Combining Pattern Languages and Reusable Architectural Decision Models into a Comprehensive and Comprehensible Design Method

Feb 21, 2008
WICSA 2008

Olaf Zimmermann
Uwe Zdun
Thomas Gschwind
Frank Leymann
Acknowledgements

- The paper presents results from joint work of several different organizations (and communities):
  - Olaf Zimmermann (presenting) and Thomas Gschwind, Business Integration Technologies Group, IBM Zurich Research Lab, Switzerland
  - Uwe Zdun, Information Systems Institute, Technical University Vienna, Austria
  - Frank Leymann, Institute for Architecture of Application Systems, University of Stuttgart, Germany

- Special thanks to the WICSA 2008 reviewers!
Abstract

When constructing software systems, software architects must identify and evaluate many competing design options and document the rationale behind any selections made. Two supporting concepts are pattern languages and architectural decision models. Unfortunately, both concepts only provide partial support: Extensive upfront education is needed for practitioners to be in command of the full pattern literature relevant in their field; retrospective architectural decision modeling is viewed as a painful extra responsibility without immediate gains.

In this paper, we combine pattern languages and reusable architectural decision models into a design method that is both comprehensive and comprehensible. Our design method identifies the required decisions in requirements models systematically, gives domain-specific pattern selection advice, and provides traceability from platform-independent patterns to platform-specific decisions. We validate our approach by applying it to enterprise applications as an exemplary application genre and a SOA case study from the finance industry.
1. Case study
   - Broker-based integration of core banking application components
2. Existing work
   - Architectural patterns and decision capturing
3. ArchPad walkthrough
   - Refinement stages
   - Reusable Architectural Decision Model for SOA (RADM for SOA)
4. Discussion
   - Tool support
   - Validation
5. Conclusions and outlook
Agenda

1. Case study
   - Broker-based integration of core banking application components
2. Existing work
   - Architectural patterns and decision capturing
3. ArchPad walkthrough
   - Refinement stages
   - Reusable decision model
4. Discussion
   - Tool support
   - Validation
5. Conclusions and outlook
Case Study: Core Banking SOA with Web Services

Some of the Required Architecture Design Activities

- Selection of top-level integration pattern
  - BROKER from “Patterns of Software Architecture” (POSA) is an obvious choice

- POSA suggests six steps to implement the pattern:

  “(1) Define an object model. (2) Decide which type of component interoperability the system should offer, binary or Interface Description Language (IDL). (3) Specify the APIs the broker component provides for collaborating with clients and servers. (4) Use proxy objects to hide implementation details from clients and servers. (5) Design the broker component (6) Develop IDL compilers.”

  Step (5) has nine sub steps: “(5.1) On-the-wire protocol, (5.2) Local broker, (5.3) Direct communication variant, (5.4) (Un)marshalling, (5.5) Message buffers, (5.6) Directory service, (5.7) Name service, (5.8) Dynamic method invocation, (5.9) The case in which something fails”.

- This is helpful, but which (design) patterns are suited to implement the steps? What about the required technology choices? Product selections?
Agenda

1. Case study
   - Broker-based integration of core banking application components

2. Existing work
   - Architectural patterns and decision capturing

3. ArchPad walkthrough
   - Refinement stages
   - Reusable Architectural Decision Model for SOA (RADM for SOA)

4. Discussion
   - Tool support
   - Validation

5. Conclusions and outlook
Architecture and Design Patterns

- In the last couple of years, patterns have become part of the mainstream of object-oriented software development
  - Single pattern is one solution to a particular, recurring problem
  - However, “real problems” are more complex
- Different kinds of patterns:
  - Design patterns (GoF)
  - Software architecture patterns (POSA, POSA2)
  - Analysis patterns (Fowler, Hay)
  - Organizational patterns (Coplien)
  - Pedagogical patterns (PPP)
  - Many others

“Each pattern is a three-part rule, which expresses a relation between a certain context, a certain system of forces which occurs repeatedly in that context, and a certain software configuration which allows these forces to resolve themselves.”
(Coplien summary of Alexander’s definition)

Remoting Patterns

- Numerous projects
  - use, extend, integrate, customize, and build
distributed object middleware
- Goals:
  - illustrate the general, recurring architecture of successful distributed object middleware
  - illustrate more concrete design and implementation strategies
- Book: published in Wiley’s Pattern Series in 2004

This slide: Zdun U., Voelter M., Kirchner M., Remoting Patterns, JAX 2004 Tutorial
Architectural Decision Modeling (ADM) Overview

- Architectural decisions capture key design issues and the *rationale* behind a chosen solution:
  - Conscious design decisions concerning a software system as a whole, or one or more of its core components
  - With an impact on non-functional characteristics and quality factors

2) Advantages and disadvantages of available design options

- From retrospective, template-based capturing to proactive decision *modeling*
  - Decision drivers include quality attributes (forces) and architectural principles
  - Scope, phase, and role attributes provide method alignment

Patterns and Decisions in Comparison

<table>
<thead>
<tr>
<th>Characteristic/Requirement</th>
<th>Pattern Languages</th>
<th>ADMs</th>
<th>Assessment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intent and main use case</td>
<td>Capture generic architecture and design elements as reusable solutions to commonly occurring problems</td>
<td>Capture and record system-specific decisions justified by project-specific decision drivers</td>
<td>Complementary, strong relationship, pattern application must also be captured as a decision on a project</td>
</tr>
<tr>
<td>Standard description format</td>
<td>Several templates, e.g., Portland style</td>
<td>Several meta models, OMS profiles</td>
<td>Present in both approaches, overlap</td>
</tr>
<tr>
<td>Level of detail</td>
<td>Comprehensive, detailed</td>
<td>Terse, telegram style, details elsewhere</td>
<td>Patterns more detailed by definition</td>
</tr>
<tr>
<td>Top-down decomposition of problem into atomic units of design work</td>
<td>Objective of pattern languages and context sections; approach depends on author, often informal</td>
<td>Modeled explicitly in our proposal [26]: AD topic groups, AD levels, AD consists of AD alternatives</td>
<td>Pattern languages often informal, ADMs can add project-specific concretization and modeling rigor</td>
</tr>
<tr>
<td>Bottom-up composition of atomic units of design work into solution</td>
<td>Not a design goal</td>
<td>Not a design goal of retrospective AD capturing, supported in our proposal</td>
<td>Additional work for project team, integration effort required</td>
</tr>
<tr>
<td>Relationships between atomic units of design work</td>
<td>Informally via consequences, related patterns or pattern language diagrams (design spaces emerging, see [23])</td>
<td>Design goal, addressed by explicit dependency management via associations in UML meta model (directed graph)</td>
<td>ADMs more precise, can be populated from pattern texts</td>
</tr>
<tr>
<td>Requirements management link</td>
<td>Context, forces sections</td>
<td>Decision driver attributes in AD, decision identification step</td>
<td>Explicit in our proposal, informal in patterns</td>
</tr>
<tr>
<td>Links to software engineering and project management methods</td>
<td>Informally only, through other related patterns, or out of scope</td>
<td>Modeled in our proposal: scope, phase, role attributes</td>
<td>Integration effort required</td>
</tr>
<tr>
<td>Separate platform-independent from platform-specific concerns in models, but preserve links between concerns</td>
<td>Most patterns are platform-independent, platform-specific aspects come in informally via known uses and examples</td>
<td>Dedicated vendor asset level exists in our ADM proposal, decision dependencies can model link between levels</td>
<td>AD levels provide explicit support, patterns can be assigned to these levels</td>
</tr>
<tr>
<td>Ease of architectural documentation (authoring)</td>
<td>As patterns are publications, significant effort for pattern author (reviews, writer’s workshops), easy to reference</td>
<td>Depends on project setup, no formal authoring and review process (yet)</td>
<td>ADMs less rigorous, more project-specific, patterns more sustainable, long lasting (by definition)</td>
</tr>
<tr>
<td>Ease of consumption (usability)</td>
<td>Depends on pattern author, time intensive due to in-depth education character</td>
<td>Search capabilities, multiple ordering dimensions to support orientation character</td>
<td>ADMs provide orientation, patterns in-depth coverage of single design concern</td>
</tr>
</tbody>
</table>

- Original purpose differs, but concepts overlap
  - Education (patterns) vs. documentation (decisions)... but: pattern selection is a decision!
  - Both concepts have rather weak linkage with software engineering processes/methods
Agenda

1. Case study
   - Broker-based integration of core banking application components
2. Existing work
   - Architectural patterns and decision capturing
3. ArchPad walkthrough
   - Refinement stages
   - Reusable Architectural Decision Model for SOA (RADM for SOA)
4. Discussion
5. Conclusions and outlook
Patterns and Decisions Throughout the ArchPad Stages

Stage 1: “Forming”
- 1.1: Executive Decisions
  (Decided by External Stakeholders or To Dos for Overall Team Leads)
- 1.0: Requirements Analysis
  incl. Quality Attributes (a.k.a. Decision Drivers, Forces)
- 1.2: Business Patterns
  (e.g., Analysis Patterns, Industry Reference Models)

Stage 2: “Storming”
- 2.1: Conceptual Decisions
  (Lead Architects, Subsystem Architects)
- 2.2: Architectural Patterns
  (e.g., POSA, PoEAA, SOA)

Stage 3: “Norming”
- 3.1: Technology Decisions
  (Subsystem Architects, Development Leads)
- 3.2: Design Patterns
  (e.g., Gang of Four, Core J2EE, remoting, messaging)

Stage 4: “Performing”
- 4.1: Vendor Asset Decisions
  (Developers and Platform Specialists)
- 4.2: Implementation and Test Patterns,
  Known Uses of Patterns from Previous Stages
Decision (5.1) from Case Study in RADM for SOA

**Service Model**

**Scope**: Operation (Layer 3 + Layer 6)

**Decision drivers**: reliability needs, systems management capabilities, availability of provider

**Phase**: Macro Design

**Role**: Integration Architect

**AD**: Msg-01, Message Exchange Pattern (Conceptual/Technology AD, Stage 2/3)

- Do consumer and provider communicate synchronously or asynchronously?

**Background reading**: Hohpe/Woolf “Enterprise Integration Patterns”

**Phase**: Macro Design

**Role**: Integration Architect

**Recommendation**: Do not follow the MOM hype – decoupling in time is just one of several dimensions of loose coupling. The equation (NOT RM => NOT SOA) does not hold true.
Other Decisions in RADM for SOA (300+ Overall)

- **In and Out Parameter Granularity**
- **Transaction Management**
- **Session Management**
- **SOAP Engine Selection**
- **Transaction Attributes**

<table>
<thead>
<tr>
<th>Refinement Stage</th>
<th>Architectural Decision (AD)</th>
<th>Alternatives (optional)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stage 1 (RADM-A) Executive Decisions (EDs)</td>
<td>Platform Language Tool Preferences</td>
<td>Selection of reference architecture, layering approach, design, method, relevant literature (many alternatives)</td>
</tr>
<tr>
<td>Stage 2 (RADM-C) Conceptual Decisions (CDs), dealing with selection of architectural patterns</td>
<td>In and Out</td>
<td>BROKER [3] (RPC or messaging) vs. direct client-server connection</td>
</tr>
<tr>
<td>Stage 3 (RADM-T) Technology Decisions (TDs), dealing with selection of design patterns, technologies</td>
<td>Transaction Management</td>
<td>SYSTEM TRANSACTIONS vs. BUSINESS TRANSACTIONS [4]</td>
</tr>
<tr>
<td>Stage 3 (RADM-T) Technology Decisions (TDs), dealing with selection of design patterns, technologies</td>
<td>Messaging Exchange Style and Format</td>
<td>WS- and SOAP vs. REST and POX/ON vs. plain TCP/IP and custom strings vs. other</td>
</tr>
<tr>
<td>Stage 3 (RADM-T) Technology Decisions (TDs), dealing with selection of design patterns, technologies</td>
<td>Transport Protocol Binding</td>
<td>HTTP vs. messaging</td>
</tr>
<tr>
<td>Stage 3 (RADM-T) Technology Decisions (TDs), dealing with selection of design patterns, technologies</td>
<td>Service Provider Container Technology</td>
<td>Enterprise Java Bean (EJB) vs. plain Java object vs. other provider in other programming language, JAX-RPC vs. JAX-WS, CORBA vs. proprietary</td>
</tr>
<tr>
<td>Stage 3 (RADM-T) Technology Decisions (TDs), dealing with selection of design patterns, technologies</td>
<td>Business Process Language</td>
<td>Business Process Execution Language (BPEL) vs. proprietary</td>
</tr>
<tr>
<td>Stage 3 (RADM-T) Technology Decisions (TDs), dealing with selection of design patterns, technologies</td>
<td>Business Process Language</td>
<td>Rule-based dispatcher or not Deployment of interface of MICROFLOW ENGINES and MICROFLOW ENGINES, CONFIGURABLE ADAPTERS and REPOSITORIES, etc. (see [11])</td>
</tr>
<tr>
<td>Stage 3 (RADM-T) Technology Decisions (TDs), dealing with selection of design patterns, technologies</td>
<td>Process Layer Organization</td>
<td>BUSINESS OBJECT CONTROLLER vs. PROCESS CONTROLLER [1] vs. HUMAN TASK LIST [18]</td>
</tr>
<tr>
<td>Stage 3 (RADM-T) Technology Decisions (TDs), dealing with selection of design patterns, technologies</td>
<td>Resource Management Patterns</td>
<td>ACTOR, REQUESTER, or CLIENT-DEPENDENT INSTANCES (all from [21])</td>
</tr>
<tr>
<td>Stage 4 (RADM-A): Vendor Asset Decisions (VADs), dealing with product selection and configuration</td>
<td>Product</td>
<td>E.g., IBM WebSphere ESB, Progress Sonic ESB, Oracle</td>
</tr>
</tbody>
</table>
Agenda

1. Case study
   - Broker-based integration of core banking application components

2. Existing work
   - Architectural patterns and decision capturing

3. ArchPad walkthrough
   - Refinement stages
   - Reusable Architectural Decision Model for SOA (RADM for SOA)

4. Discussion
   - Tool support
   - Validation

5. Conclusions and outlook
Tool Support: Decision Modeling Application Wiki AD_kwik
Architecture of Collaboration Platform AD\textsubscript{kw}ik

<table>
<thead>
<tr>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Import/Export</td>
<td>Dependencies</td>
<td>Outcome Lifecycle</td>
<td>Collaboration</td>
</tr>
<tr>
<td>1</td>
<td>Content Repository</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Presentation: Web UI (Wiki)

Domain: OO-PHP

Persistence: Rel. DB

Validation via Case Studies and Action Research

- Applied ArchPad to own SOA projects retrospectively
  - Case study presented in this paper and BPEL-based telco order management
  - Revised and integrated book content:
    - “Perspectives on Web services”, (Zimmermann/Tomlinson/Peuser)
    - “Remoting Patterns” (Voelter/Kircher/Zdun)
- Method, content, and tool used on industry projects
  - Design method/process support: five industry engagements in several industries (telecommunications, government, retail and distribution)
  - Education, review, and governance workshops (professional services)
- Hosting company-internal instance of decision knowledge wiki
  - 100 registered ADkwik users
  - Scheduled for release via IBM alphaWorks emerging technology site
Agenda

1. Case study
   - Broker-based integration of core banking application components
2. Existing work
   - Architectural patterns and decision capturing
3. ArchPad walkthrough
   - Reusable decision model
   - Reusable Architectural Decision Model for SOA (RADM for SOA)
4. Discussion
   - Tool support
   - Validation
5. Conclusions and outlook
Summary of Key Concepts in ArchPad

- **Decision models** rather than text templates
  - Three levels of refinement from concepts to technology to assets
  - Population of decision space based on experience from earlier projects

- Decision model as domain-specific guide through the stages, *architecture and design patterns* describe alternatives in detail
  - Architectural decisions are conceptual alternatives
  - Design and technology patterns come in on subsequent stages

- Prescriptive **design method** for domain-specific architecture design
  - Comprehensive, but still comprehensible
  - Works for domains and styles with recurring decisions for which patterns have been mined (new decision alternatives also are a pattern source)
  - For example: enterprise applications and SOA
Advanced Usage Scenarios and Future Research

- **Project planning** and **health checking**
  - Work breakdown structures can be created from a decision model, as open decisions correspond to required activities.
  - If there are many, frequent changes, or many questions are still unresolved in late project phases, the project is likely to be troubled.

- **Decision models as input to software configuration planning**
  - Product selection and operational modeling decisions define which software licenses are required, and on which hardware nodes the required software has to be installed.

- **Enterprise architecture** instrument
  - Company-specific instance of the SOA RADM, consisting of a subset of decisions and alternatives to give freedom of choice to individual project teams without sacrificing overall architectural integrity.
Thank You!

- Questions?
- Comments?
- Architectural decision knowledge to share?
  - Consider authoring a practitioner report for OOPSLA 2008 (Deadline: March 19th)

  http://www.oopsla.org/oopsla2008/cfp_development.html

- “Perspectives on Web Services”, Springer-Verlag
  - Captures project experience 2001-2003, incl. 26 reusable architectural decisions
  - Foreword by Grady Booch
  - Website also has case study reports and other referenced papers

  http://soadecisions.org