The architect’s dilemma
Will reference architectures help?

Bernhard Humm, Martin Haft, Johannes Siedersleben
QoSA’05 Erfurt, September 21, 2005
Context

Standard and reference architectures

Quasar

Conclusion
Our business: tailored business information systems

Area of business

- Development and integration of tailored information systems for business critical processes
- IT consulting with engineering and implementation competence

<table>
<thead>
<tr>
<th>Customer</th>
<th>Project</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMW</td>
<td>Warranty System</td>
</tr>
<tr>
<td>COMMERZBANK</td>
<td>Payment Transactions</td>
</tr>
<tr>
<td>DAIMLERCHRYSLER</td>
<td>Global Ordering</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Customer</th>
<th>Project</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deutscher Wetterdienst</td>
<td>Meteorological Visualization System</td>
</tr>
<tr>
<td>HypoVereinsbank</td>
<td>Automatic Trading Platform</td>
</tr>
<tr>
<td></td>
<td>Integrated Order Management</td>
</tr>
</tbody>
</table>

sd&m AG, 11.11.2005, Seite 3
Our problem:
heterogeneous technology

- Business information systems have a similar structure
- Example: persistence layer between application core and data base (o/r mapping)
- Numerous products and standards with different, voluminous APIs – even within Java technology
- Enormous costs in selecting, learning, adopting and maturing technologies
It could be so simple
Persistence always works like this

<table>
<thead>
<tr>
<th>Operative interfaces</th>
<th>Administrative interfaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>interface Pool {</td>
<td>- All specifics are hidden behind administrative interfaces, e.g. database hints for performance optimization in particularly critical areas</td>
</tr>
<tr>
<td>UID insert(Object object);</td>
<td></td>
</tr>
<tr>
<td>void remove (Object object);</td>
<td></td>
</tr>
<tr>
<td>Object fetch(UID uid);</td>
<td></td>
</tr>
<tr>
<td>UID getUID(Object object);</td>
<td></td>
</tr>
<tr>
<td>Set&lt;Object&gt; executeQuery(</td>
<td></td>
</tr>
<tr>
<td>Query query,</td>
<td></td>
</tr>
<tr>
<td>List&lt;Object&gt; arguments</td>
<td></td>
</tr>
<tr>
<td>);</td>
<td></td>
</tr>
</tbody>
</table>

- Administrative interfaces are neither used nor seen by the application programmer
- Administrative interfaces are separated according to their user groups, e.g., systems administrator, database administrator, etc.
Agenda

Context

- **Standard and reference architectures**

Quasar

Conclusion
The vision: standard architectures

Definition: standard architecture for a specific domain e.g., business information systems

- Set of specified abstract components, their interactions and their interfaces (standard interfaces)
- Standard interfaces are specified by their syntax and semantics
- Concrete components, commercial or non-commercial products, either implement the standard interfaces directly or can be wrapped by the standard interfaces using adapters
- Application programmers program against the standard interfaces only and never see the concrete product APIs

Standard architectures could considerably reduce development cost and time and increase the quality of the resulting systems
Theory and practice: standard architectures have hardly ever worked

### Attempts for standard architectures
- CORBA: not widely used as an architecture
- J2EE, .NET: widely used platforms but no architectures
- Architectural blueprints (e.g., three-layer-architecture): too general
- Quasar: For 7 years, we have failed to establish standard interfaces, not even within sd&m

### Reasons for not adopting standard architectures
- Acceptance: why should I learn yet another API?
- Special features: how to hide all special features behind administrative interfaces?
- Convenience: why should I do without convenience methods?
- Costs for implementing adapters: who pays for the adapters?

Standard architectures are either too specific or too general to be widely applicable: the architects‘ dilemma
Our approach: reference architectures
Less strict than standard architectures

### Definition reference architecture
- Set of abstract components, interactions and interfaces (reference interfaces)
- Specified by syntax and semantics
- Products do not have to be wrapped
- Application programmers may see the concrete product APIs

### Differences between APIs
Reference interfaces and product APIs may differ in:
- Naming: different method names
- Specialisation: e.g., more concrete parameters
- Special Features: API extension
- Convenience: additional methods

### Characteristics
- Minimal: focus on the essence
- Complete: all necessary functionality
- Disjoint: functionality for different user groups in separate interfaces
Reference architectures are still useful
Example: portal factory for pharmaceutical enterprise
Context

Standard and reference architectures

- **Quasar**

Conclusion
Quasar: Quality Software Architecture

- Principles
  - Simplicity
  - Separation of concerns; software categories
  - Components and interfaces
  - Errors and exceptions

- Architecture
  - Application architecture, technical architecture, technical infrastructure
  - Reference architectures, specified interfaces for abstract technical components
  - Guideline for parallel development

- Components
  - QuasarPersistence
  - QuasarViews
  - QuasarAuthorization
  - QuasarBusiness-Components
  - Other products like, e.g., Hibernate

- Expertise within sd&m: schools, communities for architecture and quasar components
The Quasar reference architecture for business information systems

- Separation of business application (A components) and technology (abstract T components)
- Identification of the main abstract T components
- Identification of dependencies
A zoom into Quasar: Client architecture

- Separation of call and callback interfaces
- Separation of administrative (complex, used rarely) and operative (simple, used often) interfaces
- Refinement of dependencies via uses and implements relationships
Interface specification:
Syntax and semantics are specified by QSL

interface DialogManager [export, operation]
// Central interface for managing dialogs.
uses
    Dialog,
    DialogType
    DialogState

command
Dialog makeDialog(in DialogType dialogType)
// An instance of a Dialog is provided.
// Parameters:
//   dialogType - application-specific
// Returns:
//   initialized dialog in state closed
post result.getState() == DialogState.CLOSED

command
void releaseDialog(inout Dialog dialog)
// An instance of a Dialog is released.
// Parameters:
//   dialog - dialog instance to be released
pre dialog.getState() == DialogState.CLOSED

- Reference interfaces are specified via QSL (Quasar specification language)
- Specification of basicQueries, derivedQueries and commands
- Specification of pre and post conditions
- Specification of invariants
- Specification of errors
- Complex functionality specified in prose
Interactions between components: illustrated via UML sequence diagrams

sd perform application use case

User

GUI Library

- «interface» VisualRepresentation
- ActionListener

Presentation

- «interface» Presentation
- EventManager

DialogKernel

- «interface» DataManager

ApplicationKernelFacade

- «interface» A_UseCase_1

User press button

fireActionEvent()

trigger(event)

some A use case method

UID := Pool.insert(value)

update()

Object := fetch(key)

update()
Agenda

Context

Standard and reference architectures

Quasar

■ Conclusion
Conclusions

- Standardisation is a major step towards the industrialisation of the software industry

- Standard architectures have hardly ever been established: either they are too specific or too general to be widely applicable – the architect’s dilemma.

- Reference architectures reduce the binding character of standard architectures. However, they still have proven benefits in industrial practice

- Quasar is a fully specified reference architecture for business information systems – one particularly important part is the client architecture

- Quasar is established at sd&m as architectural blue print for all custom software projects. This has considerably reduced development cost and time and increased the quality of resulting systems

- Reference architectures do not solve the architect’s dilemma. Still, they are a most valuable compromise and may act as one step towards standard architectures proper