# Cloud Computing: SOA Design Patterns, Part I

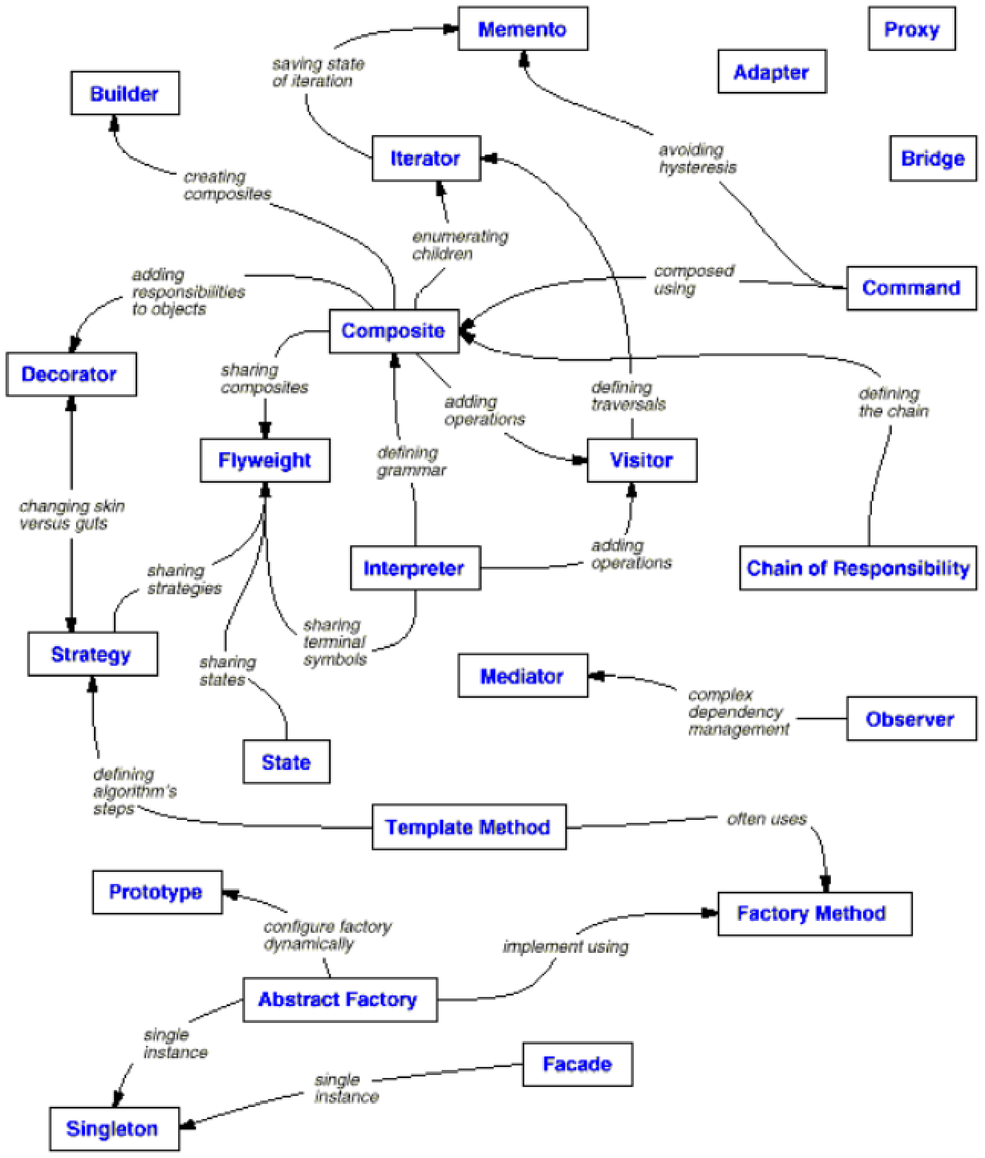
Vijay Dialani, PhD
Boise State University
vijaydialani@boisestate.edu

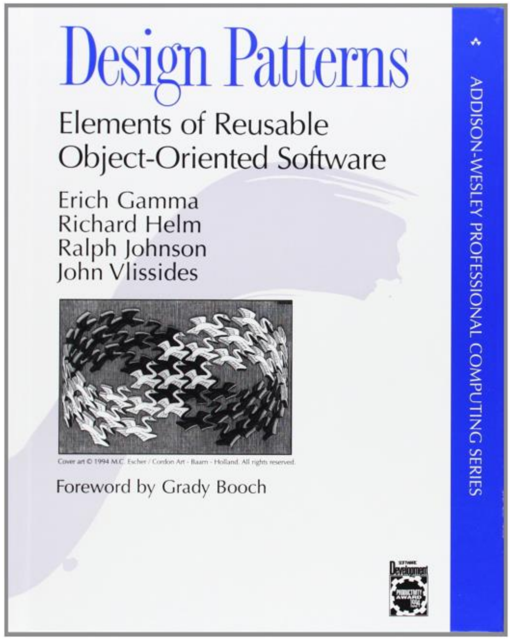
## What are Design Patterns?

**Software Design Patterns - Wikipedia**

In software engineering, a **design pattern** is a general reusable solution to a commonly occurring problem within a given context in software design. A design pattern is not a finished design that can be transformed directly into source or machine code. It is a description or template for how to solve a problem that can be used in many different situations.

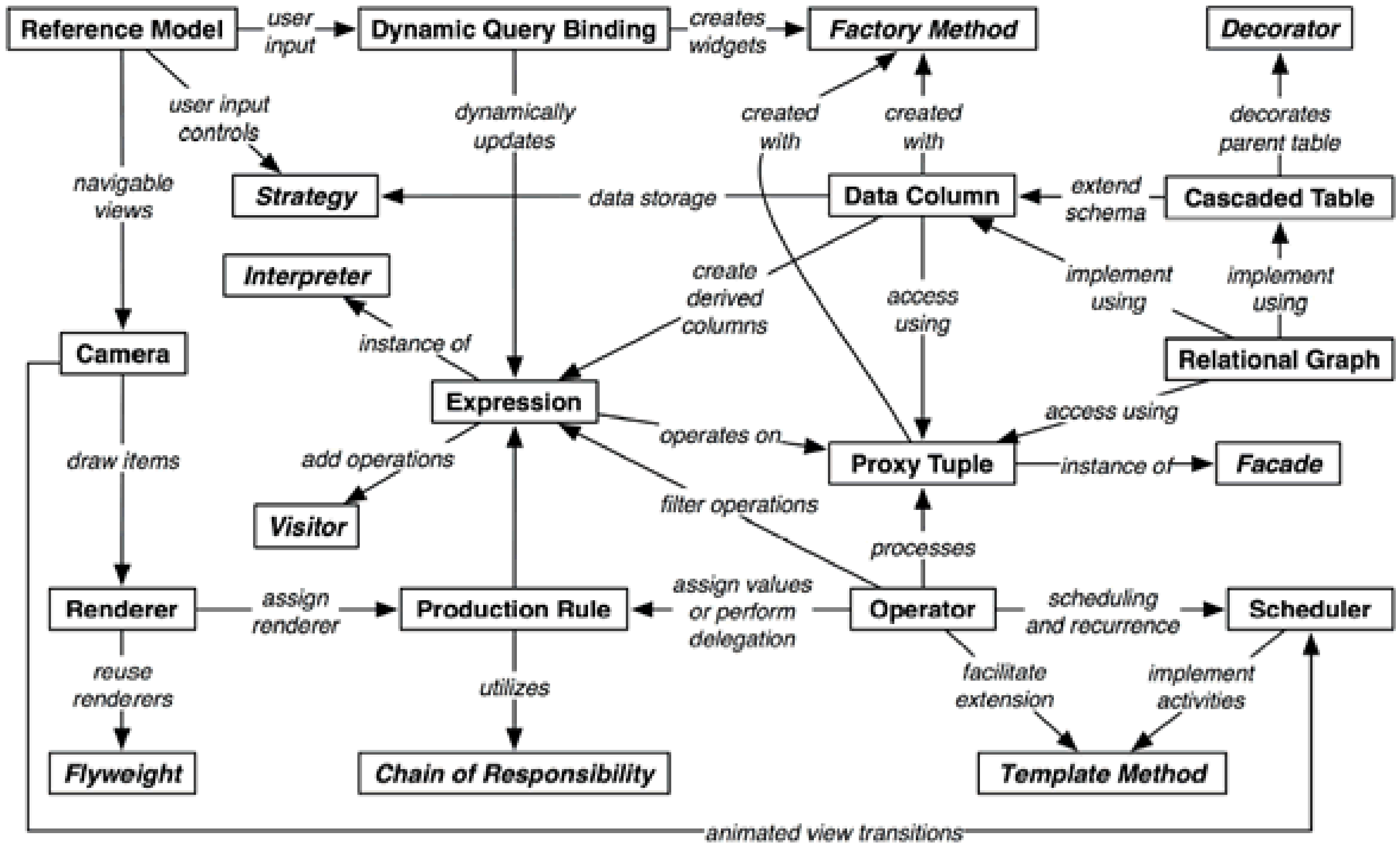
# Software Design Patterns





Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

# Softawre Design Patterns for Information Visualization



@article{2006-infovis-design-patterns, title = {Software Design Patterns for Information Visualization}, author = {Jeffrey Heer AND Maneesh Agrawala}, journal = {IEEE Trans. Visualization \& Comp. Graphics (Proc. InfoVis)}, year = {2006}, volume = {12}, issue = {5}, pages = {853--860}, url = {http://vis.stanford.edu/papers/infovis-design-patterns} }

# Software Architecture

-  ***Architecture occurs early***. It should represent the set of earliest design decisions that are both hardest to change and most critical to get right.

-  ***Architecture is an attribute of every system***. Whether or not its design was intentional, every system has an architecture.

-  ***Architecture breaks a system into components and sets boundaries***. It doesn't need to describe all the components, but the architecture usually deals with the major components of the solution and their interfaces.

-  ***Architecture is about relationships and component interactions***. We're interested in the behaviors of the individual components as they can be discerned from

# SOA Design Patterns – Why needed?

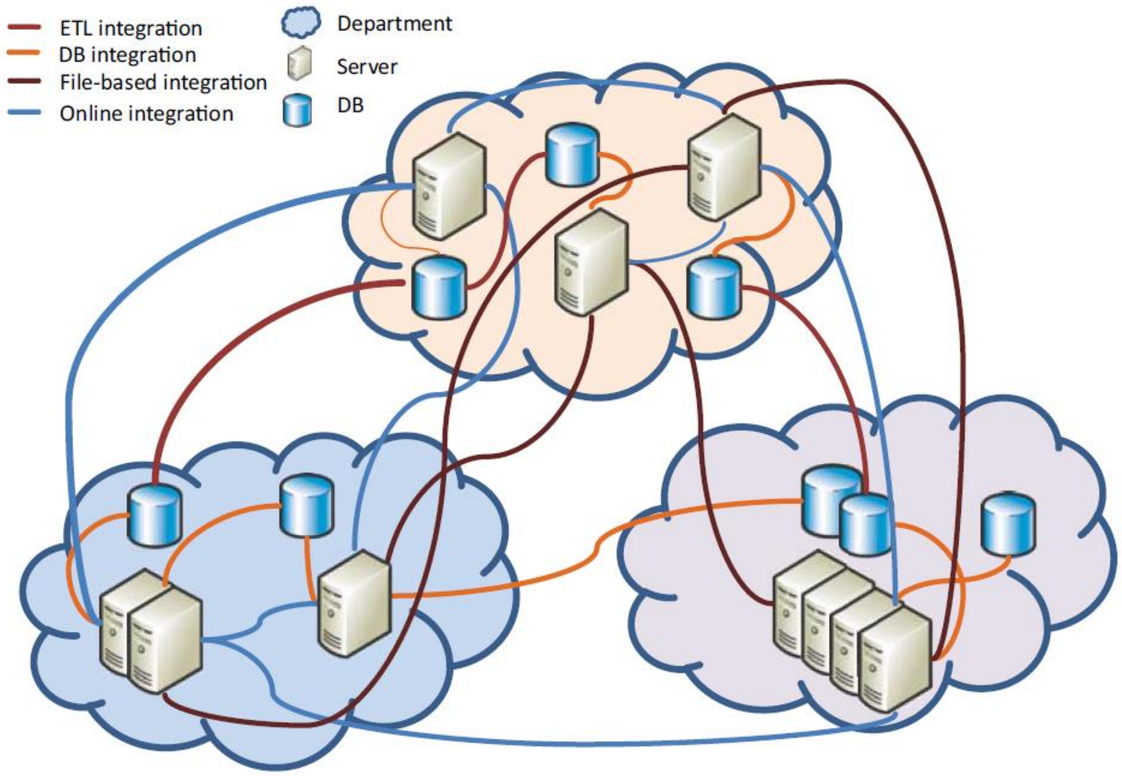


**Figure 1.2** **Typical integration spaghetti in enterprise systems. Each department builds its own systems, and as people use the systems, they find they need information from other systems. Point-to-point integration emerges.**

# SOA Design Patterns – Why needed?

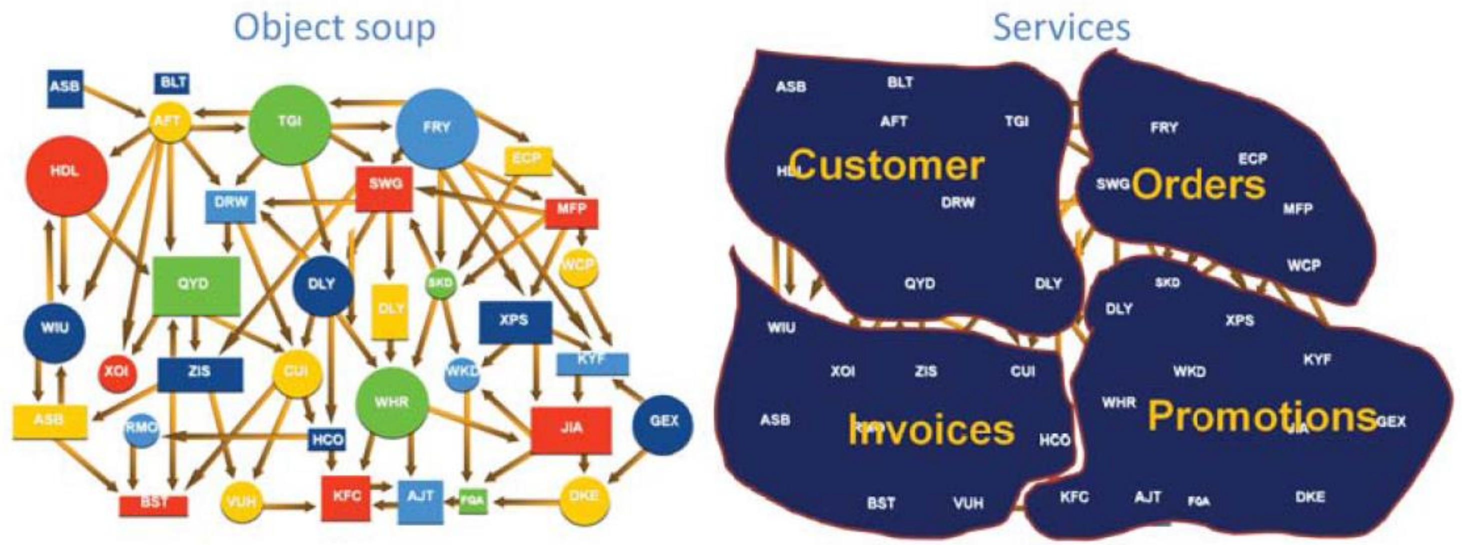
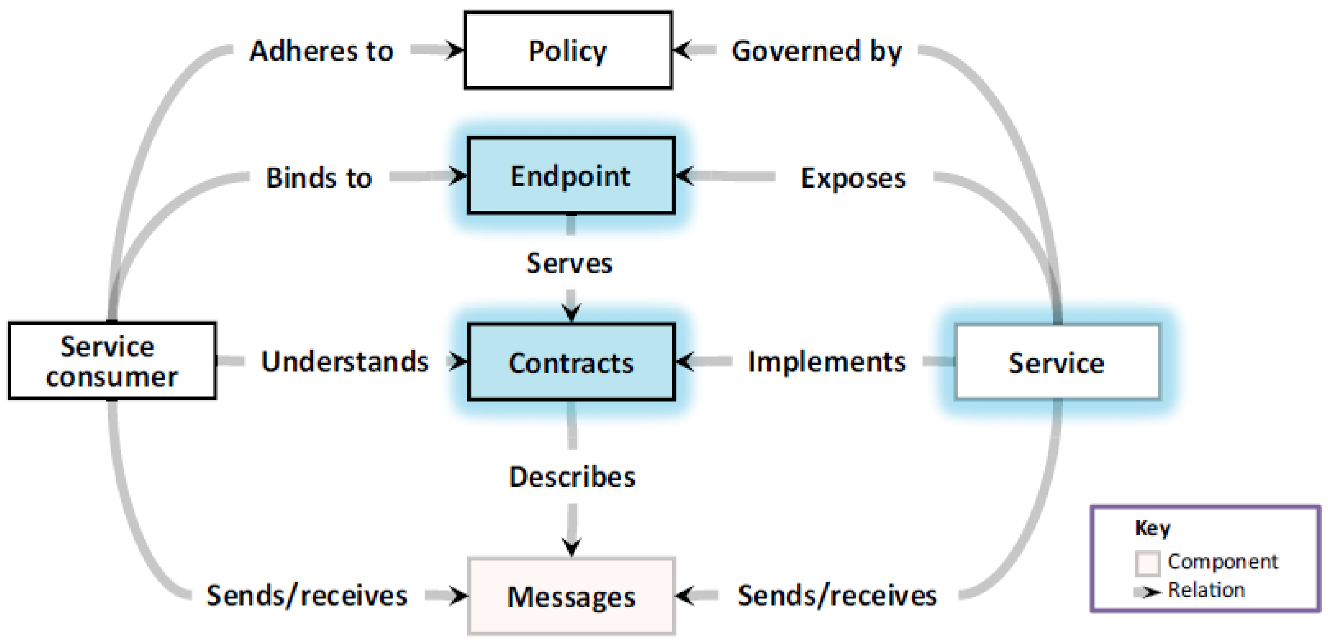


**Figure 1.3** **From object soup to well-formed services; one of the ideas behind SOA is to set explicit boundaries between larger chunks of logic, where each chunk represents a high-cohesion business area. This is an improvement on the more traditional approach, which more often than not results in an unintelligible object soup.**

# SOA Design Patterns – Service Host

*Service Host pattern*



# SOA Design Patterns – Service Host

**How can you easily configure services and avoid duplicating mundane tasks, such as setting listeners and wiring components, for each service?**

**SOLUTION**
Create a common service host component or framework that acts as a container for services. This container should be configurable and will perform the wiring and setup of services.
The Service Host, illustrated in figure 2.2, is a framework or a complete component that performs some or all of the following functions:

- *Lifecycle*—Takes care of instantiating services, recycling services on fault, inplace upgrades, and so on
- *Configuration*—Reads and applies configuration to hosted services, including configuration for security, contract policies, and ports
- *Wiring*—Performs runtime setup of component wiring such as binding a listener on a service's endpoint
- *Administration*—Lets an administrator control the lifecycle of a hosted service and may also include monitoring capabilities (this is an additional layer on top of lifecycle responsibility)
- *Environment*—Provides auxiliary services like logging, cache, database libraries (ODBC/JDBC), scheduler, and so on

# SOA Design Patterns – Service Host

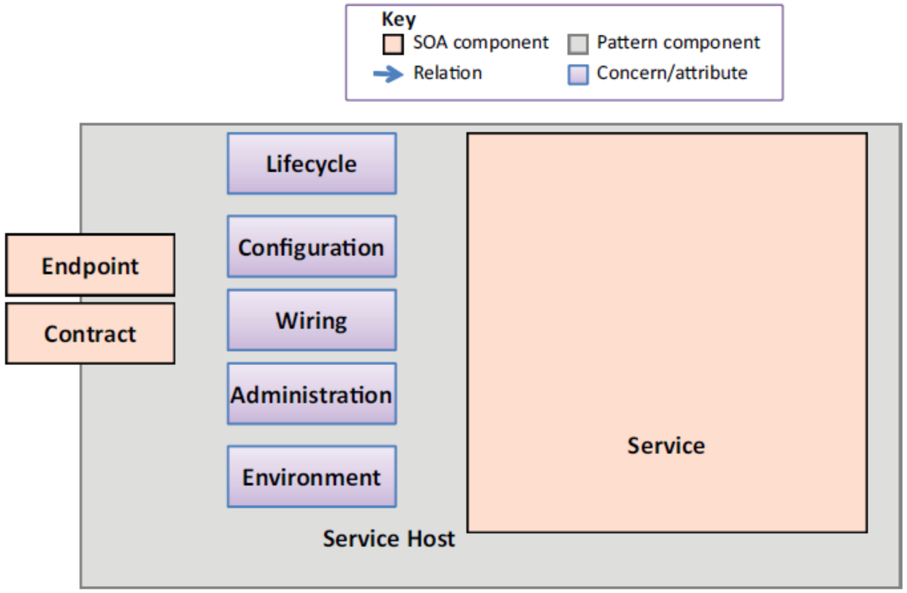


**Figure 2.2 Service Host is a container for a service, and it performs the wiring and configuration on the service's behalf.**

# SOA Design Patterns – Active Service

**Problem**:
How can you increase service autonomy and handle temporal concerns?

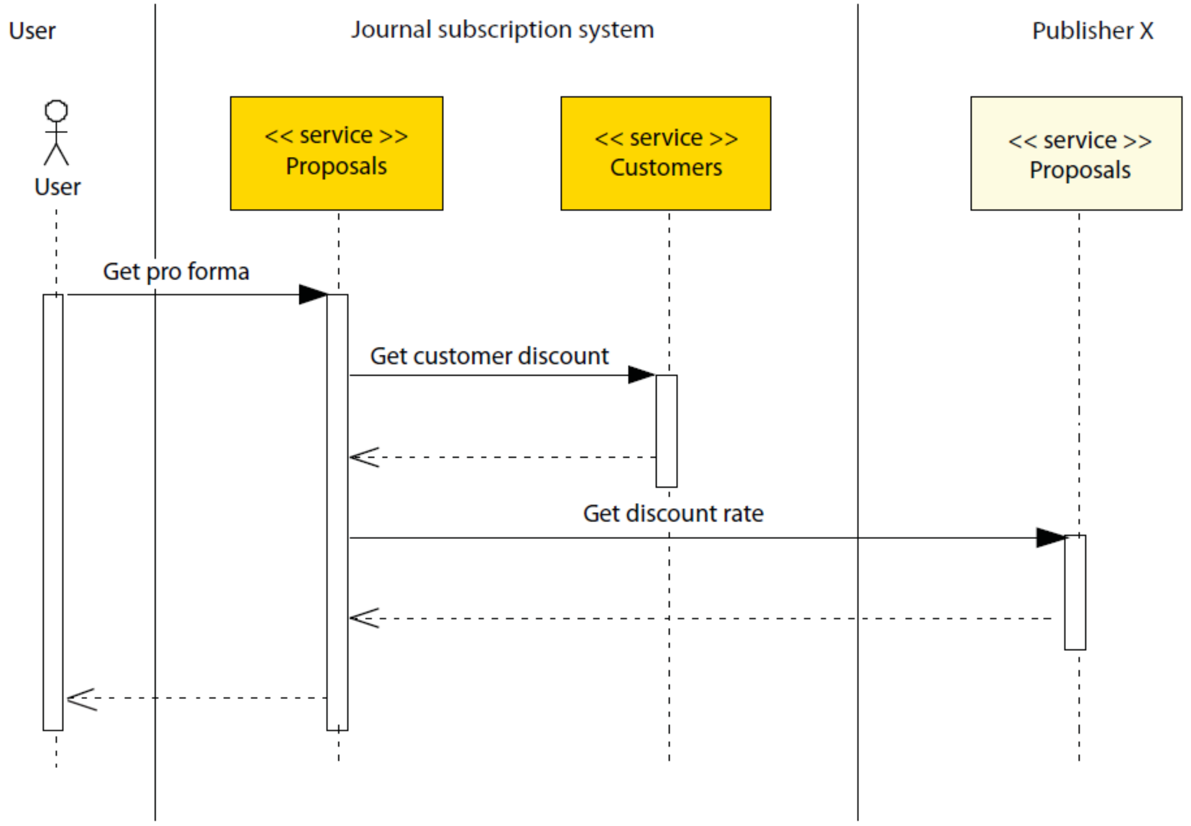


**Figure 2.5** **The Proposals service needs to get data from both internal and external services.**

# SOA Design Patterns – Active Service

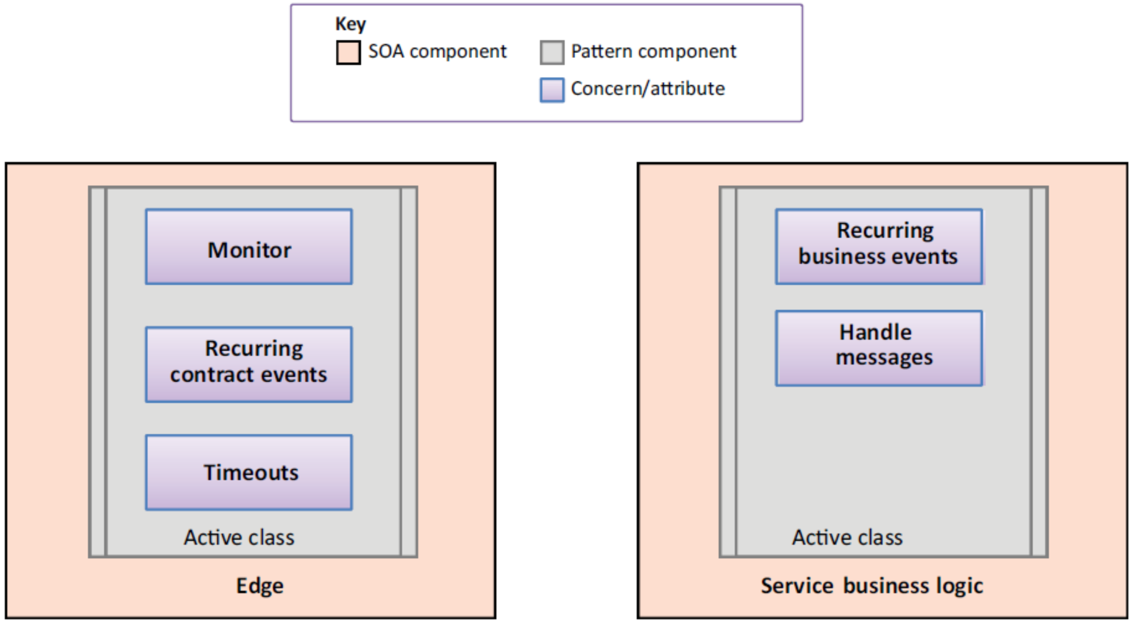


**Figure 2.6** **With the Active Service pattern, you add independent behavior to a service in its own thread of control. This pattern can be used to handle recurring events, such as timeouts and monitoring.**

# SOA Design Patterns – Active Service

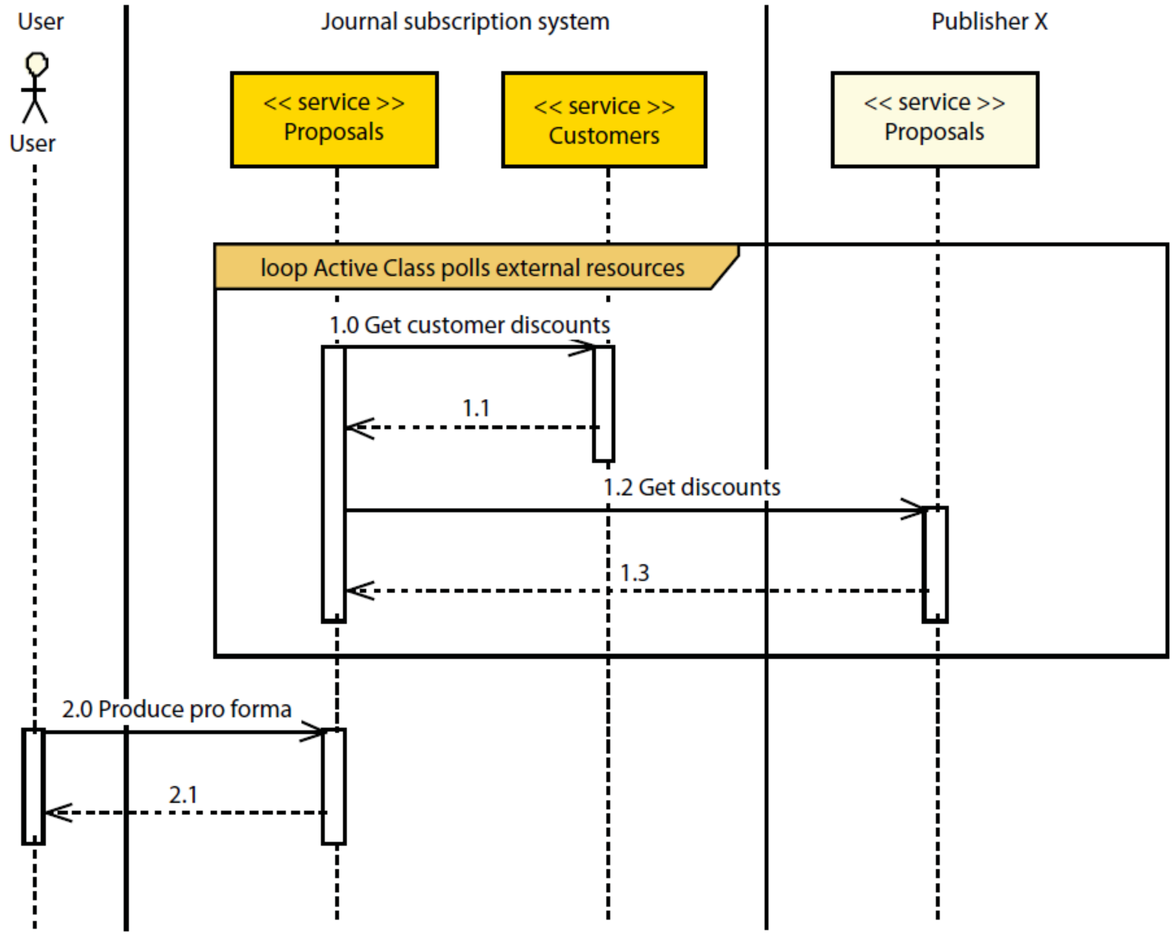


**Figure 2.7 The Proposals service actively polls the other services for the information it needs. The proposal service can then respond to pro-forma requests (2.0 in the diagram) immediately, and without dependency on any other services' availability.**

# SOA Design Patterns – Transactional Service Pattern

**Problem:** How can a service handle requests reliably?

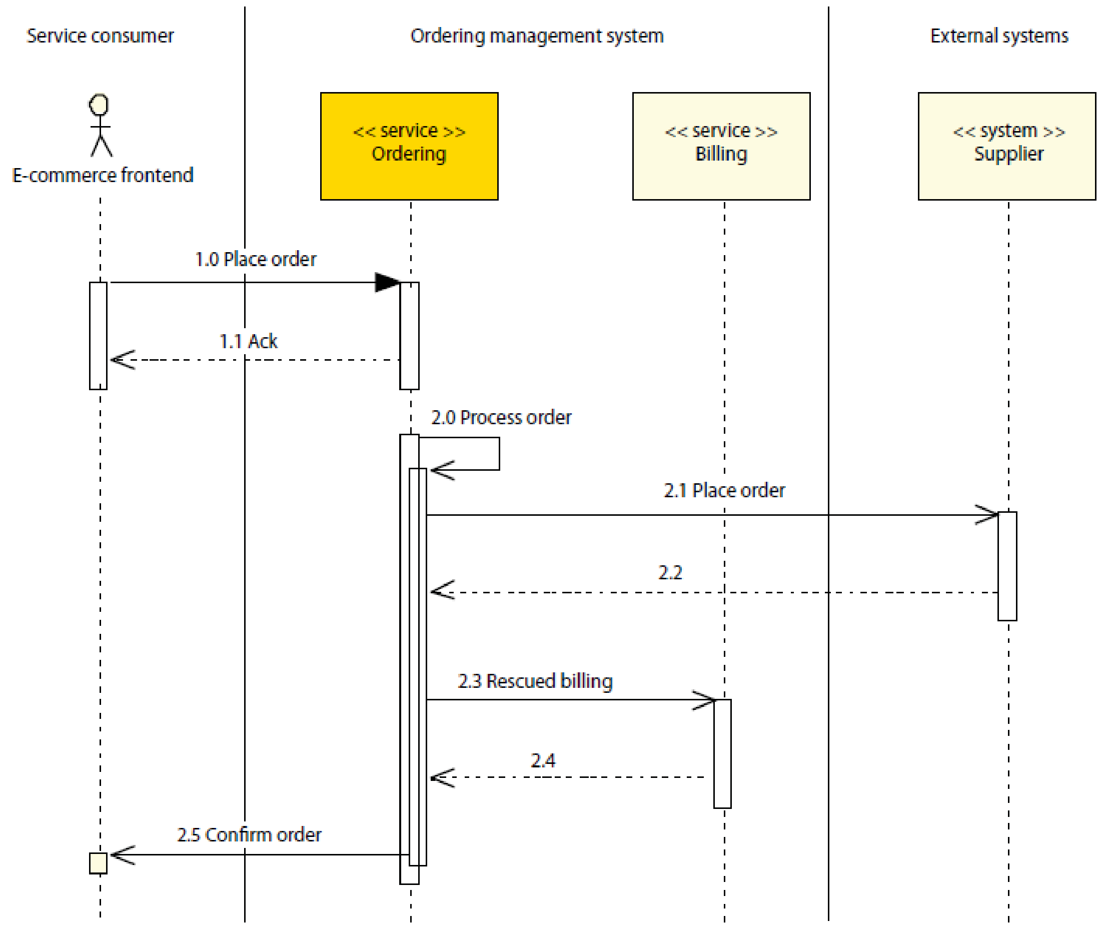


**Figure 2.8** The frontend sends an order to an Ordering service that then orders the part from a supplier and asks a billing service to bill the customer.

# SOA Design Patterns – Transactional Service Pattern

**SOLUTION**
Apply the Transactional Service pattern to handle the entire message flow, so that everything from receiving a request message to sending out a response is contained in a single transaction.

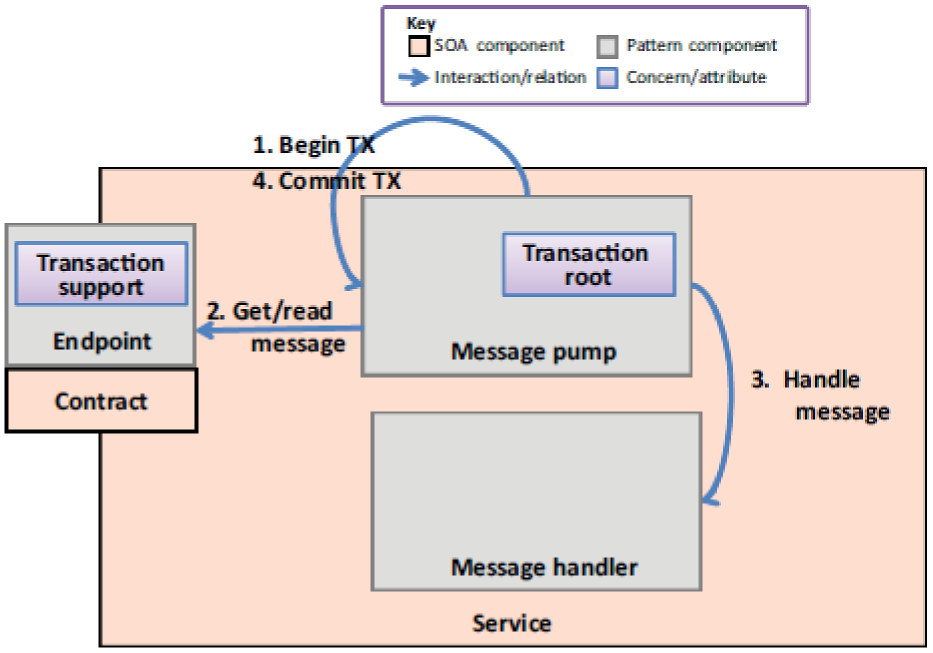
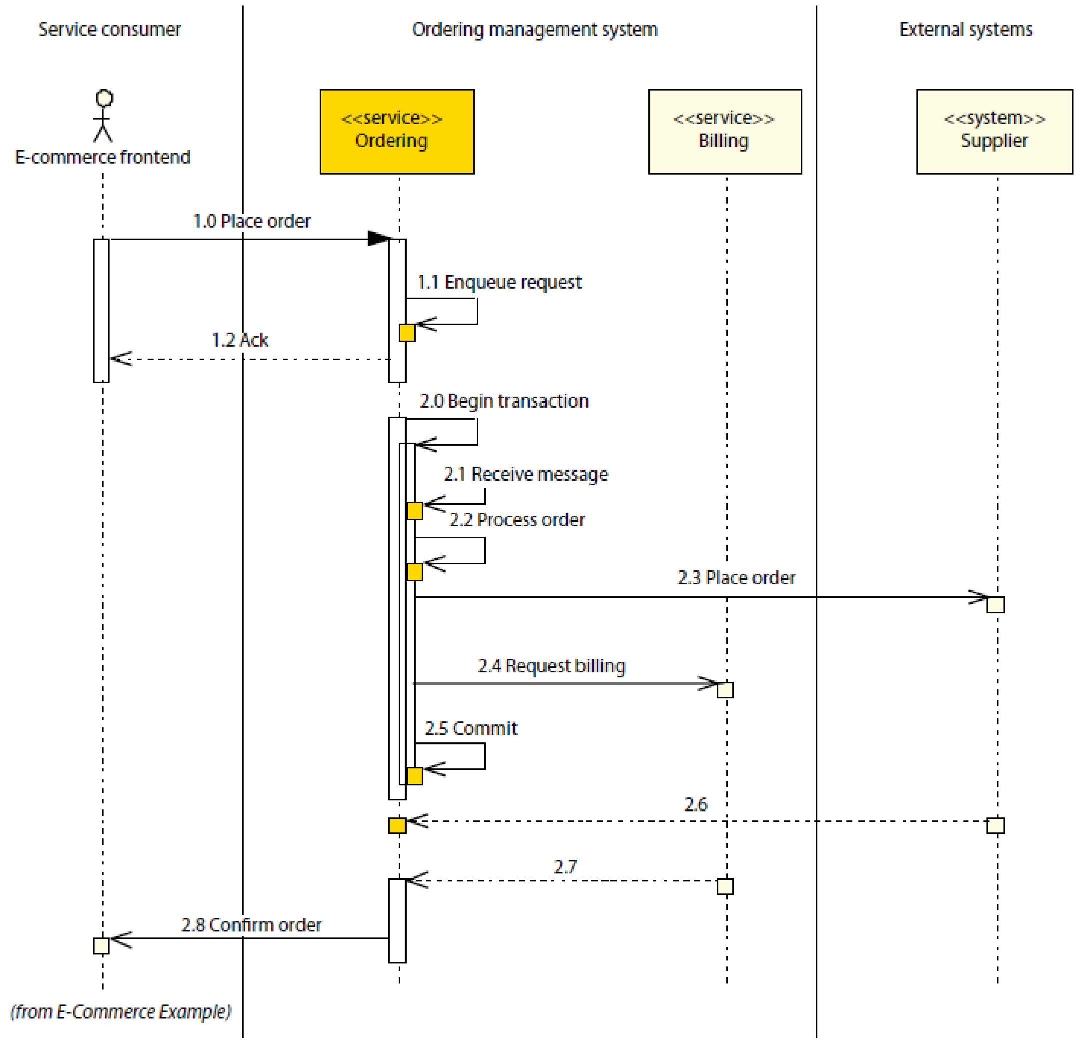


Figure 2.9 The Transactional Service pattern creates a transaction envelope: it opens a transaction, reads the request, handles the message, sends the response, and closes the transaction.

# SOA Design Patterns – Transactional Service Pattern



## SOA Design Patterns – Workflow Service Pattern

**Problem:** How can you increase a service's adaptability to changing business processes?

**SOLUTION**
Introduce a workflow engine within the service to handle the volatile and changing processes and orchestrate the stable logic.

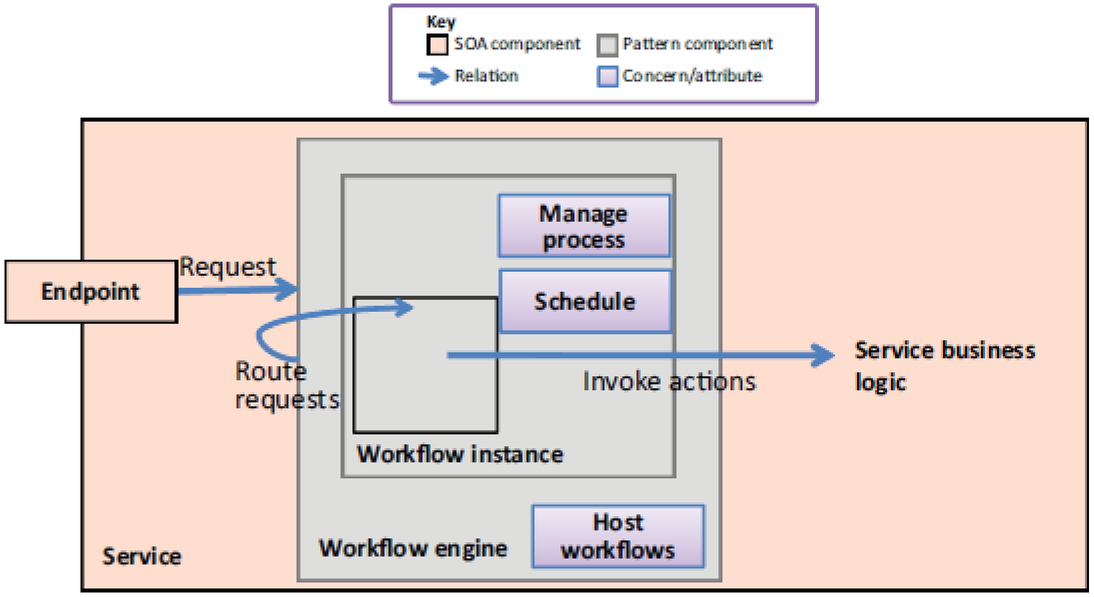


Figure 2.11 The business process is made of the small building blocks that are relatively easy to rearrange. The workflow drives the business logic.

# SOA Design Patterns – Workflow Service Pattern

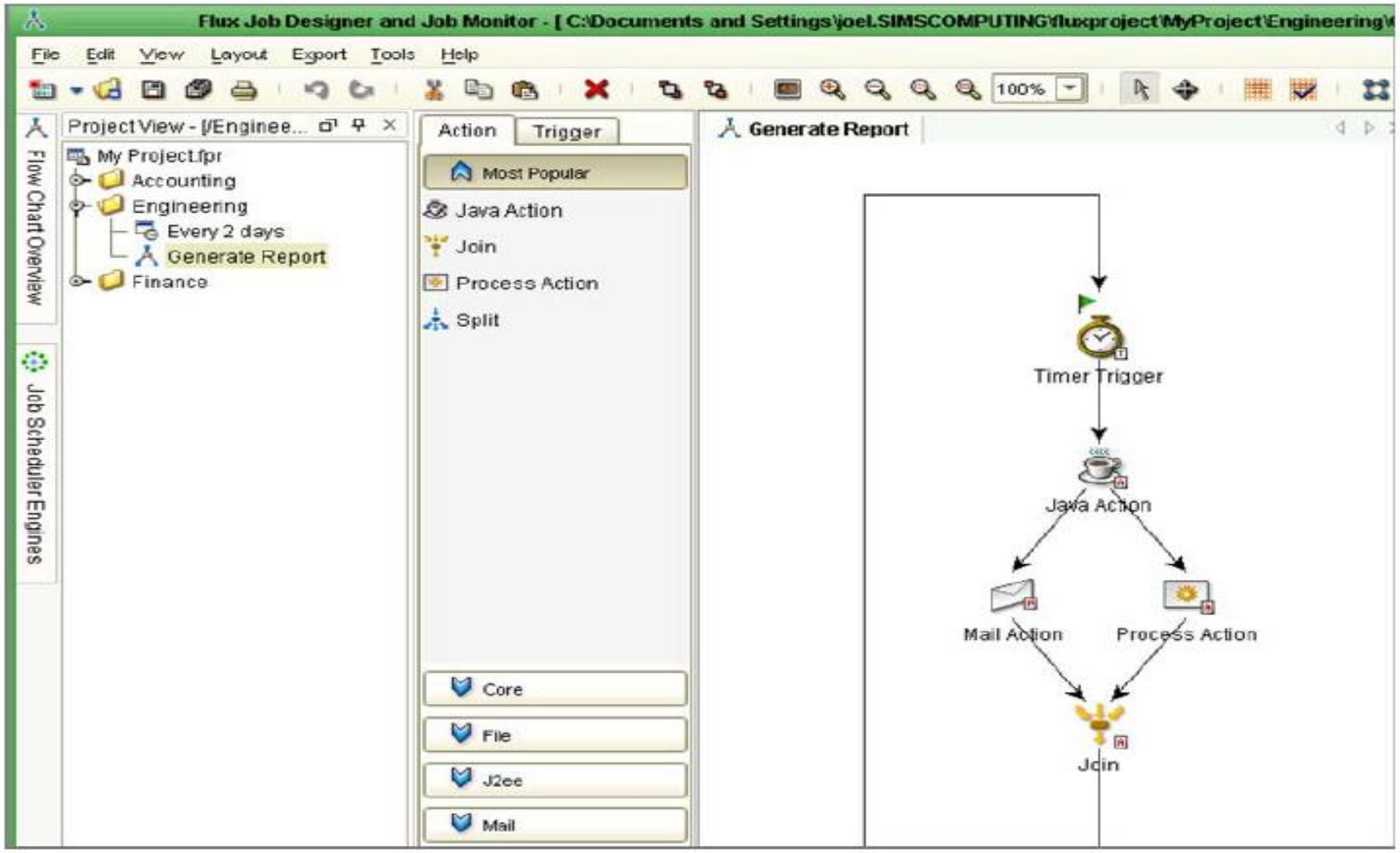


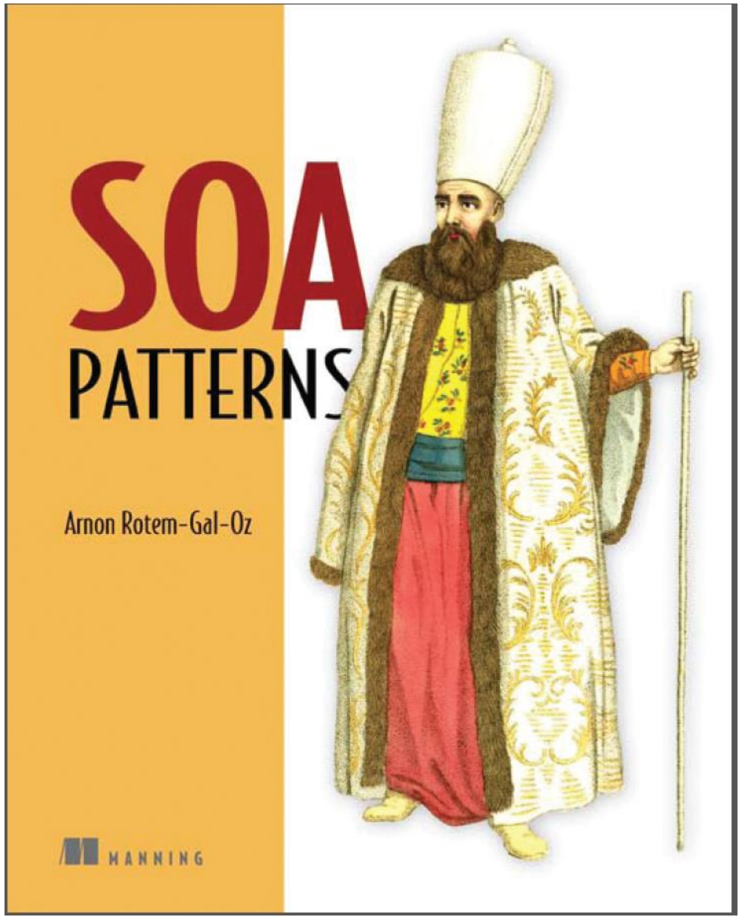
**Figure 2.12** **Most workflow engines come with a visual designer tool to help model workflows.**

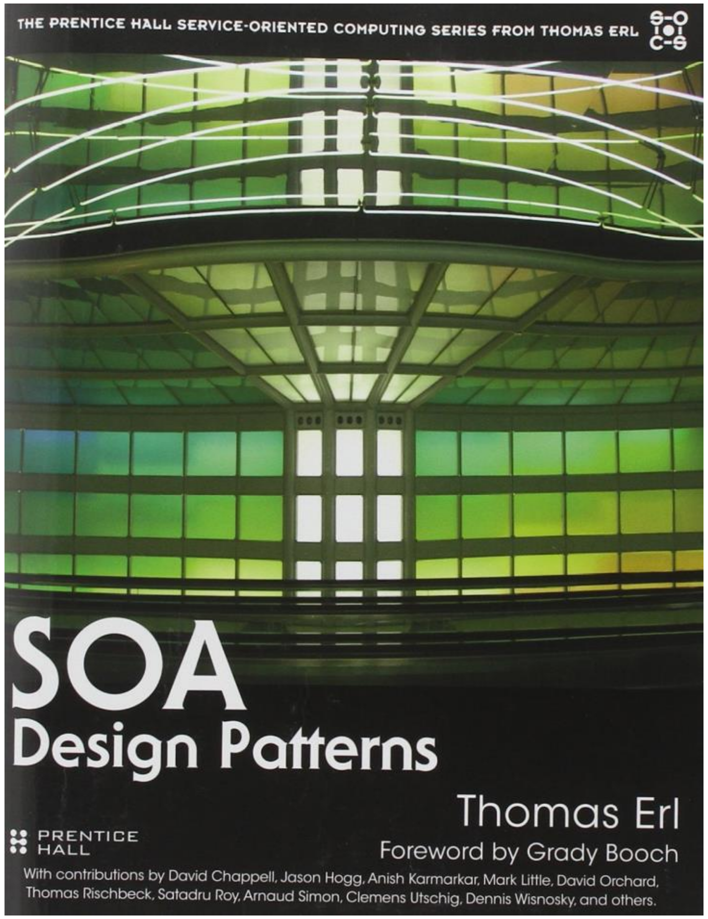
**Listing 2.1 Partial XML of a credit approval workflow implemented for jBPM**

```
<start-state name="start">
<transition to="credit approval"></transition>
</start-state>

...
```

# SOA Design Patterns




THE PRENTICE HALL SERVICE-ORIENTED COMPUTING SERIES FROM THOMAS ERL
SOA Design Patterns
Thomas Erl
Foreword by Grady Booch
PRENTICE HALL
With contributions by David Chappell, Jason Hogg, Anish Karmarkar, Mark Little, David Orchard, Thomas Rischbeck, Satadru Roy, Arnaud Simon, Clemens Utschig, Dennis Wisnosky, and others.