A Pattern-based Constraint Description Approach for Web Services

Qianxiang Wang, Min Li, Na Meng, Yonggang Liu, Hong Mei
School of Electronics Engineering and Computer Science, Peking University
Key laboratory of High Confidence Software Technologies (Peking University), Ministry of Education
Beijing, China, 100871
{wqx, limin05, mengna06, liuyg05, meih}@sei.pku.edu.cn

Abstract
Current WSDL-based description specifies only the operation signatures and locations of Web services. It is widely recognized that a richer description is needed. WS-Security, WS-Reliable, WSOL, and service interaction protocols are all typical efforts toward this direction. Based on the cognition that a complete description of service is impossible, this paper proposes a pattern based constraint description approach for Web services. We illustrate the approach with a constraint pattern hierarchy which focuses on the messages exchanged between client and server. And a RDF based constraint description language is proposed, so as to give explicit semantics to the specified constraint descriptions. The application on a web-based auction system shows that our approach is feasible.

1. Introduction
Web service technology is widely expected to enable the interoperation of heterogeneous systems and reuse of distributed functions. However, the shift of industry towards this technology is slower than researchers have expected [1]. Some of the typical contributing factors are lack of support of security and transaction, insufficient description of service [2].

Web services description is useful for both service client and service administrator. The client can use it to choose the expected service, and generate requests to services. The service administrator can use it to verify the behavior of Web services, so as to find potential behavior deviation and keep the quality of service.

1.1. Problem
WSDL is the main description language of Web services. Different from OMG-IDL [3] which provides only definitions of operation signature, WSDL adds the capability of describing locations of services. But the enhancement of location information is not enough. Ever since researchers realized that many activities, such as service discovery, service composition, and service request need more descriptive information about the services, extending the specification of Web services has become a hot topic in Web services research community. The exploration so far in this direction has gained some achievements, e.g. WS-Security [4], WS-Reliable [5], WSOL [6], and service interaction protocols [7].

However, research works on software architecture have revealed that specifications of software systems are intrinsically incomplete [8]. The system quality depends not only on computational functionality, but also on other properties such as performance, capacity, and environmental assumptions. Different tools may depend on different properties, and some tools may even generate new specification information [9]. The developers cannot anticipate all of the aspects that the clients might care about. Similarly, for Web services which are integrations of software, database, hardware, and network, they are suffering from the description problem, too.

1.2. Approach overview
Although it is nearly impossible to make out sufficient specifications to describe all Web services completely; in this paper, we still try to find a feasible approach to make easy description of constraints on web service. And as more work is done, the approach will show more and more advantages in fulfilling tasks or approaching the final target.
Although signatures of different services are quite different, there are many similarities among the constraint descriptions of different services. This conception leads to our pattern-based describing approach. This paper focuses on the formal constraints description on messages exchanged between clients and servers, and constructs a pattern-based constraint description system. And the formally described constraint can be used by both service clients as well as service administrators: by following the constraints, clients will submit appropriate requests and consume service comfortably; by monitoring request and reply messages, administrators can manage and even control the web service in a better way meanwhile.

Pattern was introduced into property specification in [10], which emphasized the interaction property for finite-state verification. People can identify similar requirements in their systems and select patterns to address them. This thought was followed by some other researchers [7] [15] [18]. Compared with these works, the contributions of this paper include: 1) a pattern hierarchy of Web services constraint, which covers value constraint and event constraint; 2) a formal description language (WSCDL) for pattern-based Web services constraint.

The remainder of the paper is organized as follows: Section 2 explains the concept of service constraint in this paper with a web-based auction system example. Section 3 introduces an overview of our pattern system, which is a general one and can be used beyond web services. Section 4 shows the web service based constraint specification language. Section 5 reports an application of the proposed approach to a web-based auction system. Section 6 introduces related work, and section 7 concludes this paper with our ideas for further work.

2. Service constraint

Constraint is a long time discussed concept, usually as an attachment of function. Actually, most non-functional software requirements can be considered as constraints. Some researchers even consider function as one special kind of constraint. In this paper, constraint is discussed in the context of Web services. The descriptions are based on WSDL. That means that all constraints are about elements which are defined in a WSDL file. In order to make clear the constraint description of web service, we use a web-based auction system to illustrate related issues.

The considered auction system is an enhanced version of an example in [7]. The system is composed of three kinds of web services: “Auction”, “CreditAuthentication”, and “Payment”. Among the three, auction service is the principal part: sellers publish the information about any article on sale; and on the other hand, the bidders register or login the system, bid for an article or retract a bid. Besides, CreditAuthentication service, used by the auction service when handling one’s registration request, can authenticate a person’s identity, and return the result whether the person can be accepted by the system. Payment service—performing an automatic payment—is used by the auction service when a successful transaction finishes. Figure 1 shows the WSDL description of the auction service partially. That is only descriptions related with operation “opBid” are listed.

```
<wsdl:operation name="opBid">
  <wsdl:input message="tns:opBidRequest"/>
  <wsdl:output message="tns:opBidResponse"/>
</wsdl:operation>
```

Figure 1. Definitions of operations “opBid”

Some constraints in this example are:
Constraint 1: The parameter “pwd” of “opRegister” is of type “String”, and consists of only letters “a-z” as well as numbers “0-9”. Additionally, the length is not longer than 16, not shorter than 8.
Constraint 2: All parameters which are instances of type “Price” are instances of type “floats” with two fraction digits.
Constraint 3: The interval between two “opBidRequest” operations should be longer than one second.

Constraint 4: For each bid session, all requests to “opBidRequest” can only happen after the request to “opLogin”.

The complete description of the auction system is available at the home page of our research group: http://www.sei.pku.edu.cn/~wqx/mass/auction-example.

3. Pattern-based constraint description

Although function-centered “operation signature” descriptions of WSDL have been successfully solved, formal complete function description of Web service is still a difficult issue. To our understanding, one of the important factors that lead to this difficulty is that the functions between different web services are too different from each other.

For constraints of Web services, things are quite different: although we cannot list constraints of Web service completely, there are still many similarities among the constraints of different services. This situation motivates our pattern-based formal constraint description approach, as it is said that “each pattern describes a problem which occurs over and over again … and describes the core of the solution to that problem in such a way that you could use this solution a million times over without doing it the same way twice” [11].

As a general repeatable solution to a commonly occurring problem in software design, design patterns have won a great success, and motivated a series of works on pattern, e.g., analysis pattern [13], process pattern [14], and specification pattern [15]. We believe that patterns can be used to capture not only descriptions of recurring solutions to software analysis and design problems, but also descriptions on service system. The description should be displayed in a way easily understood by both client and administrator so that the administrator can identify similar constraints and select patterns to address them as the client can submit request messages following their appropriate understanding.

In this paper, we take two main kinds of constraints into account: value constraint and event constraint. Both of them are divided further into multiple sub-constraints separately. The pattern hierarchy’s framework is shown in Figure 2.

3.1. Value constraint

All interactions between web services and their clients are taken by request and response messages, which are generated according to WSDL files. Value constraints concentrate on parameter values of these messages. Actually, many pre- and post-conditions are constraints on parameters and these constraints are invaluable in determining whether a request is valid, or certain web service behaves correctly. We specify this kind of constraint with XML data type and their constraint facets [16]. We have also divided it into three sub-patterns so far: the Single Value Constraint, the Type Constraint and the Inter Value Constraint.

- Single Value Constraint

This sub-pattern is used to specify the value range of a specific parameter in a message. Constraint 1 in the web-based auction example is a typical single value constraint.

In WSDL, only the type information is declared. With this pattern, we can apply any of the 12 constraint facets in XML Data Type (minExclusive, minInclusive, maxExclusive, maxInclusive, length, minLength, maxLength, totalDigits, fractionDigits, pattern, enumeration, whiteSpace) to parameters. These constraint facets are not only sub-constraint patterns belonging to the single value constraint pattern, but also operators for expressions in value constraints.

- Type Constraint

Some WSDL file defines “User-derived datatype” using “Built-in datatypes” in the XML Data Type when there are multiple parameters that have similar restrictions. We can attach the type constraint to this kind of type definition. With this pattern, we can apply any of the 12 constraint facets in XML Data Type (minExclusive, minInclusive, maxExclusive, maxInclusive, length, minLength, maxLength, totalDigits, fractionDigits, pattern, enumeration, whiteSpace) to parameters. These constraint facets are not only sub-constraint patterns belonging to the single value constraint pattern, but also operators for expressions in value constraints.

- Inter Value Constraint

This sub-pattern employs the same constraint facet operators with the “SingleValueConstraint” sub-pattern. The only difference between them is that the former is applicable to a group of parameters that are instance of the same type, while the latter is applicable to a single parameter.
computational operators: from “Add, Subtract, Multiply, Divide” to “Mod”. Because the computation result is still a value, all the constraint facets in Single Value Constraint Pattern are still applicable to this pattern.

3.2. Event constraint

Event constraints are usually temporal rules on occurrence of messages. Many APIs of library have implicit event constraints. For example, Java standard class library v1.3.1 contains 914 classes, out of which at least 81 classes have method temporal constraints [17]. We have derived two sub-patterns from this constraint pattern so far: the Time Constraint and the Order Constraint.

- Time Constraint

This pattern is used to specify time aspects of messages, without order relationship with other messages. Constraint 3 in the web-based auction example is a typical Time Constraint.

Three further sub-patterns have been derived so far: 1) frequency constraint, which means the times that a message can occur in certain duration; 2) interval constraint, which represents the interval between two messages’ occurrence; 3) response time constraint, which refers to the period between request and corresponding reply messages. These three sub-patterns are quite common constraints in Web services context.

Time constraint can be applied to messages of some specific operation, or messages of all operations. Constraint 3 is about the request message of operation “opBid”. It can be applied to messages of all operations: the interval time between all operations should be longer than one second.

- Order Constraint

This pattern is used to describe the occurrence sequence of some messages, such as pre- and post-conditions, and other limitation. Constraint 4 in the web-based auction example is a typical Order Constraint.

We proposed four further sub-patterns to this pattern: “precede”, “lead to”, “last”, and “next”. 1) “Event A precedes event B” means when event B happens, then event A must have happened at least once before. 2) “Event A leads to event B” means once event A occurs, before the end of the whole event sequence, event B must occur later. 3) “Last event of event A is event B” means that in the whole event sequence, event B is the immediate prior one of event A. 4) “Next event of event A is event B” means that in the whole event sequence, event B is the direct next one of event A.

Many researchers have mentioned scope and condition for order constraint patterns [7] [10]. We consider them as extensions to constraint patterns and will discuss them in section 3.4.
“Precede” Constraint Pattern

- **Name:** Precede Constraint Pattern
- **Intent:**
  This pattern is used to specify the pre-condition of an event. “A precedes B” means at any time when “B” occurs, then at the previous moment at least one “A” must have happened.
- **Capability:**
  We make the precise definition of this pattern by PTLTL (Past Time Linear Temporal Logic).
  \[ \Box (B \rightarrow \Diamond A) \]
- **Example:**
  Constraint 4 of the auction example proposed in Section 2 is an instance of this pattern, that is, “opLogin” (L for short) precedes “opBidRequest” (B for short). You can tell whether an event sequence is a valid one at the occurrence of B. For example, “*LB*B” and “LB*LB” are valid ones, while “*BX*B” is invalid (“*” represents any other event except L and B).
- **Relationship:**
  This pattern has some similarity with “Last” pattern, and they are all about pre-condition. While the latter one is about the constraint of immediate previous event. It is also easy to be confused with the “Lead to” pattern. “A precedes B” is not equal to “A leads to B”, and the former means when B happens, there must be some A in the past, while the latter means when A happens there must be some B in the future.

3.3. Pattern template

Most researchers [10] [15] used variation of the design pattern template developed in [12]. The constraint pattern template contains fields of: Name, Intent, Capability, Example, and Relationships. An example of preceed constraint pattern definition is given in Figure 3, and the definition for other patterns can also be found at our web site: http://www.sei.pku.edu.cn/~wqx/mass/auction-example.

3.4 Pattern extension

In this paper, the pattern-based approach is used to solve the incomplete description problem of Web service. One benefit of the pattern-based approach is that it is easy to extend when a new constraint is introduced.

There are two way to extend the pattern hierarchies: horizontal way and vertical way. With horizontal extension, we mean when some new constraints are discovered and can not be covered by current pattern hierarchies, we can add new patterns to the hierarchy. For example, we can consider WS-Security, WS-Reliable, and prices of web service as extensions in the same level with Value Constraint and Event Constraint. With vertical extension, we mean that descriptions belong to some known patterns, but need some more specifications. The typical vertical extension examples are time scope and parameter conditions.

Figure 3. Definition of “Precede” Constraint Pattern

According to [10], 10 percent of constraints have scopes, which are the extent of the program execution that pattern must hold. There are four main kinds of scope: “before”, “after”, “between…and”, and “after…until”. One example of scope is that “before event C happens, event A precedes event B”.

Parameter condition is used to filter some specific messages, and set constraint on these messages. Paper [7] provided details about parameter condition.

4. Constraint description language

In this section, we focus on a description language which is called WSCDL: Web Service Constraint Description Language.

4.1. Principle

WSDL, as one of the three crucial techniques of Web services, provides the basic approach for service description. This description is important for service consumer’s service discovery and service usage. But only the description on operation signature and location of service is far from directing the consumer to use the service correctly. So we extended the description by declaring more information about parameter and interaction property in our WSCDL file. The WSCDL has the following features:

(1) We separate the basic description and the constraint description by defining them in WSDL and WSCDL file respectively. The former
information is functional and relatively stable, while the latter is optional and configurable. In such a way, we can change or remove the constraint without modifying the basic description information in WSDL.

(2) The syntax of WSCDL is quite brief. Our specification approach is based on RDF (Resource Description Framework) [19], and we focus on the triple concept of “<subject, predicate, object>”.

(3) It inherits the extensibility mechanism of RDF naturally. You can add your own specification of the newly discovered constraints by extending the SOAC (Specification of a Constraint, which is spurred by DOAP [14]) Schema.

(4) Reusable specification is supported in our approach. If there are two same constraints in different services, we can give the integrated specification in one WSCDL, and the other one may refer to it instead of specifying once again.

WSCDL can be considered as a complement to WSDL, and it lies in the same layer (service description layer) with WSDL in the protocol stack of Web services. Each WSCDL file is attached to certain WSDL file, and their corresponding relationship can be indicated by their identical main file name but with different postfixes. It may also refer to other WSCDL with the “import” keyword declaration (dotted line in Figure 4) to reuse the constraint specification.

![Figure 4. Relationship between WSCDL and WSDL](image)

### 4.2. The structure of WSCDL

The general structure of WSCDL is quite simple. Just like other XML files, at the beginning we must declare the namespace information. Then the main body of WSCDL is composed of one or more items of constraint specification, whose amount is decided by the actual number of constraints of the corresponding service. In WSCDL, the “<constraint>” tag is the direct child tag of root element “<constraints>”, and it marks the beginning of the specification of an item of constraint. The specification of every item of constraint comprises several fixed parts. The structure is showed in Figure 5. Two examples of WSCDL are shown in section 5. Here is the description of each part:

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
<th>Description</th>
<th>Context</th>
<th>Condition</th>
</tr>
</thead>
</table>

(1) Name: Each item of constraint has a “name” attribute and it is unique in every WSCDL. If there are similar constraints in other WSCDL, you can reuse it by referring to the name.

(2) Category: This attribute is also necessary for each item of constraint, because we may take particular actions for particular kind of constraints in the later process such as service validation and so on. It represents which pattern it belongs to, and the value must be one of the strings in the black rectangles with bolder frame in Figure 2.

(3) Description: It is a general description of the constraint in natural language, which help the person who read the WSCDL file to understand the constraint.

(4) Context: A sub-element of a constraint. It specifies which parameter or which event the constraint is about. It may be composed of several “subject”, and each “subject” can be defined by the name of the service, operation, data type, parameter or the direction of parameter (“in” for input parameter, and “out” for return value). Subject of different pattern includes different attribute items.

(5) Content: This sub element specifies the details of the constraint by the appropriate facets of the corresponding pattern. Examples will be introduced in the next section.

(6) Condition: This part is optional and it defines the applicable condition of the constraint, and event constraint tends to have the condition property. For most value constraints, it’s just empty. It may be composed of value condition (expression of parameters) and event condition (scope mentioned in section 3.4). The expression may contain logic operators like “AND”, “OR” or “NOT”. If an item of constraint is similar to another one in other WSCDL and you want to reuse it, you need not declare it again. Instead, you can use the “ref” attribute of “constraint” and refer to the name of the reusable constraint. This part is optional and is visible only when the reusable constraint is available.
4.3. Tool support

To extend the use of our approach, we have implemented it partially as tools. Therefore, users can specify constraints simply by filling in some forms or drawing some graphs instead of manually editing xml files. Besides, all the tool supports which have been implemented and those to be implemented will present themselves as plug-ins in Eclipse.

4.3.1 Principle of tool support

The WSCDL file which specifies the constraints of certain service is an XML file in essence. As we know, although XML is a simple and convenient format for machines to deal with; to edit such xml files manually is a boring job because it has strict grammar rules which are liable to be violated. In order to help users write WSCDL files with less effort by filling in forms or drawing in a picture editor, we have developed the tool called SOAC. Eclipse is an extensible framework and our tool is implemented as an Eclipse Plug-in. It can parse WSDL files, get necessary information, fill in forms, and then generate WSCDL files automatically after users finish specification complying with given schemas.

4.3.2 Graphical representation

To represent order constraints mentioned above in a visual way, we have introduced Event Constraint Graph (ECG) and its formal representation. And here we will introduce the ECG’s important features.

Order constraint on future event

Two notations “——>” and “---->” are adopted to express two kinds of future relationship: “next” and “lead to”. 1) “Next event of event A is event B” means that in the whole event sequence, event B is the direct next one of event A. 2) “Event A leads to event B” means once event A occurs, then before the end of the whole event sequence, event B must occur.

![Figure 7. Notations of constraints on future](image)

Order constraint on past event

Two notations “>——” and “>----” are employed to represent two kinds of past relationship: “last” and “precede”. 1) “Event A precedes event B” means when event B happens, then event A must have happened at least once before. 2) “Last event of event A is event B” means that in the whole event sequence, event B is the immediate prior one of event A.

![Figure 8. Notations of constraints on past](image)

Propositional operators

Notations for propositional operations “OR”, “AND” and “NOT” are also necessary for order constraint specification.
We import a dashed arc between constraints to show “OR” operation. Similarly, when the relation between different constraints is “AND”, the arc will be solid. And by default, the relation between different constraints should be “AND” if no arc is drawn.

![Figure 9. Arcs of “OR” and “AND”](image)

In Figure 9 (a), with the help of “OR” arc, the order constraint can be described as

\[ C = \text{OR} (\text{next}(e, e1), \text{leadto}(e, e2)) \]

In Figure 9 (b), with the help of “AND” arc, the order constraint can be described as

\[ C = \text{AND} (\text{last}(e, e1), \text{precede}(e, e2)) \]

Besides, if we hope to express constraints like “event A must not lead to event B”; we will use the notation “\( \neg \)” on order edges to help express such “not” relation.

Additionally, to express constraints related with the beginning or ending of event sequence, two special symbols are imported to express two special events respectively as well.

- ‘start’: no event occurs before. This special event represents the beginning of the whole event sequence.
- ‘end’: no event occurs later. This special event represents the ending of the whole event sequence.

With the representation mentioned above, we can express C4 constraint marked already in Section 2 in the following way (See Figure 10).

![Figure 10. ECG for C4](image)

### 5. Case Study

We apply the proposed approach to the auction example mentioned in the second section. This example is a quite representative one, and we list 14 items of constraints in all, 6 items for value constraints and 8 items for event constraints. The constraints in this example cover 9 of the 10 patterns proposed in Figure 2. As for the whole list of constraints and the integral version of the WSCDL file, please refer to our web site [http://www.sei.pku.edu.cn/~wqx/mass/auction-example](http://www.sei.pku.edu.cn/~wqx/mass/auction-example).

Due to the space constraint, here we specify only two items of constraints to show the expression examples of WSCDL.

![Figure 11. Snapshot of SOAC for specifying C1](image)

Figure 11 is the snapshot of SOAC when specifying the “minLength” facet of C1 mentioned in Section 2, which is about the “pwd” parameter of “opRegister” operation. Currently user can specify only one facet at a time and for C1 which has three facets (“minLength”, “maxLength” and “Pattern”) we have to specify three constraints with each at a time. This is a limitation of the current version and in future we will improve it. And Figure 12 is the automatically generated WSCDL file fragment for C1. Here no condition is needed.

```xml
<Constraint>
  <name>AuctionopRegisterpwd0</name>
  <category>SingleValueConstraint</category>
  <description>The “pwd” parameter of “opRegister” operation should be a String composed of letters and numbers, and the length should be no longer than 16 and no shorter than 8</description>
  <context>
    <subject name="pwd">
      <serviceName>Auction</serviceName>
      <operationName>opRegister</operationName>
      <parameterType>string</parameterType>
      <index>1</index>
    </subject>
    <content>
      <minLength>8</minLength>
      <maxLength>16</maxLength>
      <pattern>[a-zA-Z0-9]+</pattern>
    </content>
  </context>
</Constraint>
```

![Figure 12. Specification of C1](image)
involving “opBid” operation and “opLogin” operation. Here the constraint has a condition “the same person”, and we employ a parameter expression to express it.

```
<Constraint>
  <name> BidLoginConstraint </name>
  <category> PrecedeConstraint </category>
  <description> To the same person, when he sends the “opBidRequest”, he must have done the “opLogin” operation. </description>
  <context>
    <subject name="opLogin"> 
      <serviceName>AuctionService</serviceName> 
      <operationName>opLogin</operationName>
    </subject>
    <subject name="opBid"> 
      <serviceName>AuctionService</serviceName> 
      <operationName>opBid</operationName>
    </subject>
  </context>
  <content>
    <event1="opLogin"/>
    <event2="opBid"/>
  </content>
  <condition>opLogin.userName=opBid.userName</condition>
</Constraint>
```

Figure 13. Specification of C4

From these two examples, we can conclude the sub elements of “subject” and “content” are quite different with different patterns. We describe the syntax of WSCDL by RELAX NG (REgular LAnguage description for XML Next Generation) schema.

6. Related Work

The two key contributions of the proposed approach are enhanced description for Web services and pattern-based description. We introduce related work from these two views separately.

There are many enhancement works on Web services description. While WS-Security [4] and WS-Reliable [5] have gained wide focuses, there are still many other enhancement works. WSOL [6] is a formal specification language for many kinds of constraints such as functional constraint, Quality of Service, access rights, prices, monetary penalties, etc. WSOL can be widely used in fields of quality evaluation, monitoring, and managing of Web services. Paper [2] extended the functionality of the Web services broker to include constraint specification and processing, which enable the broker to find a good match between a service provider’s capabilities and a service requestor’s requirements. But it concerns only the parameter constraints of the operation with the goal of getting a better matching service to the service consumer’s request by their constraint matching algorithm.

Dwyer introduced pattern into property specification firstly in [10]. The author argued that pattern-based specification is easy to understand, and can be used for finite-state verification. Propel [18] extended the speciation patterns introduced in [10] to address important and subtle aspects about a property, such as what happens in a cause-effect relation if the cause recurs before the effect has occurred. Betty [15] followed this thought and created a real-time specification patterns in terms of three commonly used real-time temporal logics. Jun Han [7] introduced pattern to the description of Web services firstly. The proposed pattern is mainly applied to interaction protocols between client and server, especially the temporal constraints on messages. The approach proposed in this paper is for constraint on Web service, which focuses on a different area from those of [18] and [15], and covers more constraints than what is covered in [7].

7. Conclusion

In this paper, we propose a pattern hierarchy of constraint for Web services. This pattern hierarchy is based on the most common constraint classes we have known so far: value constraint and event constraint. The constraint description of Web services is separated from the basic description, by defining them in WSDL and WSCDL files respectively. We have also provided RDF-based specification templates for each pattern. Particularly, in order to ease burden of users when they specifying constraints, we have also developed an Eclipse plug-in tool partially, which can generate corresponding formal WSCDL files automatically from users’ informal constraint specification. Additionally, we have applied the tool to a web-based auction system in the paper and displayed the flexibility of our approach.

The research about constraint is still under way. Our ongoing work includes: 1) detection of potential conflicts between different constraint descriptions; 2) the acquirement of constraints on different Web services; and 3) the application of the constraint specification. Certainly, we will improve the runtime monitoring of service at the same time to help improve the specifying approach.

Acknowledgement
This paper is supported by the National High-Tech Research and Development Plan of China No. 2006AA01Z175, the National Grand Fundamental Research Program of China No. 2005CB321805, and Key National Science Foundation of China, No. 90412011.

References

[12] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1994.