Software Architecture Patterns

Mark Richards
Hands-on Software Architect
Author of Enterprise Messaging Video Series (O'Reilly)
Author of Java Message Service 2nd Edition (O'Reilly)
Co-author of Software Architecture Fundamentals Video Series (O'Reilly)
agenda

introduction
layered architecture pattern
event-driven architecture pattern
microkernel architecture pattern
space-based architecture pattern
Software Architecture
Pattern Analysis
**Component**

an encapsulated unit of software consisting of one or more modules that has a specific role and responsibility in the system
how are components classified?
how do components interact?
does the architecture scale?
how responsive is the architecture?
is there a logical flow to the components?
what are the deployment characteristics?
how does the architecture respond to change?
is the architecture extensible and if so how?
how maintainable is the architecture?
architecture patterns help define the basic characteristics and behavior of the application
layered architecture

- presentation layer
  - component
  - component
  - component

- business layer
  - component
  - component
  - component

- persistence layer
  - component
  - component
  - component

- database layer
  - component
  - component
  - component
  - component
layered architecture

request

database layer
persistence layer
business layer
presentation layer

component component component
component component component
component component component
component component component

CLOSED
CLOSED
CLOSED
CLOSED
layered architecture

- Presentation layer
- Business layer
- Persistence layer
- Database layer

separation of concerns
layered architecture

presentation layer
business layer
persistence layer
database layer

layers of isolation
layered architecture
hybrids and variants

presentation layer
business layer
services layer
database layer
layered architecture
hybrids and variants

presentation layer
business layer
services layer
persistence layer
database layer

component component component
component component component
component component component
component component component

OPEN
CLOSED
CLOSED
CLOSED
CLOSED
layered architecture
hybrids and variants
layered architecture
hybrids and variants
layered architecture considerations

good general purpose architecture and a good starting point for most systems

watch out for the architecture sinkhole anti-pattern

tends to lend itself towards monolithic applications
layered architecture

overall agility
deployment
testability
performance
scalability
development
complexity
loose coupling

analysis
event-driven architecture

mediator topology

broker topology
event-driven architecture
mediator topology
event-driven architecture

you move...

process engine

change address

reclc quote

update claims

adjust claims

notify insured

customer process

quote process

claims process

adjustment process

notification process
event-driven architecture
broker topology

event

event processor
module
module
module
module

event channel

event processor
module
module
module
module

event channel
event-driven architecture

broker topology
event-driven architecture

you move...

you moved!

customer process

change address

change address

quote process

recalc quote

quote process

recalc quote

notification process

claims process

update claims

update claims

adjustment process
event-driven architecture

considerations

contract creation, maintenance, and versioning can be difficult

must address remote process availability or unresponsiveness

reconnection logic on server restart or failure must be addressed
event-driven architecture

analysis

overall agility
deployment
testability
performance
scalability
development
complexity
loose coupling

(event -> event queue) -> (event processor) -> (event topic) -> (process)
microkernel architecture
(a.k.a. plug-in architecture pattern)
microkernel architecture

architectural components

- **core system**: minimal functionality to run system, general business rules and logic, no custom processing
- **plug-in module**: standalone independent module, specific additional rules or logic
microkernel architecture
microkernel architecture

source validation tool

- check header standards
- check interceptors
- check contract standards
- check SQL calls
- check audit writes
- check other stuff...
microkernel architecture

claims processing
microkernel architecture

registry

core system

registry
1: <location>, <contract>
2: <location>, <contract>
3: <location>, <contract>
4: <location>, <contract>

plug-in component 1
plug-in component 2
plug-in component 3
plug-in component 4
microkernel architecture
registry

```java
static {
    pluginRegistry.put(NAMING, "ValidatorNamingPlugin");
    pluginRegistry.put(SYSOUT, "ValidatorSysoutPlugin");
    pluginRegistry.put(AUDIT, "ValidatorAuditPlugin");
    pluginRegistry.put(TODO, "ValidatorTodoPlugin");
    pluginRegistry.put(COMMENTS, "ValidatorCommentsPlugin");
    pluginRegistry.put(SVC_CALLS, null);
}
```
private String executeChecks(String moduleName) throws Exception {
    for (Map.Entry<String, String> entry : pluginRegistry.entrySet()) {
        if (entry.getValue() != null) {
            Class<?> c = Class.forName(PLUGIN_PKG + entry.getValue());
            Constructor<?> con = c.getConstructor();
            ValidatorPlugin plugin = (ValidatorPlugin)con.newInstance();
            data = plugin.execute(data);
        }
    }
}
microkernel architecture

plug-in contracts

core system

plug-in component 1
plug-in component 2
plug-in component 3
plug-in component 4
public class ValidatorData {
    public String moduleName;  //input
    public List<String> moduleContents;  //input
    public String validationResults;  //output
}

public interface ValidatorPlugin {
    public ValidatorData execute(ValidatorData data);
}
microkernel architecture

considerations

can be embedded or used as part of another pattern

great support for evolutionary design and incremental development

great pattern for product-based applications
microkernel architecture analysis

overall agility
deployment
testability
performance
scalability
development
complexity
loose coupling
space-based architecture

let's talk about scalability for a moment...
space-based architecture
space-based architecture

architectural components

processing unit

virtualized middleware

messaging grid

data grid

processing grid

deployment manager
space-based architecture
processing unit
space-based architecture

middleware

- messaging grid
- data grid
- processing grid
- deployment manager
space-based architecture

middleware

manages input request and session
space-based architecture

middleware

manages data replication between processing units

- messaging grid
- data grid
- processing grid
- deployment manager

[Diagram showing a network of processing units connected to a virtualized middleware, which manages data replication between the processing units.]
space-based architecture

middleware

manages distributed request processing
space-based architecture

middleware

manages dynamic processing unit deployment

- messaging grid
- data grid
- processing grid
- deployment manager
space-based architecture

product implementations

javaspaces
gigaspaces
ibm object grid
gemfire
ncache
oracle coherence
it's all about variable scalability...
good for applications that have variable load or inconsistent peak times
not a good fit for traditional large-scale relational database systems
relatively complex and expensive pattern to implement
space-based architecture

analysis

overall agility
deployment
testability
performance
scalability
devlopment
complexity
Software Architecture Patterns

Mark Richards
Independent Consultant
Hands-on Software Architect
Published Author / Conference Speaker
http://www.wmrichards.com
http://www.linkedin.com/pub/mark-richards/0/121/5b9