Streamlined Object Modeling Principles

A

Object Modeling

PRINCIPLE 1
THE HOW OF WHY IS WHAT

Conceptual planning for an endeavor goes through three stages: why build, what to build, and how to build. How needs what to define its scope, and what needs why to define its purpose.

PRINCIPLE 2
THE OBJECT MODELING VIEWPOINT

Use object modeling to build group consensus by focusing on impersonal objects, not subjective users.

PRINCIPLE 3
OBJECT MODELING A NEW BUSINESS

Use object modeling with clients building a new business to flesh out details and issues, and document the proposed business in an impersonal and concrete manner.

PRINCIPLE 4
OBJECT MODELING FOR PROCESS RE-ENGINEERING

Use object modeling with clients re-engineering a business process to get them out of the current way of doing things and to help them see the big picture, so they can discover a better solution.

PRINCIPLE 5
OBJECT MODELING BEFORE USE CASES

For understanding a complex business process, use object modeling to bring out what needs to happen. Consider use cases afterward to illustrate how users interact with the objects in the system.

PRINCIPLE 6
MANAGING COMPLEXITY WITH OBJECT MODELING

Object modeling handles complexity by encapsulating information, rules, and behaviors within successive layers of objects. Each layer provides a stable intermediate form for building a larger system.

Finding Objects

Object Think

PRINCIPLE 7
PERSONIFY OBJECTS

Object model a domain by imagining its entities as active, knowing objects, capable of performing complex actions.

PRINCIPLE 8
GIVE OBJECTS RESPONSIBILITIES

Turn information about a real-world entity and the actions performed on it into responsibilities of the object representing the entity.

PRINCIPLE 9
OBJECT'S RESPONSIBILITIES

An object's responsibilities are: whom I know—my collaborations with others; what I do—my services; and what I know—my properties.

PRINCIPLE 10
TALK LIKE AN OBJECT

To scope an object's responsibilities, imagine yourself as the object, and adopt the first-person voice when discussing it.

Object Selection

PRINCIPLE 11
THE PEOPLE PRINCIPLE

Use an `actor` object to model individual people participating in a system. Also use an `actor` object to model an organization of people participating in a system as a single entity.

PRINCIPLE 12
THE CONTEXT PRINCIPLE

A context of participation exists whenever a person or organization undertakes actions that are tracked and recorded. Actions that require different permissions or information from the person or organization belong in different contexts.

PRINCIPLE 13
THE ROLE PRINCIPLE

For each context an entity participates in, create a separate `role` object. Put the information and permissions needed for that context into the `role`.

PRINCIPLE 14
THE PLACE PRINCIPLE

Model a location where recorded actions occur with a `place` object. Model a hierarchical location with an `outer place` containing a `place`. Model the uses of a `place` or `outer place` in different contexts with `role` objects.

PRINCIPLE 15
THE THING PRINCIPLE

Model a thing with two objects: an `item` that acts as a description defining a set containing similar things, and a `specific item` that distinguishes a particular thing from others in the set. Model the uses of a thing in different contexts with `role` objects.

PRINCIPLE 16
THE AGGREGATE THING PRINCIPLE

Model a receptacle of things as a `container` with `content` objects. Model a classification of things as a `group` with `member` objects. Model an ensemble of things with an `assembly` of `part` objects.

PRINCIPLE 17
THE EVENT PRINCIPLE

Model the event of people interacting at a place with a thing as a `transaction` object. Model a point-in-time interaction as a `transaction` with a single timestamp; model a time-interval interaction as a `transaction` object with multiple timestamps.

PRINCIPLE 18
THE HISTORY PRINCIPLE

To record historical or time-sensitive information about a person, place, or thing, use a time-interval `transaction`.

PRINCIPLE 19
THE COMPOSITE EVENT PRINCIPLE

Model people interacting at a place with multiple things as a `composite transaction`; for each thing involved, include a `line item` to capture specific interaction details.

PRINCIPLE 20
THE FOLLOW-UP EVENT PRINCIPLE

Model an event that follows and depends on a previous event with a `follow-up transaction`.

Collaboration Rules

Distributing Rule Checking

PRINCIPLE 21

MOST SPECIFIC, LOCAL, OR DETAILED OWNS THE RULE

For collaboration patterns involving people, places, and things, put the collaboration rules in the most specific, local, or detailed pattern players.

PRINCIPLE 22

INTERACTING ENTITY OWNS THE RULE

For collaboration patterns involving people, places, and things interacting with an event, each interacting pattern player that represents an entity owns its collaboration rules.

Services and Properties

Object Think Processes

PRINCIPLE 23

BE OBJECTIVE WITH PROCESSES

Be objective when asking about processes. Talk instead about the objects—people, places, things, and events—involved in the process and the actions on these objects, rather than asking clients how they "want to do it."

PRINCIPLE 24

DO IT MYSELF

Objects that are acted upon by others in the real world do the work themselves in the object world.

PRINCIPLE 25

DO IT WITH DATA

Objects encapsulate data representing an entity together with the services that act on it.

Distributing the Work

PRINCIPLE 26
DIRECTOR PRINCIPLE

Real-world actions on entities map to one of the objects representing that entity. This object is called the director of the action because it directs itself and its collaborators in carrying out the action.

PRINCIPLE 27
MOST SPECIFIC DIRECTS

When a real-world action maps to two collaborators representing a single entity or an aggregation of entities, the director is the most specific, local, or detailed pattern player.

PRINCIPLE 28
EVENTS DIRECT THE WORK

When an action requires cooperation among the collaborating entities of an event, the event directs the action.

Types of Services

PRINCIPLE 29
LET THE DIRECTOR CONDUCT

Use the "specific directs" and "event directs" principles to find the director of a process. Assign the director a conduct business service to initiate the process.

PRINCIPLE 30
MOST KNOWLEDGEABLE IS RESPONSIBLE

When a `role` acts on a `specific item` at a given place and the event is recorded, give the most knowledgeable or restrictive object a conduct business service that establishes the transaction.

PRINCIPLE 31
LET AN OBJECT DETERMINE MINE

Provide an object with determine mine services so it may answer requests for current information.

PRINCIPLE 32
LET AN OBJECT ASSESS EVENTS

Provide an object with analyze transactions services so it may assess its historical information, past events, and future scheduled events.

Descriptive Properties

PRINCIPLE 33
MAKE IT REAL AND RELEVANT

Descriptive properties come from an object's relevant real-world characteristics. Use domain experts, legacy databases, and information architectures to locate relevant descriptive properties.

PRINCIPLE 34
TRACK BUT DON'T KEY

Keep keys and object IDs off the diagram. Include identifying properties only if they come from the domain.

PRINCIPLE 35
HIDE REDUNDANT ACCESSORS

Assume each property listed in the object definition has a read and write accessor, but don't put them in the diagram.

PRINCIPLE 36
SHOW DERIVED ACCESSORS

Represent a derived property with a read accessor in the service section.

PRINCIPLE 37
ALWAYS DATE EVENTS

`Transaction` objects always include date and/or time properties.

PRINCIPLE 38
DATE OBJECTS WITH SPECIAL OCCURRENCES

Put date and/or time properties in non-`transaction` objects to record a non-repeatable occurrence or a repeatable occurrence that does not require history.

PRINCIPLE 39
HISTORICAL PROPERTIES NEED OBJECTS

Use history event objects to keep an audit trail of values for a property. Treat the property like a derived one; include a special accessor to read the property value for a given date.

State Properties

PRINCIPLE 40
KNOWING WHERE IN THE LIFECYCLE

In a person, place, or thing object, make the lifecycle state a property derived from event collaborators. In an event, make the lifecycle state a property, unless it is derived from follow-up events.

PRINCIPLE 41
KNOWING WHICH OPERATIONAL STATE

Put an operating state property in any person, place, or thing object that switches between different operational modes.

PRINCIPLE 42
CACHE WHEN FINAL

When an object reaches one of its final lifecycle states, consider caching its derived properties.

PRINCIPLE 43
ONLY CHANGE STATE WHEN CONDUCTING BUSINESS

Allow only conduct business services to change an object's lifecycle or operational state properties.

Complex Properties

PRINCIPLE 44
COLLAPSE CLUTTER OBJECTS

Collapse objects whose only purpose is to represent complex information into properties.

PRINCIPLE 45
CLASSIFY ROLES

Use a role classification property to distinguish different levels of participation only if the participation level requires no history and no additional properties, behaviors, or collaborations.

PRINCIPLE 46
CLASSIFY TYPES

Use a type classification property to distinguish different object types only if the type requires no history and has no additional properties, behaviors, or collaborations.

Object Inheritance

Parent – Child Responsibilities

PRINCIPLE 47
OBJECT INHERITANCE

Use object inheritance between two objects representing a single entity or event when the entity participates in multiple contexts, when the entity comes in many variations, or when the event involves multiple interactions.

PRINCIPLE 48
PARENT RESPONSIBILITIES

In object inheritance, the parent object contains information and behaviors that are valid across multiple contexts, multiple interactions, and multiple variations of an object.

PRINCIPLE 49
CHILD RESPONSIBILITIES

In object inheritance, the child object represents the parent in a specialized context, in a particular interaction, or as a distinct variation.

PRINCIPLE 50
CHILD ASSUMES THE PARENT'S PROFILE

In object inheritance, the child object assumes its parent's profile, enabling it to answer read-only requests for information about properties and collaborators of the parent.

Object Inheritance vs. Class Inheritance

PRINCIPLE 51
OBJECTS NOT CLASSES

Object inheritance relates two objects, each representing different views of the same entity or event. Class inheritance relates two classes, one extending the structure defined in the other.

PRINCIPLE 52
REPRESENTATION VS. SPECIALIZATION

Use object inheritance to represent multiple views of an entity. Use class inheritance to specialize an existing class of objects.

PRINCIPLE 53
VALUES VS. STRUCTURE

Object inheritance is the sharing of actual property values from a parent object. Class inheritance is the sharing of the structure for holding property values from an existing class definition.

PRINCIPLE 54
DYNAMIC VS. STATIC

Object inheritance is dynamic since shared property values often change their state during the course of a parent object's lifetime. Class inheritance is static because the structure for holding property values rarely changes during a class definition's lifetime.

AppA.fm Page 325 Friday, August 24, 2001 10:57 AM

Object Inheritance of Properties

PRINCIPLE 55

VALUES THROUGH SERVICES

Use object inheritance to allow a child to share property values with its parent. Add a read accessor in the child for each property value it object inherits from its parent.

PRINCIPLE 56

READ BUT NO WRITE

Never allow a child object to change property values in its parent.

PRINCIPLE 57

ONLY PUBLIC PROPERTIES

Properties of the parent that are not publicly accessible cannot be object inherited by a child object.

PRINCIPLE 58

NO DESIGN, JUST BUSINESS

Don't allow a child to object inherit design properties that were added to the parent to improve efficiency, support persistence storage, allow interactive display, or satisfy programming practices.

PRINCIPLE 59

QUERIES NOT STATES

Don't allow a child to object inherit read accessors for state, type, or role properties. Do allow the child to object inherit related property value services, such as "isPublished," "isCancelled," "isAdmin," etc.

Object Inheritance of Collaborations

PRINCIPLE 60

IN MY PARENT'S GROUPS

Always allow a child to object inherit its parent's `group`, `assembly`, and `container` collaborations.

PRINCIPLE 61
REMEMBERING MY PARENT'S EVENTS

Always allow a child to object inherit its parent's historical and event `transactions`.

PRINCIPLE 62
FAMILY TIES

Always allow a child to object inherit its parent's parent, but do not allow a child to object inherit other child objects belonging to its parent.

PRINCIPLE 63
SHARE AND SHARE ALIKE

Allow a child to object inherit `follow-up transactions` for its parent's events if and only if the `follow-up transactions` are valid for all the parent's children.

PRINCIPLE 64
MY PARENT THE EVENT

Allow a line `item child` to object inherit the `role` and `place` collaborations of its `composite transaction` parent.

Object Inheritance of Services

PRINCIPLE 65
DETERMINE MINE, TOO

A determine mine service of a parent is object inheritable if every child object could be asked the question the determine mine service answers.

PRINCIPLE 66
ANALYZE ONLY WHAT YOU KNOW

An analyze transactions service of a parent is object inheritable if the child object can object inherit the `transactions` being analyzed.

PRINCIPLE 67

CHILDREN CANNOT CONDUCT BUSINESS

A conduct business service of a parent is never object inheritable because the child cannot alter the parent or the context of the parent.

Child vs. Strategy Objects

PRINCIPLE 68

IT'S A CHILD NOT A FUNCTION

Use internal, stateless Strategy objects to encapsulate pluggable functionality for Context objects. Use external, stateful child objects to model another view of parent objects.

Implementing Collaboration Pairs

Object Definition Interfaces

PRINCIPLE 69

SHOWING YOUR PROFILE EVERYWHERE

To implement object inheritance, describe the parent's object inheritable services with a profile interface, and require all child objects to exhibit the profile interface.

PRINCIPLE 70

CONDUCT BUSINESS INTERFACES

A conduct business interface includes all the business services of an object, either directly or by extending the object's profile interface.

PRINCIPLE 71

HOW I SEE YOU

Collaborators refer to one another using their conduct business interfaces.

PRINCIPLE 72
MAKE THE CHILDREN PARENT-READY

To allow future system growth, define profile interfaces for child objects so they can later become parents.

Implementing Objects

PRINCIPLE 73
MINIMUM PARAMETER RULE

Only properties and collaborations necessary for an object to exist should be passed into the object's construction method.

PRINCIPLE 74
MOST SPECIFIC CARRIES THE LOAD

When work requires cooperation between two collaborators, encapsulate the majority of the effort within the most specific collaborator.

PRINCIPLE 75
PROPERTIES BEFORE COLLABORATIONS

Object construction methods initialize properties before establishing collaborations because collaboration rules may check property values.

PRINCIPLE 76
PART CARRIES THE LOAD

When work requires cooperation between a `whole` collaborator and a `part` collaborator, encapsulate the majority of the effort within the `part` collaborator.

PRINCIPLE 77
PUTTING PARENTS FIRST

When an object must establish two or more collaborations to be valid, parent collaborations must be established first.

AppA.fm Page 329 Friday, August 24, 2001 10:57 AM

PRINCIPLE 78
LET THE COORDINATOR DIRECT

When different types of objects are united by a single common coordinator and must work toward a common goal, allow the coordinator to direct the actions.

Implementing Business Rules

PRINCIPLE 79
WHERE RULES COME FROM

Business rules come from clients; logic rules come from good programming practices.

Implementing Property Business Rules

PRINCIPLE 80
ISOLATE PROPERTY RULES

When a property has domain-specific limits on its values, define a separate method to enforce these limits, and call this test method from within the set property accessor.

PRINCIPLE 81
ISOLATE VALUE ASSIGNMENT

Define a separate method to assign a value into the property and bypass business rule when necessary. The set property accessor calls this method after checking the business rules.

PRINCIPLE 82
DESCRIPTIVE AND TIME PROPERTY BUSINESS RULES

Descriptive and time properties are governed by business rules that define when the values can change and what ranges of values are possible.

PRINCIPLE 83
ENUMERATED PROPERTY BUSINESS RULES

Properties with enumerated types are governed by business rules that define the set of legal values and the legal transitions from one value to another.

Implementing Collaboration Rules

PRINCIPLE 84
DUAL RULE CHECKING

To achieve pluggability, extensibility, and scalability, each object must check its own collaboration rules.

PRINCIPLE 85
COMMUTATIVE RULE CHECKING

Implement collaboration rules so that either collaborator can request to be checked.

PRINCIPLE 86
ISOLATE COLLABORATION RULES

Define separate methods to enforce collaboration rules for establishing and dissolving a collaboration. Define the collaboration add and remove accessors to call the appropriate test method.

PRINCIPLE 87
ISOLATE COLLABORATION ASSIGNMENT

Define separate methods to assign and remove a reference to a collaborator and to bypass business rule checking when necessary. The collaboration add and remove accessors call the appropriate assignment method after checking business rules.

PRINCIPLE 88
STREAMLINING COLLABORATION ACCESSORS

To streamline the collaboration accessors, allow one collaborator to delegate the process of establishing and dissolving the collaboration to the other collaborator.

PRINCIPLE 89
CHOOSING YOUR DIRECTOR

To find the director of a streamlined collaboration, choose the `specific` of a `generic – specific`, choose the `part` of a `whole – part`, and choose the `transaction` of a `transaction – specific`.

Collaboration Pluggability

PRINCIPLE 90
PLUGGABLE MEANS INTERFACES

To make a collaboration pluggable, factor the essential communication requirements out of the current conduct business interfaces and into separate collaboration interfaces.

PRINCIPLE 91
ESSENTIAL CHARACTERISTICS

Extract from business requirements any properties, services, and collaboration methods that are essential across many variations of a pluggable collaborator; include these in the pluggable collaboration interface.

PRINCIPLE 92
PLUGGABILITY WITH INTEGRITY

To allow pluggability without sacrificing model integrity, design pluggable collaborations by fixing one collaborator and creating a pluggable interface for the other collaborator.

PRINCIPLE 93
SELECTING PLUGGABLE COLLABORATORS

Make pluggable the collaborator that varies the most. Lacking guidelines from the client, plug `specifics` into a `generic`; plug `specifics` into a `transaction`; plug `parts` into `containers` and `assemblies`; and allow `groups` and `members` to go either way.

Object Model Documentation

Components Description

PRINCIPLE 94
LOOK FOR OBJECT GROUPS

Group together objects that work together to achieve a shared purpose; give them a name reflecting their shared purpose using the client's vocabulary.

PRINCIPLE 95
OBJECT GROUP PROVIDING A FEATURE

If a group of objects has a shared purpose that is only a small step toward achieving a larger goal, then consider the group as providing a feature or function for a component.

PRINCIPLE 96
OBJECT GROUP AS A COMPONENT

If a group of objects has a shared purpose that accomplishes a significant goal, then consider the group as a component.

PRINCIPLE 97
OBJECT GROUP AS A SUB-COMPONENT

If a group of objects has a shared purpose that involves many steps, but is too large to be a feature and too insignificant to be a component, then consider the group as a sub-component of a larger component.

PRINCIPLE 98
DOCUMENT THESE OBJECT MODELS

An object model description is necessary when: the object model is large or has complex collaboration rules or services; the object model is being handed off to a new team for design and development; or the object model is subject to a formal review process.