...) On Rest and web service patterns