An Introduction to Software Architecture Case Studies

David Garlan & Mary Shaw – 94
Key Word In Context (KWIC)

- Search index
  - searching for keywords with context sensitive display
  - provides the user with more information

Harry Potter

Harry Potter - The Official Site

Harry Potter and the Half-Blood Prince

Harry Potter - Wikipedia, the free encyclopedia

Harry Potter (film series) - Wikipedia, the free encyclopedia

Show more results from en.wikipedia.org
## Example

**Input: Titles**
- Clouds are white
- Ottawa is beautiful

**Output: Index**
- are white Clouds
- beautiful Ottawa is
- Clouds are white
- is beautiful Ottawa
- Ottawa is beautiful
- white Clouds are
Solution 1

Main Program/Subroutine with Shared Data

- Functional decomposition
- Components are subroutines
Solution 1

Strengths

• Centralized data
  – efficient representation of data
• Modular decomposition

Weaknesses

• Resistant to change
  – consider the impact of data storage format
  – difficult to enhance the overall functionality
  – reuse of component is difficult
Solution 2

Abstract Data Types

- Similar to one with data encapsulation
  - data access via component interface invocation
  - no direct data access

- Components similar to solution 1
Abstract Data Types

Solution 2
Solution 2

Advantages

- Handles change well
  - algorithm and data are encapsulated in individual modules
- Reuse
  - modules interact via defined interfaces

Disadvantages

- Evolution still a problem
  - to add new features may require changes to existing or addition of new components
Solution 3

Implicit Invocation

- Similar to solution 1
  - shared data
- Two main differences
  - data is more abstract
    - underlying storage is not exposed to components
  - components are invoked implicitly
    - e.g. when a line is added
Calls to circular shift and alphabetizer are implicit, and are the result of inserting lines.
Solution 3

Advantages

• Strong evolution path
  – functional enhancements are easy
  – new components can be attached and removed
  – components are shielded from data storage representation
  • REALLY WHY?

• Minimal component coupling/dependency
  – data events are the source of all interactions
Solution 3

Disadvantages

• Difficult to control the ordering of processing
• Requires more storage capacity
  – IS THIS REALLY A DISADVANTAGE?
Pipes & Filters

- Four filters
  - input, shift, alphabetize, output
  - each filter can compute when data is available at the input
  - data sharing is restricted by pipes
Solution 4

Advantages

- Intuitive flow of processing
- Reuse
- Evolution
  - new filters can be easily added
Solution 4

Disadvantage

- Virtually impossible to support an interactive system
- **Is this a true pipes & filters?**
  - consider the data flow
- What is the LCD data unit?
## Comparison

<table>
<thead>
<tr>
<th></th>
<th>Shared Memory</th>
<th>ADT</th>
<th>Implicit Invocation</th>
<th>Pipe &amp; Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>change in algorithm</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>change in data representation</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>change in functionality</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>performance</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>reuse</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>
Reading

Will be on exam

- Case Study 2: Instrumentation Software
- Case Study 3: A Fresh View of Compilers

Will not be on exam

- Case Study 4: A Layered Design with Different Styles for the Layers
- Case Study 5: An Interpreter Using Different Idioms for the Components