Pattern-based Design Research (PDR)
29.06.2016, Prof. Dr. Florian Matthes

Software Engineering für betriebliche Informationssysteme (sebis)
Fakultät für Informatik
Technische Universität München

wwwmatthes.in.tum.de
1. Pattern-based Design Research (PDR)
Using patterns as a research group

**Informatics**
- Identify Pattern Language
- Explain Patterns with Theories

**Engineering**
- Select & Apply Patterns

**Evaluation**
- Select & Apply Patterns

**Informatics Models**
- Identify Patterns
- Define & Enhance Patterns

**Abstraction**
- Engineering

**Application Domain**
- Application

**Prototypical Solutions**
- Practical Experience

**Information & Communication Technology**
- Application
Rigor and relevance are essential for design research.

Design research has to account for rigor and relevance
- Rigor can be achieved by applying sound methodologies
- Relevance can be achieved by addressing practitioners' needs
- The designed artifact has to be evaluated

Practitioners play an important role in design research
- Problems occur in practice
- Cooperation during solution design (e.g. in action research)
- Artifact evaluation needs a proper context

Research projects with industry need to overcome several challenges
- The project is subject to the partnering organization’s pace
- The partnering organization might demand an early delivery of results
- Different requirements for methodological rigor
- Different level of abstraction (problem and solution)
Potential benefits of PDR

Design Patterns and Pattern-based Design Research

- provide intermediary artefacts and abstractions for incremental research
- enable an iterative mode of research and evaluation
- lead to an early availability of building blocks (short “time to market”)
- create a foundation for the long-term communication between and within academic and industry expert communities
- serve as a foundation to generate new hypotheses
- incorporate knowledge and experience already available in industry
- can be used to identify anti-patterns
A pattern is a general, reusable solution to a common problem in a given context

Alexander et al. [Al77] (Architecture)

- Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.
- Each pattern is a three-part rule, which expresses a relation between a certain context, a problem and a solution

Buschmann et al. [Bu96] (Software Architecture)

- A pattern for software architecture describes a particular recurring design problem that arises in specific design contexts, and presents a well-proven generic scheme for its solution. The solution scheme is specified by describing its constituent components, their responsibilities and relationships, and the ways in which they collaborate

Gamma et al. [Ga94] (Software Engineering)

- Descriptions of communicating objects and classes that are customized to solve a general design problem in a particular context.
An enterprise architecture management pattern (EAM pattern) is
- a general, reusable solution to a common problem
- in a given context,
- identifies driving forces,
- known usages and
- consequences.

An EAM pattern takes a holistic perspective
- It address problems at the enterprise (systems of systems) level.
- It considers social, technical and economic forces in a balanced manner.
- It is discovered in working solutions rather than being invented or hoped for.
- It uses a clear, accessible, and informal language that allows practitioners to describe their knowledge and experience.
Example: Building pattern

**Problem:** How to attract people to visit a certain house?

**Solution:** *Opening to the street.* The wall along the street is made essentially out of glass, and the view in is of some inviting activity.

---

**Example:** Software architecture pattern

**Problem:** How to structure components of an information system?

**Solution:** Three Tier Architecture

Example: Design pattern

Problem: Compose objects into tree structures to represent part-whole hierarchies and discriminate between leaf-nodes and branches.

Solution:

Theorizing in IS is challenging

Design theories
- Provide knowledge support to design activities
- Are considered as theorized practical knowledge
- Require a close cooperation of scientists and practitioners
- Must account for rigor and relevance

The expected level of abstraction in the research outcome is determined by the community that is conducting the research.

---

Patterns can be viewed as candidate building blocks for theory building.

<table>
<thead>
<tr>
<th>Constituents of a design theory (Gregor and Jones, 2007)</th>
<th>Pattern contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Justificatory knowledge</td>
<td>(Related pattern)</td>
</tr>
<tr>
<td>Purpose and scope</td>
<td>Context and problem</td>
</tr>
<tr>
<td>Principles of form and function</td>
<td>Solution and forces</td>
</tr>
<tr>
<td>Artifact mutability</td>
<td>“Refined by” relationship of a pattern language</td>
</tr>
<tr>
<td>Principles of implementation</td>
<td>Guidelines for pattern selecting in a pattern language</td>
</tr>
<tr>
<td>Expository instantiation</td>
<td>Example in pattern description</td>
</tr>
<tr>
<td>Testable hypotheses</td>
<td>Pattern (only testable solution model hypotheses)</td>
</tr>
</tbody>
</table>

Situational method engineering
Identify methods suitable for a specific situation

There is no method that fits all problems.

See also:
*The Logic of Failure* (D. Dörner 1989, 1997)
Linking competing solutions in a design theory nexus instantiation

Design theory nexus

- Address the challenge of wicked problems
- Connects alternative design theories

1. Identify approaches
2. Analyze approaches
3. Formulate assertions
4. Develop decision making process
5. Develop tool

Components of a design theory nexus

1. **Observe & conceptualize**
   - Use patterns to document good practice solutions
   - Apply rule of three
   - Use grounding theories to elicit a pattern terminology for the application domain
   - **Result:** pattern candidates, or patterns
2. Pattern-based theory building & nexus instantiation

- Evolution: pattern candidate, pattern, pattern language, design theory
- Patterns as elementary design principles
- Assurance of linguistic compatibility (homonyms, synonyms)
- **Result**: design theory nexus
Pattern-based Design Research (PDR)

3. Solution design & application

- A concrete solution is constructed as a situated design artifact
- Solution design is configured and adapted to fit the organizational context
- **Result**: an implemented solution

![Diagram showing the process of solution design and application](image)
Pattern-based Design Research (PDR)

4. Evaluation & learning

- Instantiated solution evolves and deviations occur
- Reasons might be changes of the goals or the context
- Deviations as source for nexus evolution
- **Result**: Ongoing evolution of the design theory nexus
Pattern-based Design Research (PDR)

Grounding theories

- observe & conceptualize
- guide & structure

Organized collection of reusable practice-proven solutions

- Design Theories
- Pattern Language
- Pattern Candidates

Solution design

- select

Configured design

- learn

△ deviations

- establish

Instantiated solution

Theory (academics)

Practice (industry)

Observations
How to transform the enterprise

Models

plan development

describe

enact

Reality

enterprise

transform

current

planned
An enterprise architecture provides a common language for business and IT.

- Technical, social, economic and legal aspects
- Layers and crosscutting concerns
- Static and dynamic relationships more important than element details
- Current, planned and target architecture
An enterprise architecture management pattern (EAM pattern) is
- a general, **reusable solution** to a common **problem**
- in a given **context**,
- identifies driving **forces**,
- **known usages** and
- **consequences**.

An EAM pattern takes a holistic perspective
- It addresses problems at the **enterprise** (systems of systems) **level**.
- It considers **social**, **technical** and **economic** forces in a balanced manner.

- It is **discovered** in working solutions rather than being invented or hoped for.
- It uses a **clear**, **accessible**, and **informal** language that allows practitioners to describe their knowledge and experience.
Motivation und Forschungsfragen

- Gibt es ein Kernmodell zur EA Dokumentation? (Klassen und Beziehungen die jeder pflegt)
- Wie werden die Klassen und Beziehungen benannt?
- Gibt es Muster bei der Evolution des Kernmodells?
- Was sind spezifische EA-Aspekte, die man außerdem in der Praxis findet?
- Wie gut ist die Datenqualität der Klassen, Attribute und Beziehungen?
- Wie wird die Datenpflege unter den verschiedenen Rollen aufgeteilt?
- Gibt es eine automatisierte Datenerhebung/-integration?
- Wie zufrieden sind Unternehmensarchitekten heute mit der EA Datenqualität?
- Kann eine bestehende Dokumentationspflicht durchgesetzt werden?
- Welche konkreten Visualisierungen/Kennzahlen/Berichte kommen zum Einsatz?
  - Wer nutzt diese?
  - Wie oft werden diese aktualisiert?
## Uniform structure for EAM patterns

<table>
<thead>
<tr>
<th>Name</th>
<th>Short and sound name of the pattern.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short Description</td>
<td>Short summary of the pattern to get a first look at its content.</td>
</tr>
<tr>
<td>Example</td>
<td>An example illustrating the problem to be addressed by the pattern.</td>
</tr>
<tr>
<td>Context</td>
<td>The situations in which the pattern may apply.</td>
</tr>
</tbody>
</table>
| Problem | The problem a pattern addresses, including a discussion about its associated forces.  
- Only one problem per pattern  
- Forces: Goals and constraints, which occur in the context |
| Solution | The fundamental solution principle underlying the pattern |
| Implementation | Guidelines for implementing the pattern. E.g. the need to introduce a special board |
| Variants | A brief description of variants or specializations of a pattern. |
| Known Uses | Examples where the pattern has been used.  
- Usage in companies  
- Usage in tools  
Also Known As: Alternative names for the pattern (if any). |
| Consequences | The benefits the pattern provides, and any potential liabilities |
| See Also | References to other patterns solving similar problems, and to patterns that help to refine the pattern under consideration |
| Credits | Credits to other authors, reviewers and shepherds of the pattern. |
A pattern language is a system of interrelated patterns

<table>
<thead>
<tr>
<th>Pattern Relationship</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Used by</td>
<td>A smaller pattern is used by a larger pattern.</td>
</tr>
<tr>
<td>Refined by</td>
<td>A general pattern is refined by a specific pattern.</td>
</tr>
<tr>
<td>Variant</td>
<td>A variant pattern refines a more well-known pattern.</td>
</tr>
<tr>
<td>Variant uses</td>
<td>A variant pattern of one pattern uses another pattern.</td>
</tr>
<tr>
<td>Similar</td>
<td>A pattern is similar to another pattern.</td>
</tr>
<tr>
<td>Combines</td>
<td>Two patterns combine to solve a single problem.</td>
</tr>
<tr>
<td>Requires</td>
<td>A pattern requires the solution of another pattern.</td>
</tr>
<tr>
<td>Tilling</td>
<td>A pattern uses itself.</td>
</tr>
<tr>
<td>Sequence of Elaboration</td>
<td>A sequence of patterns from the simple to the complex.</td>
</tr>
</tbody>
</table>

From patterns and reference methods to practice-proven, configurable, modular building blocks.

**Pattern**
- Extract + Theory
- Observe

**Building block**
- Fragment

**Reference method/model**
- Monolith
  - Evaluate
  - Apply

**Ad-hoc method**
- Apply
Method building blocks (MBBs) help to eliminate redundancies in the descriptions of EA methods.

**Tasks:**
- The enterprise architect creates an interview guideline.
- The interviewer conducts interviews based on the guidelines and consults the information steward to gather information.
- Afterwards, the enterprise architect puts the results in the repository.

**Force**
- Telephone vs. face-to-face
BEAMS uses building blocks to configure a situational EA management function.
The fitness of a method BB depends on the given organizational context.

- Example: Method BB for “develop & describe”

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Central IT</td>
<td>+</td>
<td>0</td>
<td>+</td>
</tr>
<tr>
<td>Federal IT</td>
<td>+</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Decentral IT</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Central CMDB</td>
<td>0</td>
<td>0</td>
<td>+</td>
</tr>
</tbody>
</table>

- Select Method BB with best fit

BEAMS makes tacit expert knowledge explicit
- Collected from pattern catalog, workshops, online questionnaire, and existing literature
EAM method building blocks (2012)

**Develop & Describe**
The activity develop & describe covers MBBs for developing current, planned states and target states of the EA. Furthermore, MBBs for the development of architectural principles that guide the future evolution of the EA, are provided. Similarly, MBBs for maintaining descriptions of the different EA states are provided.

- Approve architecture description
- Derive projects from target state
- Describe automatically by crawler
- Describe by central repository
- Describe by interview
- Describe by questionnaire
- Describe by workshop
- Develop planned states of the EA
- Develop target state based on business strategy
- Develop target state based on IT strategy
- Develop target state based on potential for improvement
- Develop target state in strategy board
- Document lessons learned
- Ensure information consistency

**Communicate & Enact**
The communicate & enact activity contains MBBs for communicating EA artifacts and enacting EA plans. Various ways to perform this activity exist, ranging from fairly non-infering ways of informing via enacting, i.e. consulting and conducting workshops, to enforcing, i.e. punishment for deviating from existing plans.

- Caution deviating projects
- Control adherence to conditions
- Educate by in-house training
- Impose conditions for deviations
- Officially gratify standard conformance
- Provide financial rewards
- Publish architectural description
- Request acknowledgement of architectural description
- Require justification for deviations
- Review architectural compliance
- Tax non compliant projects
- Veto non compliant projects

**Analyze & Evaluate**
The analyze & evaluate activity provides MBBs for different kinds of analysis, which can be performed on one state of the EA, e.g. the current state to identify potential for improvement as well as measure the achievements of objectives, or gap analysis between different states of the EA, e.g. to evaluate different planned states.

- Aggregate analysis results based on prioritization
- Control adherence to business capabilities
- Control adherence to conditions
- Develop by group discussion
- Multi expert evaluation
- Pattern-based analysis
- Perform single expert evaluation
- Quantitative assessment
You want to know which business applications support which business processes at which organizational units.

**Explanation:**
The business processes conducted in the enterprise are supported by according business applications at different organizational units. This represents a ternary relationship between the three concepts.

**EA information model:**

Open model in new window

<table>
<thead>
<tr>
<th>Area of interest</th>
<th>Business &amp; Organization Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Application &amp; Information Layer</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EA concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Application</td>
</tr>
<tr>
<td>Organizational Unit</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SuperConcern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Applications support Business Processes for Products at Organizational Units</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Applications support Business Processes at Organizational Units</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mapping SyCaStore to BEAMS IBBs</td>
</tr>
<tr>
<td>Combine BEAMS with ArchiMate</td>
</tr>
<tr>
<td>Fictitious example</td>
</tr>
</tbody>
</table>
Example: Your EAM goal is to “improve compliance of projects with EA principles”

The BEAMS method library provides several method building block variants

1. Financial reward for conformance to EA principles
2. Request justification for deviations
3. Request financial reserves for deviations from EA principles
4. Power of veto for enterprise architects

Governance structure of your enterprise

- Centralized IT
- EA architect participates in project portfolio management

Organizational context of your enterprise

- EA management team has no budget power
- Business can overrule EA project portfolio recommendations

=> BEAMS method base recommends variant (3) for this situation
Recent EAM pattern project

2014: EA Visualization Tool Survey

- **Vendors**: 18
- **Tools**: 19
- **Practitioners**: 109
- **Visualization types**: 26

![Visualization Types Chart]

- Visio (Microsoft)
- Powerpoint (Microsoft)
- Excel (Microsoft)
- Enterprise Architect (Sparx Systems Pty Ltd)
- ARIS IT Architect & Designer (Software AG)
- iteraplan (Iteratec GmbH)
- Archi (Institute of Educational Cybernetics)
- planningIT (alfabet AG)
- Rational Suite (IBM)
- Troux Enterprise Portfolio Management Suite (Trox Technologies)
- ADOit (BOC)
- iServer Enterprise Architect (Orbus Software)
- MEGA Solution for Enterprise Architecture Suite (MEGA International)
- Tricia (Infoasset)
- The Essential Project (Enterprise Architecture Solutions Ltd)
- PowerDesigner (SAP-Sybase)
- Eclipse Agile EA (The Eclipse Foundation)
- Casewise Modeler Suite (Casewise Ltd)
- Other

![Vendor List]

- Casewise Modeler Suite (Casewise Ltd)
- Enterprise Architect (Sparx Systems Pty Ltd)
- PowerDesigner (SAP-Sybase)
- Troux Enterprise Portfolio Management Suite (Trox Technologies)
- iServer Enterprise Architect (Orbus Software)
- MEGA Solution for Enterprise Architecture Suite (MEGA International)
- Tricia (Infoasset)
- The Essential Project (Enterprise Architecture Solutions Ltd)
- PowerDesigner (SAP-Sybase)
- Eclipse Agile EA (The Eclipse Foundation)
- Casewise Modeler Suite (Casewise Ltd)
- Other
More examples of pattern research…

EAM Complexity Metrics
- 6 companies (Financial Services, Automobile), Quarterly Workshops
- 6 patterns identified

Process Patterns – Building Large-Scale Systems Using Object Technology
- Large collection of process patterns
- Continuous evolution

Workflow Patterns (van der Aalst et al.)
- ~3000 citations
- 19 patterns

Enterprise 2.0 Usage Patterns
- 70 use cases & 21 patterns identified
- >40 companies
When (not) to use PDR?

- Complex design problem
- Recurring design problem
- Multiple (successful) solutions to the design problem already exist in practice
- Lack of sufficiently rich *local theories* [applying to a specific context]
- Established modeling language to describe a solution in a structured way
  - data model, object model, process model, KPI template, business model template, program, system diagram, …
  - often not yet available for disruptive innovations (e.g. systems biology)
- Access to a sufficiently large set of designers
  - willing and authorized to share their knowledge with the public
  - experienced in the respective field
Some practical hints for conducting PDR

- Use a two phase data collection process including a Pre-Test
- Make a contract with participating practitioners about knowledge usage (NDA)
- Get as much data as possible in the first run, you won’t get a second chance (but do not overwhelm practitioners)
- Do not expect large funding, participants are spending time and travel budget
- Try to establish a personal relationship with a core team of participants to ensure continuous improvement
- Let participants think in their own mental models, translation to a generic terminology / modeling notation is up to you
- Focus on experienced practitioners
- Expect to receive (allow) data in arbitrary formats
- Establishing and maintaining a living and active community is a real challenge!
Solution hints regarding EA meta model extension

Terminology to be used by the different language communities

Which associations?
Which cardinalities?
Who is responsible for which entities and associations?
Summary

- Pattern identification is common in IS research
- PDR builds upon well known research approaches
- PDR addresses challenges occurring in academic-industry cooperation
- Steps 1 – 3 have been successfully applied in the EA management domain

Future Work

- Implementation of tool support (documentation, solution design, monitoring)
- Detail on the role of the participating researcher
- Evaluate phase 4: evaluation & learning
- Evaluate PDR in other sub-disciplines of Information Systems other than EA management
Core Publications


Foundation

Examples


Thank you for your attention. Questions?