Derivation of Domain Specific Design Patterns
Applications to design and construction of Intranets and SGS
Dan Port, USC Center for Software Engineering
CSE Annual Research Review 1998

Intelligence is the ability to perceive patterns. Genius is the ability to perceive patterns where the bulk of mankind cannot. Scholarship is the ability to perceive patterns where there aren't any.
— Michael J. Moran

The Problem

- Full SGS frameworks
  - too general
    - specialization needed
    - specialization of entire framework impractical
  - costly to adopt in whole
  - unclear how to adopt parts
  - Ex. Nescape SuiteSpot, Lotus Domino, SSCS, Storm IMT-ISIS

- Object repositories/libraries
  - too many parts
    - hard to find the "right" part
  - parts too specialized or so general that heavy modification is required
Leveraging existing technology: frameworks
(RAD, Re-use, legacy integration, COTS)

- Full Intranet and SGS frameworks
  - too general
    - specialization needed
    - specialization of entire framework impractical
  - costly to adopt in whole
  - unclear how to adopt parts

Examples:
- Netscape Suitespot
- Lotus Domino
- ActiveWeb
- SCS
- STORM-ICIS

Leveraging existing technology: repositories
(RAD, Re-use, legacy integration, COTS)

- Object repositories/libraries
  - too many parts
    - hard to find the "right" part
  - parts too specialized or so general that heavy modification is required

Examples (maybe...):
- Netscape JFC
- Microsoft Foundation Classes
- STORM-IMT
What is a Design Pattern?

**Design Patterns:** A description of a solution to a common problem in terms of components and objects. (see [Gamma et al. Sec. 1.1])

Examples:
- Gof:
  - Command, Reactor, Facade, Strategy
- Common OOD:
  - Delegation, CB-Subtyping, CB-States, Metadata
- Architectural:
  - MVC, MVCA, Representational Subsystems

Note that some of these patterns are context-sensitive and others are context-free.

For a variety of reasons, developers rarely make direct application of context-sensitive design patterns and there aren’t many context-free patterns.

Problems With General Design Patterns

**Main reasons:**
- No context given for which a pattern’s qualities (such as interface attributes, I/O parameters, etc.) can be coherently resolved.
- Scope of use is difficult to ascertain.
- Boundaries not always clear within system (does not always naturally decompose into patterns) - makes identification difficult and application artificial
- Generally too abstract

This often leads to attempts to “force fit” patterns into use, or worse, changing the requirements to accommodate them, resulting in wasted time and inelegant designs which lead to quality attribute consequences (e.g. “ilities” such as maintainability, scalability, extensibility, ...)
What is a DSDP?

Domain Specific Design Pattern: are collections of components and objects that represent, in software, well-defined (encapsulated, clear boundary, highly cohesive) partitions of a particular system domain.

Important principle: a system should be designed in a context larger than itself, but not so large that qualities cannot be easily resolved.

Goal: To clearly describe a flexible architecture in terms of well-defined, highly encapsulated, manageable parts that are in alignment with the natural constraints of the domain.

Why Use DSDPs?

Main idea: discover the commonalties across the system domain, use them as natural partitions, create well-defined representations for them, then use them to construct systems within that domain.

- Assumption is that domain partitions are an inherent part of the domain and provide natural constraints for any application within that domain.
- Partitions are small, well-defined, and manageable.
- Qualities (such as relationships) are resolvable and meaningful within the domain context.
- Partitions are easy to identify from sample applications.
- Patterns can now be identified and fashioned to partitions.
Domain Factoring: Decomposition followed by Classification

General forms of DSDPs

<table>
<thead>
<tr>
<th>Name</th>
<th>Uses a graphical user interface</th>
<th>How deployed</th>
<th>Independent or dependent on other components to define its operations</th>
<th>Active in generating events and actions or only responds to events and actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applications</td>
<td>GUI</td>
<td>Client-side</td>
<td>independent</td>
<td>active</td>
</tr>
<tr>
<td>Mechanisms</td>
<td>no GUI</td>
<td>Server-side</td>
<td>dependent</td>
<td>passive</td>
</tr>
<tr>
<td>Glue</td>
<td>no GUI</td>
<td>Server-side</td>
<td>dependent</td>
<td>passive</td>
</tr>
<tr>
<td>Components</td>
<td>N/A</td>
<td>N/A</td>
<td>dependent</td>
<td>N/A</td>
</tr>
</tbody>
</table>
### General forms of DSDPs

<table>
<thead>
<tr>
<th>Name</th>
<th>Uses a Graphical User Interface</th>
<th>How Deployed: Client or Server</th>
<th>Independent or Dependent on Other Components to Define Operations</th>
<th>Active in Generating Events or Actions: Active or Passive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applications</td>
<td>GUI</td>
<td>client-side</td>
<td>independent</td>
<td>active</td>
</tr>
<tr>
<td>Mechanisms</td>
<td>no GUI</td>
<td>server-side</td>
<td>dependent</td>
<td>active</td>
</tr>
<tr>
<td>Glue</td>
<td>no GUI</td>
<td>server-side</td>
<td>dependent</td>
<td>passive</td>
</tr>
<tr>
<td>Components</td>
<td>N/A</td>
<td>N/A</td>
<td>dependent</td>
<td>N/A</td>
</tr>
</tbody>
</table>

### Intranet Sample Application Domain

<table>
<thead>
<tr>
<th>1-way Communication (Messages)</th>
<th>2-way Communication (Messages)</th>
<th>General Information (Resources)</th>
<th>Schedulers</th>
<th>Requests</th>
<th>Managers</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>memo</td>
<td>conversation forms</td>
<td>FAQ</td>
<td>work scheduler</td>
<td>service requests</td>
<td>meeting</td>
<td>login app</td>
</tr>
<tr>
<td>phone messages</td>
<td>teleconference</td>
<td>Occupancy</td>
<td>non-resource</td>
<td>time off/vacation</td>
<td>request</td>
<td>none</td>
</tr>
<tr>
<td>Fan</td>
<td>video conferencing</td>
<td>meeting</td>
<td>meeting scheduler</td>
<td>OLC</td>
<td>RCS</td>
<td></td>
</tr>
<tr>
<td>Yell</td>
<td>VR conferencing</td>
<td>EVC directory/who’s in line</td>
<td>to-do/priorities</td>
<td>archive</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>inventory</th>
<th>mail</th>
<th>setting</th>
</tr>
</thead>
</table>

6
DSDPs from factored partitions

<table>
<thead>
<tr>
<th>mechanisms</th>
<th>glue</th>
<th>components</th>
</tr>
</thead>
<tbody>
<tr>
<td>repository</td>
<td>relator</td>
<td>item</td>
</tr>
<tr>
<td>notification center</td>
<td>translator</td>
<td></td>
</tr>
<tr>
<td>journalizer</td>
<td>general form generation</td>
<td></td>
</tr>
<tr>
<td>authentication</td>
<td></td>
<td></td>
</tr>
<tr>
<td>login</td>
<td></td>
<td></td>
</tr>
<tr>
<td>encryption</td>
<td></td>
<td></td>
</tr>
<tr>
<td>session</td>
<td></td>
<td></td>
</tr>
<tr>
<td>routing</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example Application: Service Request
Domain Unique Example: Intranet Login

Hardware Interfaces
- RF Modulation/Demodulation
- Ground-HW Control

Middleware
- Command
- Telemetry
- Alarm Processing

High-level
- Satellite Control
- Payload Control
- Ephemeris

SGS Application Domain
### Some SGS DSDP's

<table>
<thead>
<tr>
<th>Mechanisms</th>
<th>Glue</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ephemeris Prediction</td>
<td>Data</td>
<td>Command</td>
</tr>
<tr>
<td>Alarm Notification</td>
<td>Request</td>
<td>Messages</td>
</tr>
<tr>
<td></td>
<td>Composition</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Verification Method</td>
<td></td>
</tr>
</tbody>
</table>

### Value of DSDPs

**Customer**
- Identification of standard technology
- Evolution of systems based on unforeseen requirements
- Evolution of COTS products

**Developer**
- Manageable production line
- Optimize re-use
- Evolution/migration of components

**Vendor**
- Production focus
- Larger (more general) markets
- Not tied to particular customer
- Product line evolution