How Rhetorical Objects Resemble Programming Objects

A **rhetorical object** resembles a **programming object** in many ways, but, being part of your content, the rhetorical object exists to communicate with a particular set of audiences for one or more purposes. Such an informative object, then, has a rhetorical goal: to communicate with other people.

In this article, I draw out the analogy between rhetorical objects and traditional programming objects.

—Jonathan Price

Key Points:

Consider a rhetorical object as an instance of a class, inheriting its properties.

As a member of that class, a rhetorical object has a responsibility, or purpose.

A rhetorical object has relationships with other rhetorical objects.

A rhetorical object has attributes.

A rhetorical object communicates via messages with other rhetorical objects.

A rhetorical object can be reused in many different designs.

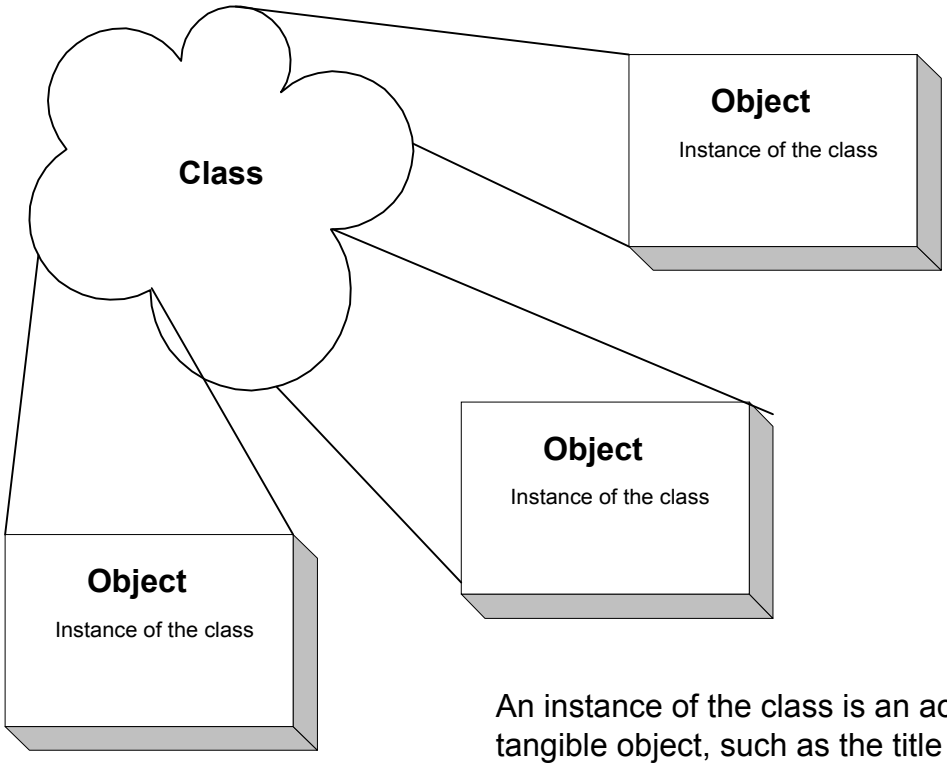
Each rhetorical object has a unique identity, or content.

A rhetorical object may perform operations on its own data.

A rhetorical object changes states.

Consider a rhetorical object an instance of a class, inheriting its properties.

A class is a concept--a title in general, a footnote in the abstract.



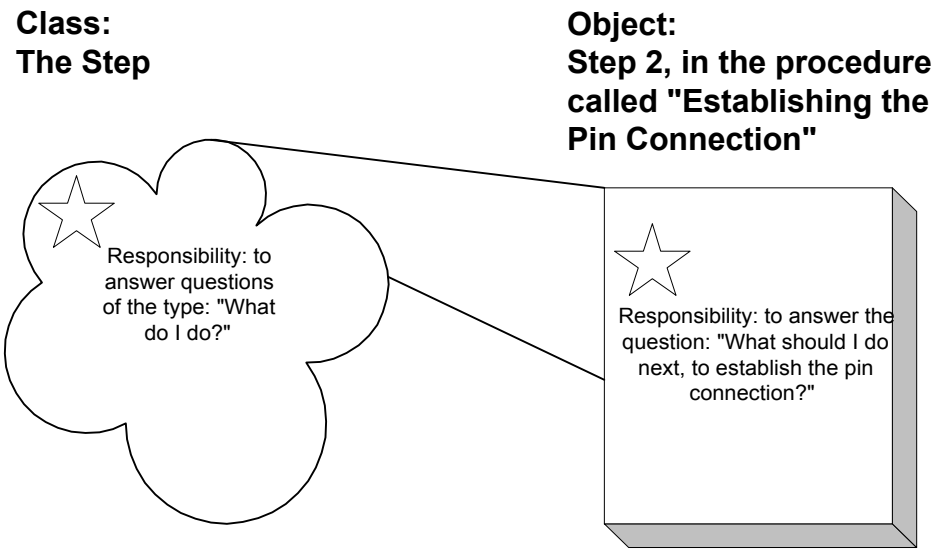
An instance of the class is an actual, tangible object, such as the title of a particular manual, or footnote 53.

Class: Defines the characteristics that will be inherited by every rhetorical object within it, such as:

- Its responsibilities, and type of data
- The messages it can send to other objects
- The types of attributes it has
- The operations it may perform, and, therefore, states it can take on
- The kinds of relationships it can have with other objects

Each object is an instance of the class, inheriting the class properties, but with unique content.

As a member of its class, a rhetorical object has a responsibility, or function.



Each class has one overriding **responsibility**.

- For writers and other communicators, **that responsibility is rhetorical**..
- The rhetorical object responds to our audience's needs, carries forward our conversation with these audiences.

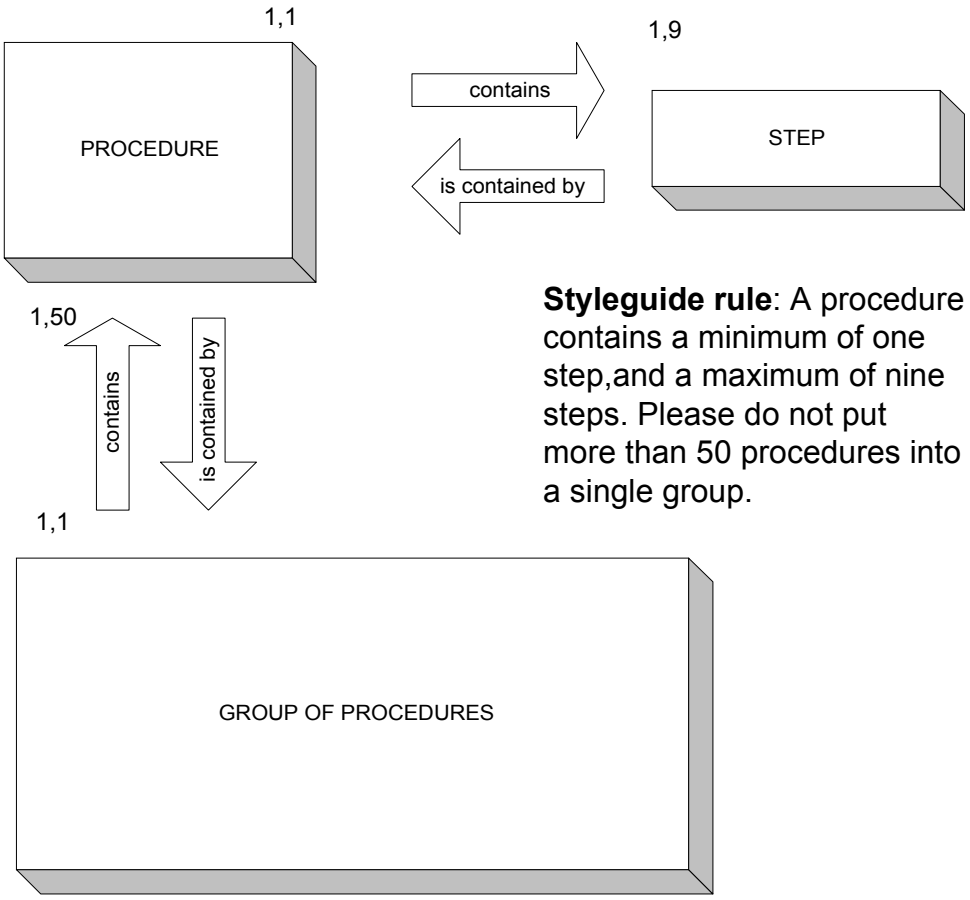
The rhetorical responsibility of each of our objects is **to answer a question** from the audience.

- The class responds to a type of question.
- The instance, the individual object, answers to that type of question, focused on a particular topic.

Wondering whether an element is a member of a class of objects? Ask yourself

- Why was it invented in the first place?
- What does the object have to do, to justify its existence?
- What user question does it answer?

A rhetorical object has relationships with other such objects.



Styleguide rule: A procedure contains a minimum of one step,and a maximum of nine steps. Please do not put more than 50 procedures into a single group.

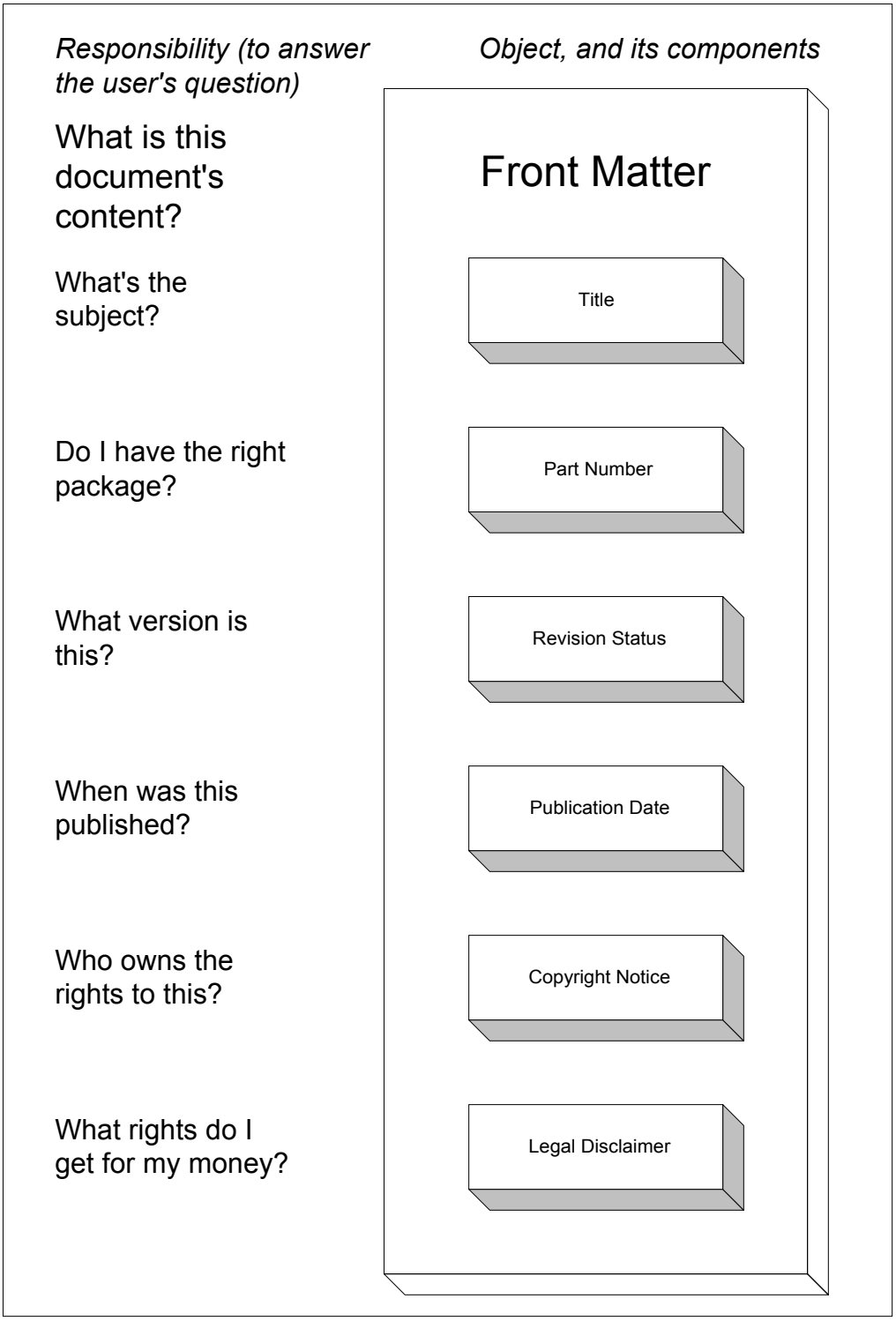
You define what kind of **relationships** can legally exist between or within classes of objects.

- Relationships do not change anything; they simply **are**.
- They represent a **static** connection; they do not perform some operation or send a message.
- You can spell out **how many** objects of one class can have this relationship with a **minimum or maximum** number of objects in the other class.

In the abstract, relationships associate one or more classes of objects through a function, such as *contains*, or *is contained by*.

The **function** starts with an object in one class, goes to another class, and fetches all its objects that are associated in a particular way with that first object. For instance, starting with a *procedure* object, the *contains* function identifies all the *step* objects that belong to that particular procedure.

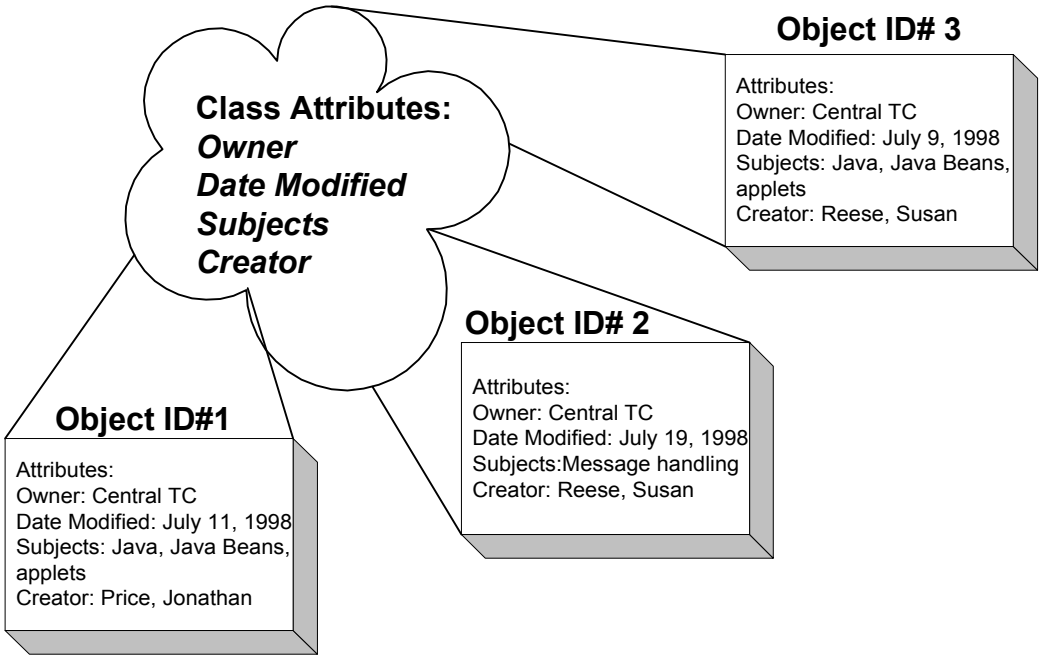
Practical building blocks: You define what objects are components of some other object, and how many of each may be required or optional.



Analysis of Front Matter

Name of Object	Responsibility	Component objects (and #)	Are Components Required or Optional?	Forms a component of what other object?
Front Matter	Answers the question: what is this package about and when did it emerge?	Package title (1), Part Number (1), Revision Status (1), Publication Date (1), Copyright Notice (1) Legal Disclaimer (1)	All Required	Package
Package Title	Answers the question: what is this package about?			Front Matter
Part Number	Answers the question: Do I have the right package?			Front Matter
Revision Status	Answers the question: When was this package updated?			Front Matter
Publication Date	Answers the question: When was this package published?			Front Matter
Copyright Notice	Answers the question: Is this valuable material copyrighted?			Front Matter
Legal Disclaimer	Answers the question: What rights do I get for all my money?			Front Matter

Each rhetorical object has a set of attributes inherited from its class.



The **class** defines what kind of attributes each object will have.

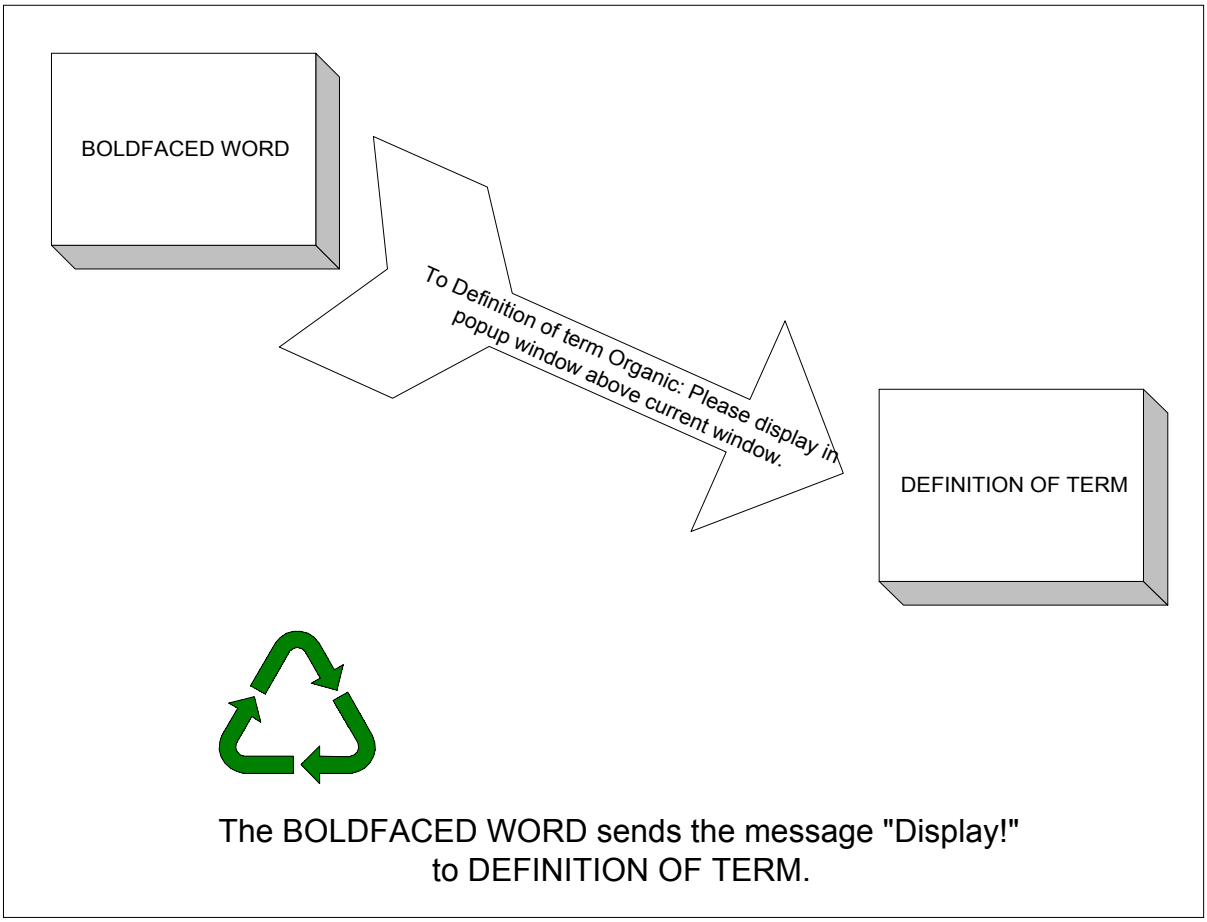
- Each object **inherits** these attribute types, and the preset values (which can change).
- The **values** for each attribute in an object change depending on what operations have been performed on the object, and what state it is in.
- The class defines a set of legal values or data types for each kind of attribute.

Attribute **values** describe qualities or properties of the object; they give us information about its particular data, creator, context, filetype, dates, or circumstances.

Bonus: Using attributes allows us to track changes in an object's state, without having to create a new object.

We can consider the whole set of attribute values as, collectively, representing **the state of the object** right now.

A rhetorical object communicates via messages with other rhetorical objects.



An object sends out **a request** that another object perform some operation, such as displaying itself, or hiding.

- A request contains the name of the object, the requested operation, and, if necessary, parameters for that operation.
- A request is one way an object interfaces with another. The message coming back after the operation has been performed is another.

Request and **response** allow interaction between objects.

Responsibility (to answer user questions)

Object, and its components

Where can I find what I need?

What are the major sections or chapters about?

What are the major topics in the chapter?

What are the subtopics under one of those topics?

What are the sub-subtopics?

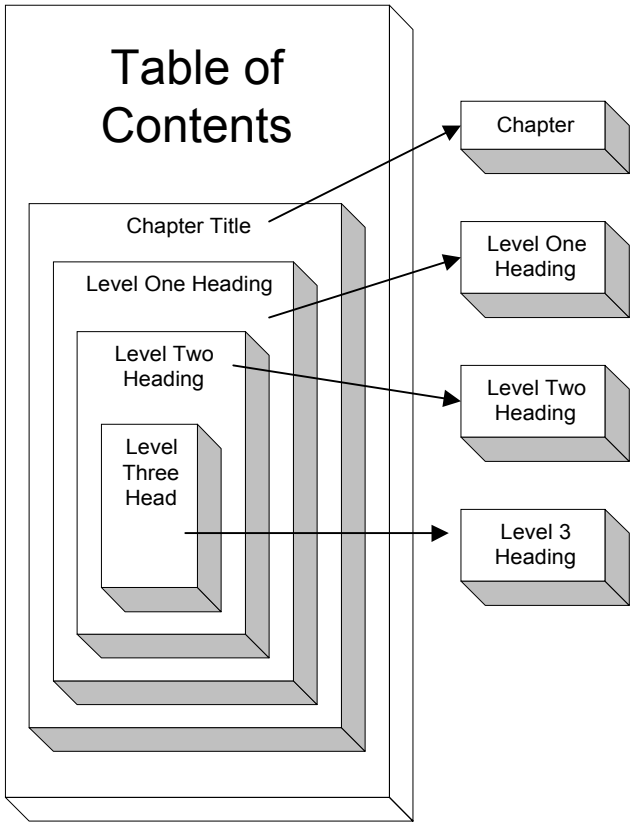


Table of Contents

Name of Object	Responsibility	Component objects (and #)	Are Components Required or Optional?	Sends message(s) to other objects?	Forms a component of what other object?	Is it a Required or Optional Component?
Main Table of Contents	Answers the question: where can I find what I need in the document?	chapter.toc*	1 required		Front Matter	
Chapter Table of Contents	Answers the question: what are the major sections about?	chapter.title (1), level1+, level2*,.level3*	chapter.title required, level1 required, others ptional		Main Table of Contents	At least one required
Level One	Answers the question: what are the main topics in the chapter? (Displays Level One headings from within chapter).	level2*, level3*	Optional	Sends message to major headings in chapter (H1)	Chapter TOC	Required
Level Two	Answers the question: what are the subtopics in this section of the chapter? (Displays Level Two headings)	level3*	Optional	Sends message to minor headings in chapters (H2)	level1	Optional
Level Three	Answers the question: what are the subtopics within the subtopics? (Displays Level Three headings)		Optional	Sends message to H3 headings in chapters	level2	Optional

DTD for Table of Contents

<!ELEMENT main.toc (H1, chapter.toc+)>

<!ATTLIST main.toc anchor ID #REQUIRED>

<!ELEMENT chapter.toc (chapter.title, level1+)>

<!ATTLIST chapter.toc anchor ID #REQUIRED>

<!ELEMENT chapter.title (#PCDATA)>

<!ATTLIST chapter.title target IDREF #REQUIRED>

<!ELEMENT level1 (#PCDATA, level2*)>

<!ATTLIST level1 target IDREF #REQUIRED>

<!ELEMENT level2 (#PCDATA, level3*)>

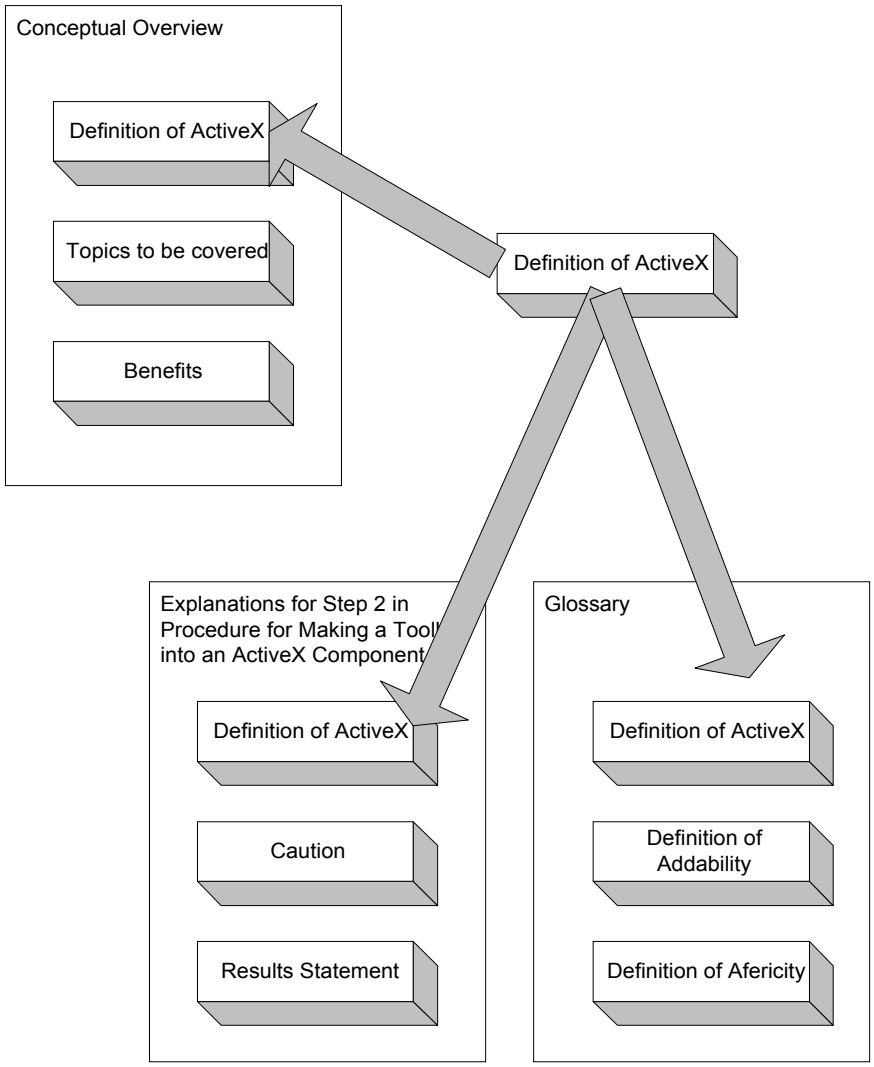
<!ATTLIST level2 target IDREF #REQUIRED>

<!ELEMENT level3 (#PCDATA)>

<!ATTLIST level3 target IDREF #REQUIRED>

In this approach, the chapter sections are conceived of as nested within each other, in a hierarchy. Each line of the table of contents has its own IDREF to link the user to the right section of the chapter.

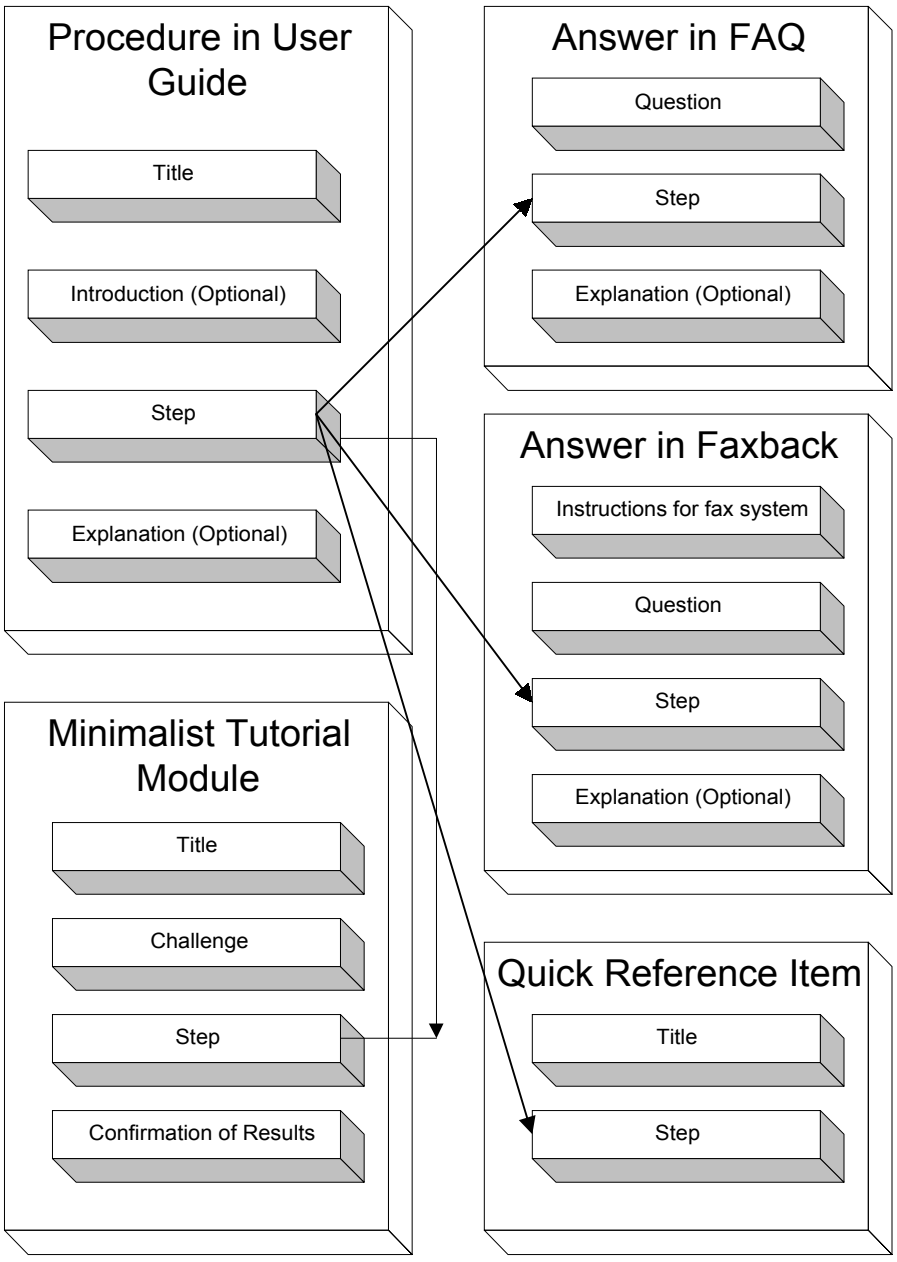
A rhetorical object can be reused in many different designs.

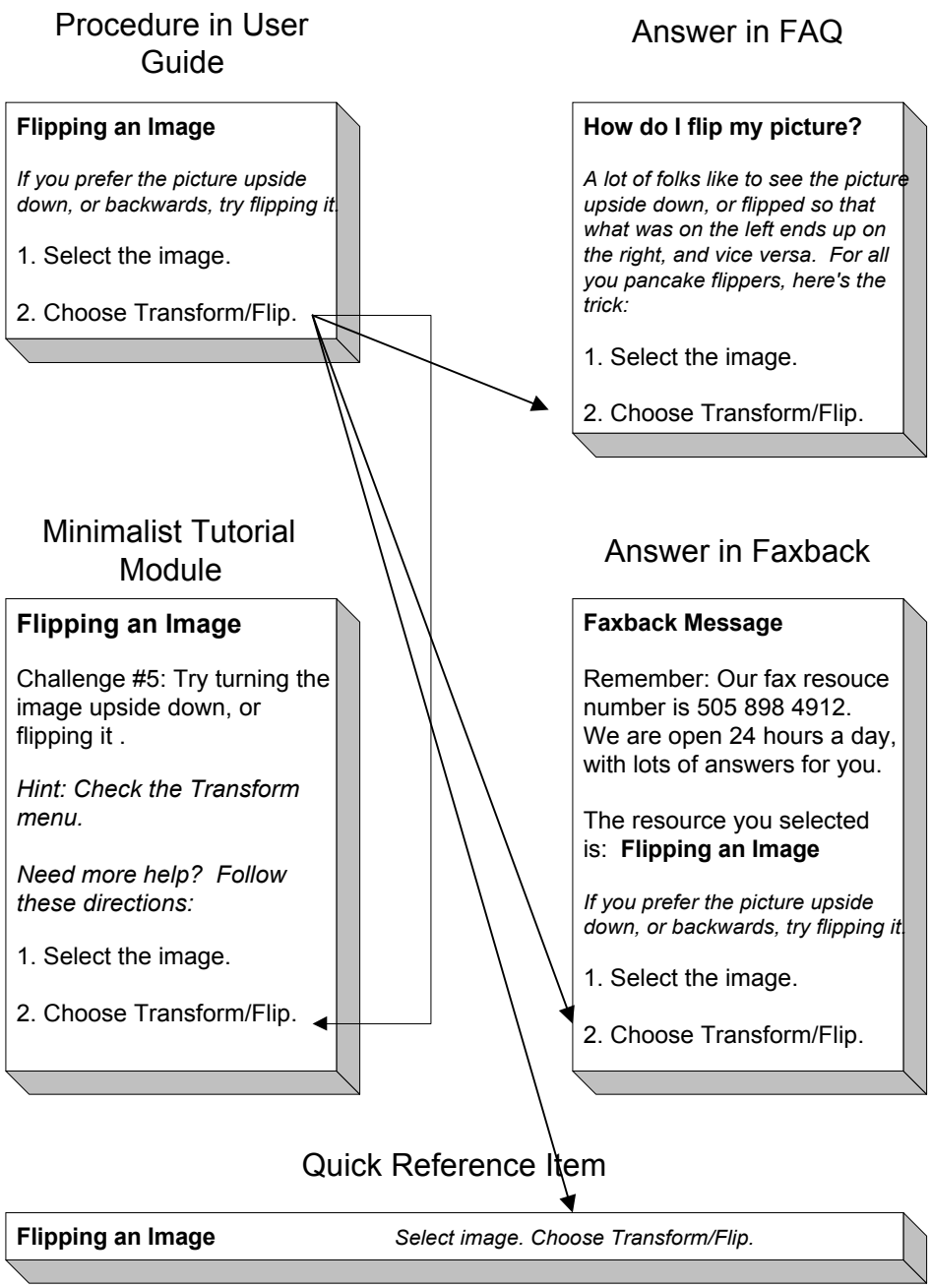


Each object can be used as a component in a different object, or placed in a different position in a sequence of objects, or used without some of its optional components.

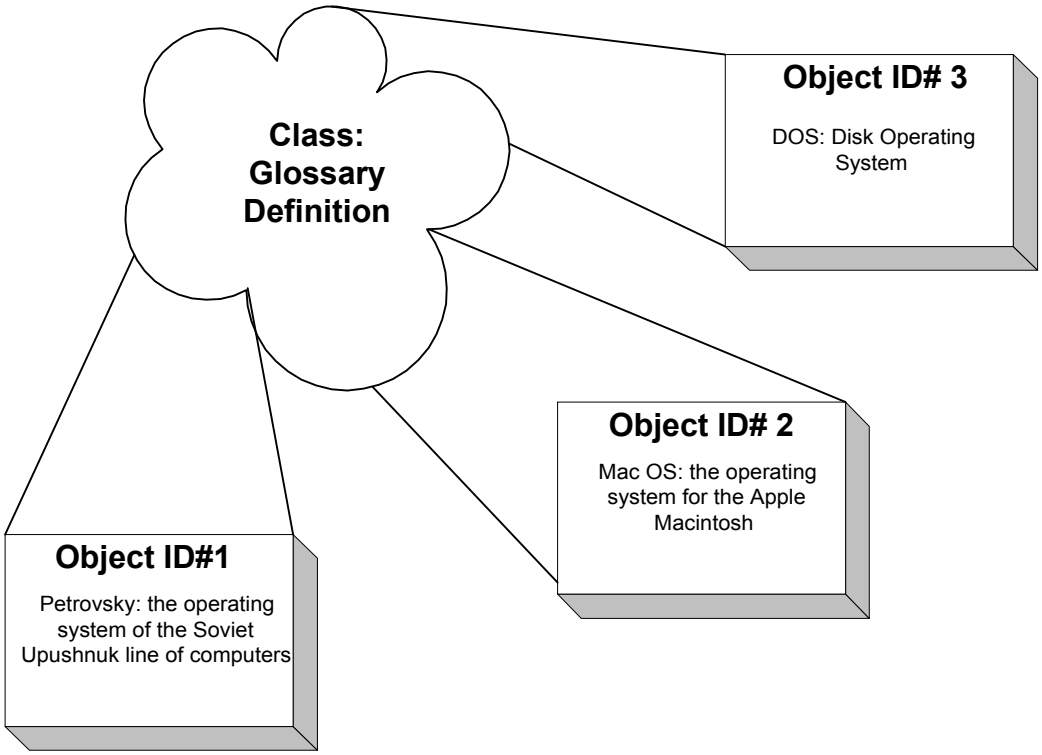
Re-use goes more easily with objects because

- We can use exactly the same object in many locations without rewriting it.
- We can search for a set of objects by class name (Give me all the procedures in this chapter).
- We can refine the search by defining attributes as well (Give me all procedures in which the Subject attribute contains F-16 and flaps, and the date is after 1997).
- We can take apart an existing object and use selected components, but not others, in a new object.





Each rhetorical object has a unique identity, or content.



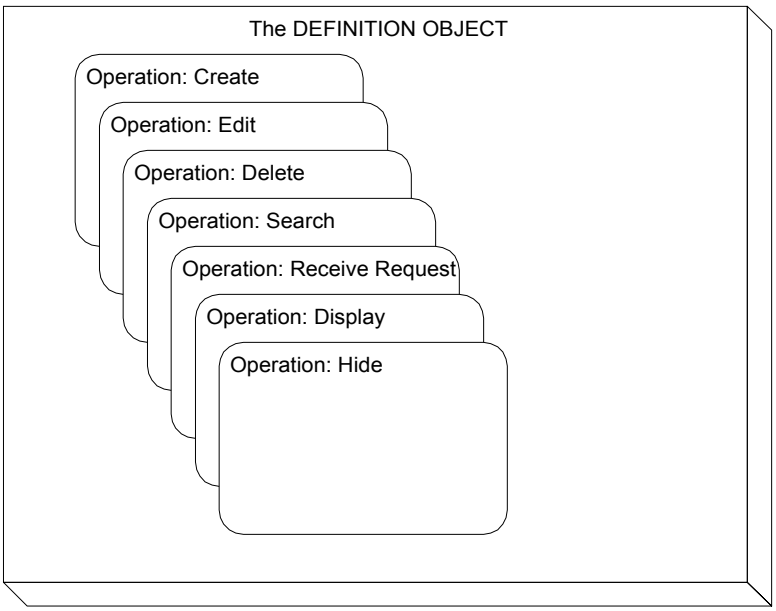
Each object has **its own data**:

- The actual words in a step
- The image in the figure
- The video clip

So even though two objects may be members of the same class, you cannot swap one for the other without destroying sequence and meaning.

Each object usually has **its own unique identifier**, as well, like a serial number, used by the software to distinguish one object from another.

A rhetorical object may perform operations on its own data.



An operation is a procedure or activity that can be performed on the data within the object, following a **method** defined in the class or superclass to which the object belongs.

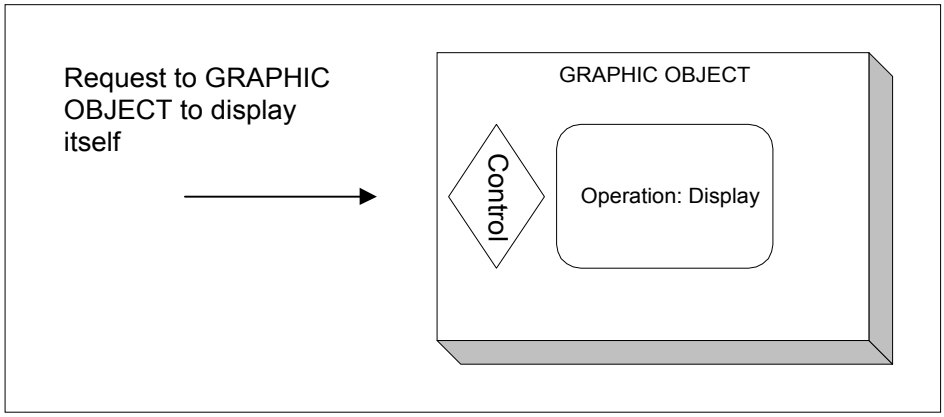
An object may be able to performs several operations.

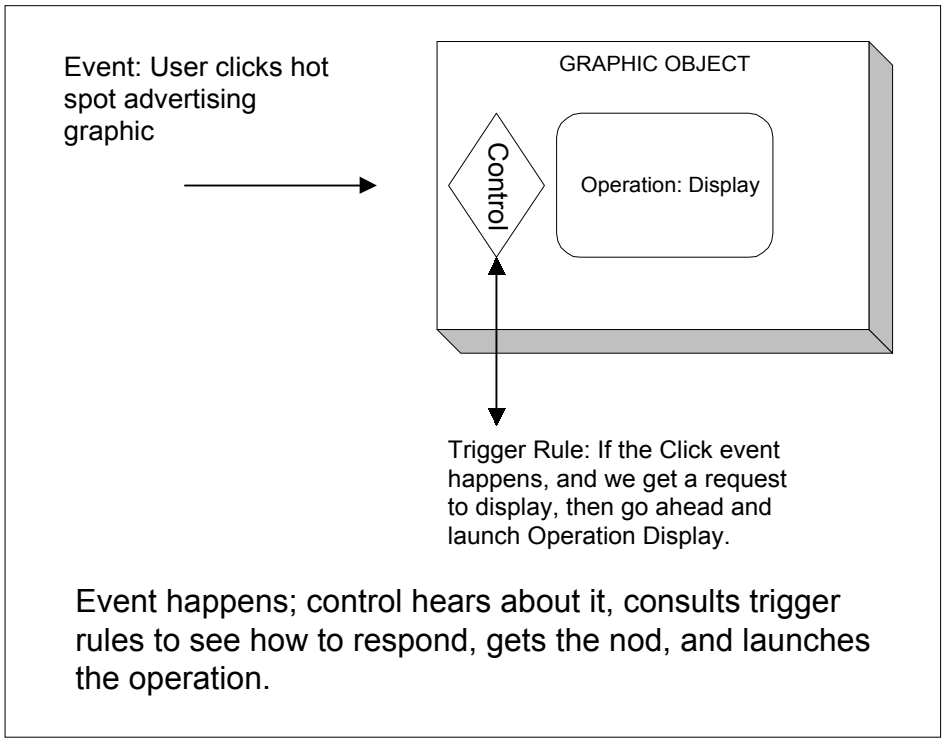
Scenario: A request arrives at the object's door, asking it to perform an operation.

1. The request is examined by a control to see if it is legitimate.
2. If so, the control consults guidelines (called trigger rules) to decide which operation to launch. A trigger rules responds to a specific request or message, and triggers a specific operation.
3. The control launches the operation.

With rhetorical objects, the operations tend to be rhetorical: appear, disappear, in different formats, interface objects, locations.

In object software, an object, then, encapsulates data and behavior. Nothing else can manipulate its own data.





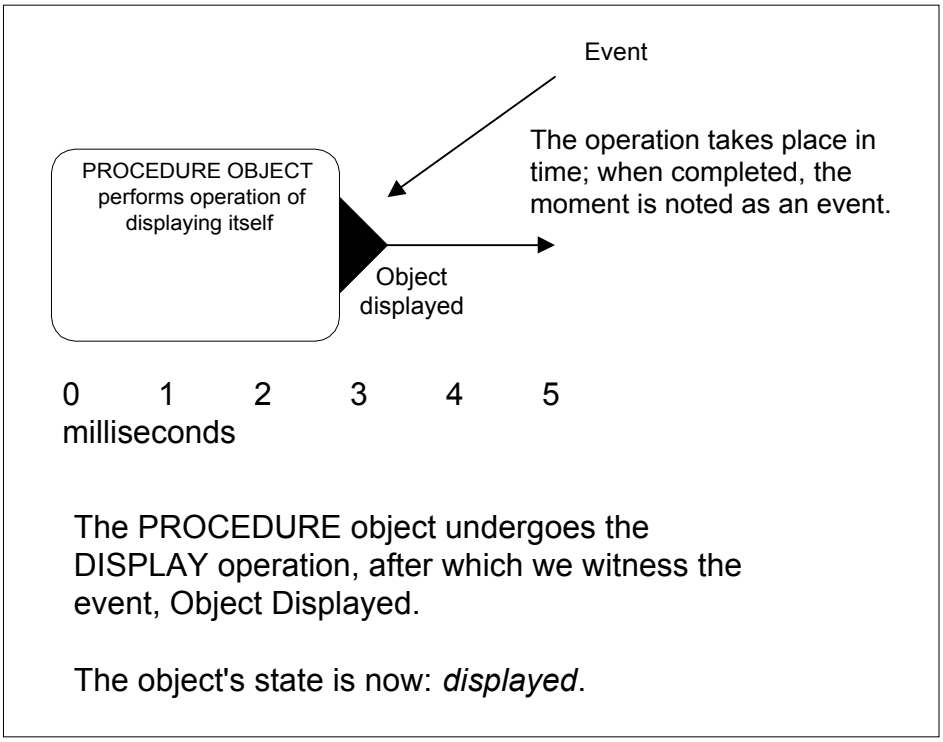
Event happens; control hears about it, consults trigger rules to see how to respond, gets the nod, and launches the operation.

Dan Ingalls, one of the creators of Smalltalk

The old way involves bit-grinding processors raping and plundering data structures.

New way: a universe of well-behaved objects that courteously ask each other to carry out their various desires.

A rhetorical object changes states.



After an object completes an operation, its state has changed.

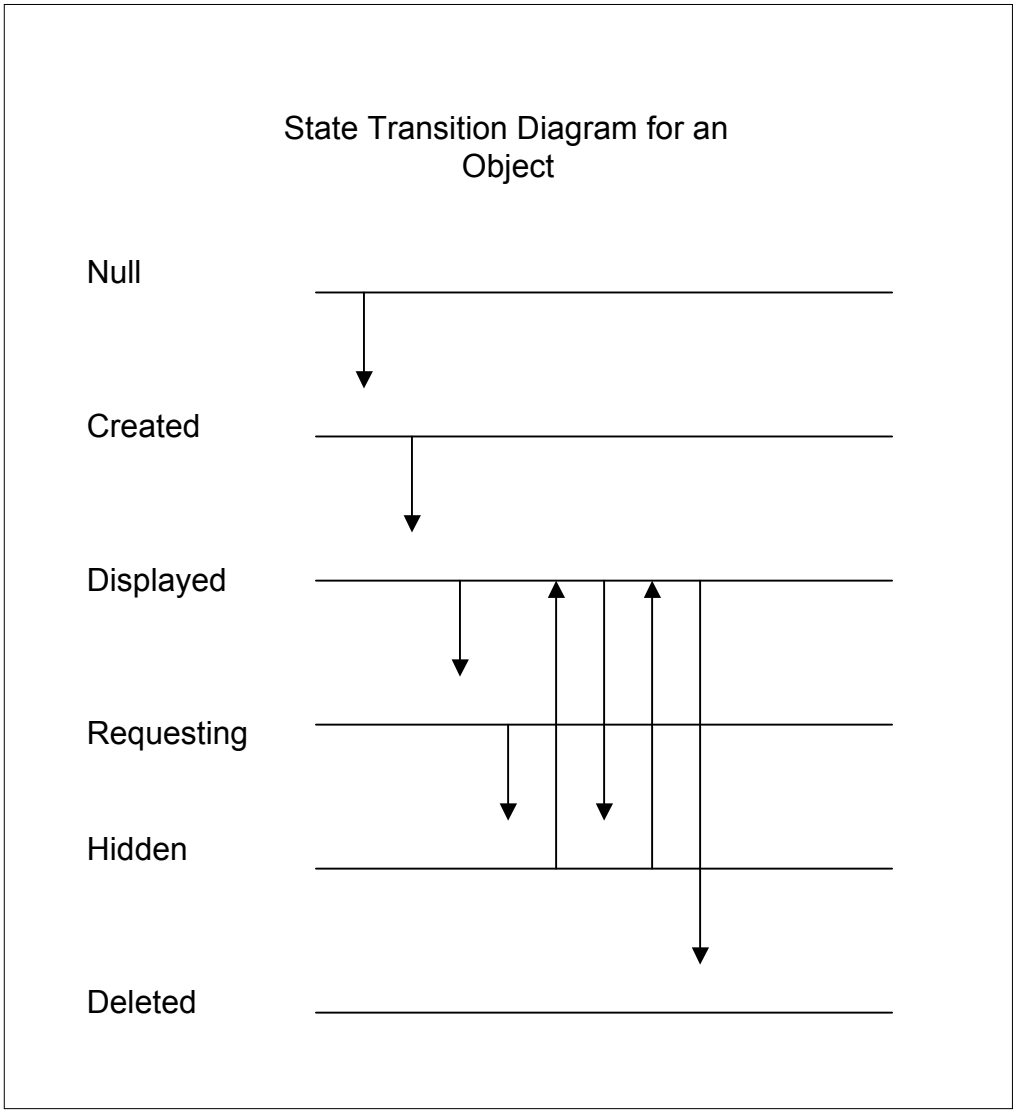
The moment of change is sometimes marked as an **event**.

There are many kinds of events:

- A new object is created, as an instance of some class.
- The object performs some operation.
- The object is deleted.
- The value of an attribute is changed.

One event may trigger further operations in this object or others.

Most objects go through a standard lifecycle, or succession of events. As the object moves from one event to another, it changes state, as analyzed in a state transition diagram.



Just as each object has a limited repertoire of operations it can perform, each object has a **limited number of state changes** that are valid.

A state is usually **an observable condition** that the object can be in. (A state is not itself an action, although a state may be "Continuing to wait for instructions.")

The state of a rhetorical object can be thought of one or all of these ways (depending on implementation):

- as its current set of attributes
- as the set of classes it is now a member of
- as the current set of relationships it has with other objects
- as the result of its most recent operation

Summary

	Characteristics	Examples
Class	A rhetorical object is an instance of a class, inheriting its properties.	One particular procedure is an instance of a whole class. In the Procedure class, by definition, we expect there will be at least one object called a title, and another object called a step.
Respons-ibility	A rhetorical object has a responsibility, or purpose, as a member of that class.	All procedures have the responsibility of answering a question that begins, "How do I?"
Identity	A rhetorical object has a unique identity, as an individual thing.	This procedure is different from all other procedures. It tells its own story.
Attributes	A rhetorical object has attributes.	Owner, date of creation or modification, the data itself.
Relation-ships	An object has a variety of relationships with other objects, particularly through composition. • Acts as a component in a larger object • Contains smaller objects as its own components, organized according to certain rules.	The procedure contains other objects, such as a title object, introduction, various steps, and several optional explanations and illustrations. The procedure belongs within a larger object known as a Procedure Group.
Messages	An object sends messages to other objects, and receives messages from them.	Within the procedure, a boldfaced word sends a message, when clicked, asking the term's definition to pop up in its own window.
Operations	An object performs certain operations on its data.	When asked to do so by a message from a menu item, the procedure displays itself

	An object encapsulates its information, refusing to allow outside agents to manipulate its own data directly.	onscreen. When asked to do so by a message from an editing program, the procedure accepts cuts, edits, and formats. You have to ask it to use its own limited set of operations; no other operations will be accepted. The procedure allows an editing program to remove parts of the text, or to delete the whole procedure, but does not allow the text to be replaced by hexadecimal notation.
States	An object changes states.	The procedure may be hidden, waiting to be displayed, or it may be displayed.
Reuse	An object can be reused, in a new design.	The same procedure can be used in a manual about this particular product, and in another manual that deals with this product and several others, in a bundle.

Objects vary enormously in scale, from a hairline to a suite of books.

Objects seem to be creatures with minds of their own; they appear to act, do something,send messages, receive, react.