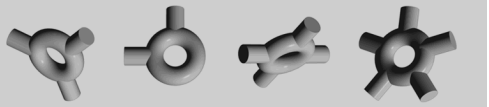
elements



# Rate Scenarios Package

TARMS Inc.

September 07, 2000

Typeset in LaTeX.

# Contents

## List of Figures



## List of Tables



## Package Description

The rate machinery in elements is designed to allow multiple ways of deriving a rate for a particular instrument. Rates may be supplied from a direct market feed or implied from other rates. A deal, position or analytic, however, simply requires *a* rate, for valuation purposes.

Rate scenarios allow the creation of consistent "views" of rates. These views allow rate clients to make requests for rates based on a logical specification of the rate; the rate is then mapped onto a derivation method.

# 1 Use Cases

## 1.1 Rate Request

A *rate request* is a request sent from a rate client to a rate manager asking for a rate corresponding to a rate specifier. Clients such as analytics calculators usually require a stream of notification events, as the rates that they use change.

## 1.2 Derivation Allocation

A rate manager needs to be able to allocate a derivation method to a logical rate specifier to form a complete rate specification.

# 2 Actors

## 2.1 Rate Client

A *rate client* is a component that needs an external source of rate information for valuation purposes. Example rate clients include position screens, analytics calculators and end-of-day.

Rate clients are, usually, only concerned with the kind of rates that they need — logical rates.

### 2.1.1 Relationships

| | Class | Description | Notes |
|---|---|---|---|
| ↔ | Rate Request §1.1 | | → |

↔:Association →:Navigable ◊:Aggregate ⧫:Composite

## 2.2 Rate Manager

A *rate manager* is a component that translates rate requests from a rate client into a concrete supply of rates.

The translation involves first associating a derivation method with a logical rate. A source of rates then has to be found and attached to; the rate manager can then supply a stream of updates to the client.

### 2.2.1 Relationships

| | Class | Description | Notes |
|---|---|---|---|
| ⇓ | End of Day Rate Manager §2.3 | | |
| ↔ | Rate Request §1.1 | | → |
| ↔ | Rate Scenario §2.4 | | |
| ↔ | Derivation Allocation §1.2 | | → |

⇓:Inherited by ↔:Association →:Navigable ◊:Aggregate ⧫:Composite

## 2.3 End of Day Rate Manager

During the end of date process, positions are usually revalued against market prices. The market prices are normally supplied as part of the end of day process and are usually fixed prices set by the middle or back office. To ensure consistency, these prices are not allowed to vary during end of day, despite market price movements.

An end of day rate manager needs to ensure that the rates are gathered from a specific end of day source and that the rates, once gathered, are fixed and not subject to further movement.

### 2.3.1 Relationships

| | Class | Description | Notes |
|---|---|---|---|
| ⇑ | Rate Manager §2.2 | | |

⇑:Inherits

## 2.4 Rate Scenario

Rates for a particular contract type can be derived in many ways. For example, a yield curve can be obtained directly from a set of interest rates, it can be implied by combining a yield curve in another currency with an FX curve, it can be built by adding a premium to another yield curve, etc.

A *rate scenario* maps a logical rate specifier — a description of the instrument and other information — onto a derivation method — the mechanism by which that rate is derived.

### 2.4.1 Relationships

| | Class | Description | Notes |
|---|---|---|---|
| ↔ | Rate Manager §2.2 | | → |
| ↔ | Derivation Allocation §1.2 | | → |

↔:Association →:Navigable ◊:Aggregate ⧫:Composite

# 3 Interfaces

## 3.1 RateScenario

A rate scenario supplies a derivation method for a logical rate.

### 3.1.1 Relationships

| | Class | Description | Notes |
|---|---|---|---|
| ⇑ | Identifiable | | |
| ↓ | StandardRateScenarioModel §5.4 | | |
| ↓ | RateScenarioReferenceData-Model §5.1 | | |
| ↔ | StandardRateScenarioModel §5.4 | parents 0..n | |
| ↔ | RateScenarioReferenceData-Model §5.1 | model 0..1 | |

⇑:Inherits ↓:Realized by ↔:Association →:Navigable ◊:Aggregate ♦:Composite

### 3.1.2 Operations

**RateDerivationSpecifier derivationMethod(RateFunctionSpecifier logicalRate)** derivation-Method

**logicalRate: RateFunctionSpecifier** The logical rate specifier that needs to be allocated a derivation method.

**Raises:** RateScenarioException

The derivation method for a logical rate. Return a derivation method for a logical rate, if there is one.

Otherwise, if this scenario is not complete, return nil, to indicate that there is no derivation method. If this scenario is complete, then raise a RateScenarioException §6.2.

**Boolean isComplete()** isComplete

Return true if this rate scenario attempts to supply derivation methods for all possible rate requests; otherwise return false.

## 3.2 RateScenarioRule

A rate scenario rule provides a potential mapping from a logical rate specifier onto

a derivation method. If a supplied RateFunctionSpecifier matches the rule, then a derivation method is constructed and returned.

### 3.2.1 Relationships

| | Class | Description | Notes |
|---|---|---|---|
| ↓ | RateScenarioRuleMatching-Model §5.2 | | |
| ↔ | StandardRateScenarioModel §5.4 | rules 0..n | |

↓:Realized by ↔:Association →:Navigable ◊:Aggregate ♦:Composite

### 3.2.2 Operations

**Boolean matches(RateFunctionSpecifier specifier, LogicalRateParameterD deriver)** matches
**specifier: RateFunctionSpecifier** The logical rate specifier to test.
**deriver: LogicalRateParameterD** The parameter deriver to use when reducing constraints.

Test to see if the supplied specifier triggers the rule. Return true if the argument specifier matches the rule, false otherwise, using the supplied deriver to reduce any constraints.

The behavior of this operation is class-specific, but generally involves making a partial match of the parameters supplied by the specifier against an internal template.

**RateDerivationSpecifier derivationMethod(RateFunctionSpecifier specifier, LogicalRateParameterD deriver)** derivation-Method
**specifier: RateFunctionSpecifier** The logical rate specifier to test.
**deriver: LogicalRateParameterD** The parameter deriver to use when reducing constraints.
**Raises:** RateScenarioException

Return the derivation method that this rule provides for the supplied specifier and deriver. Raise a RateScenarioException if a deriver cannot be built.

**String type()** type

The type of rate that this rule matches. Return a string matching one of the strings that an instance implementing RateFunctionSpecifier would return for the type() operation.

# 4 Architectural Service Interfaces

## 4.1 RateManager

A rate manger provides an interface to the rate management and caching machinery. Rate managers have the following functions:

- Allow clients to access rates for calculations.
- Allow clients to register for a stream of update notifications for a particular rate, so that event-driven recalculation can be performed.
- Allow clients to access/register for a logical rate, with the derivation method being supplied by the rate manager.
- Derive implied rates from more basic rates. Consistency between derived rates and supplied rates needs to be maintained.
- Connect to a source or sources of basic rates, rate constructors or other rate managers. A networkof rate managers is possible, with a rate manager acting as a client to another rate manager.

  When deriving implied rates, a rate manager may act as a client to itself. In this case, the rate manager will need to ensure that loops within the derivation mechanism are prevented.
- Ensure atomic updates of rate information, with source and derived rates being updated as an atomic action.

A *fixed rate manager* always returns the same value for a rate, once a rate has been obtained. Fixed rate managers allow consistent behavior for such processes as end of day.

### 4.1.1 Relationships

| | Class | Description | Notes |
|---|---|---|---|
| ⇓ | RateScenarioManager §4.2 | | |

⇓:Inherited by

### 4.1.2 Operations

**Rate rate(RateFunctionSpecifier specifier)** rate
**specifier: RateFunctionSpecifier**
**Raises:** RateManagerException, RateScenarioException

Get a rate from the rate manager. The rate specifier is a full rate specifier, with a derivation method. If the rate manager can supply this rate immediately, then this rate is returned.

If no suitable rate exists, then return nil.

If this rate manager returns true to isFixed(), the first non-nil value returned for a specific specifier fixes this rate to that value; any further query will always return the same rate.

**request(RateFunctionSpecifier specifier, Object requester, Dictionary<String, Object> options)** request
**specifier: RateFunctionSpecifier** The rate requested.
**requester: Object** The object that is making this request.
**options: Dictionary<String, Object>** Any options for this request.
**Raises:** RateManagerException, RateScenarioException

Request a stream of updates for a rate.

The rate specifier supplied gives the rate to acquire. If the rate cannot be supplied — possibly because it has an incomplete specification — then a RateManagerException is raised.

The requester is assumed to have some form of architecture-specific event channel or notification mechanism that the rate manager can use to inform the requester of an update.

The options dictionary is a dictionary of options for such things as update frequency, specialized source requests and priorities.

**derequest(RateFunctionSpecifier specifier, Object requester)** derequest
**specifier: RateFunctionSpecifier** The rate requested.
**requester: Object** The object that is making this request.
**Raises:** RateManagerException, RateScenarioException

De-request a stream of updates for a rate. This operation is the inverse of the request operation, shutting down an update stream.

**Boolean isFixed()** isFixed

Is this a fixing rate manager? Return true if this rate manager fixes a rate to its first value, false otherwise.

## 4.2 RateScenarioManager

A rate scenario manager is a rate manager that can use a RateScenario §3.1 to provide derivation methods for logical rate specifiers.

### 4.2.1 Relationships

| | Class | Description | Notes |
|---|---|---|---|
| ⇑ | RateManager §4.1 | | |

⇑:Inherits

### 4.2.2 Operations

**RateFunctionSpecifier scenario()** scenario

Return the rate scenario that this manager uses.

**Rate rate(RateFunctionSpecifier specifier)** rate
**specifier: RateFunctionSpecifier**
**Raises:** RateManagerException, RateScenarioException

Get a rate from the rate manager. The rate specifier may be a RateFunctionSpecifier, in which case the rate with the derivation method given by the rate scenario is used to build a complete RateFunctionSpecifier. Otherwise, this operation inherits its behavior from its super-interface.

**request(RateFunctionSpecifier specifier, Object requester, Dictionary<String, Object> options)** request
**specifier: RateFunctionSpecifier** The rate requested.
**requester: Object** The object that is making this request.
**options: Dictionary<String, Object>** Any options for this request.
**Raises:** RateManagerException, RateScenarioException

Request a stream of updates for a rate. The rate specifier may be a RateFunctionSpecifier, in which case the rate with the derivation method given by the rate scenario is used to build a complete RateFunctionSpecifier. Otherwise, this operation inherits its behavior from its super-interface.

**derequest(RateFunctionSpecifier specifier, Object requester)** derequest
**specifier: RateFunctionSpecifier** The rate requested.
**requester: Object** The object that is making this request.

**Raises:** RateManagerException, RateScenarioException

De-request a stream of updates for a rate. The rate specifier may be a RateFunctionSpecifier, in which case the rate with the derivation method given by the rate scenario is used to build a complete RateFunctionSpecifier. Otherwise, this operation inherits its behavior from its super-interface.

# 5 Classes

## 5.1 RateScenarioReferenceDataModel

A reference data implementation of the RateScenario interface. All operations on this interface are delegated to the associated model.

### 5.1.1 Relationships

| | Class | Description | Notes |
|---|---|---|---|
| ⇑ | ReferenceDataModel | | |
| ↑ | RateScenario §3.1 | | |
| ↔ | RateScenario §3.1 | model 1..1 | → |

⇑:Inherits ↑:Realizes ↔:Association →:Navigable ◊:Aggregate ⧫:Composite

## 5.2 RateScenarioRuleMatchingModel

A rule based on the matching of parameters. This rule simply tests the type of rate and the parameters that it has against the supplied specifier. If there is a match, then a derivation method is returned.

This class is an abstract class, from which different ways of supplying a derivation method can be defined in subclasses. Rate-type specific derivation methods can be used as subclasses, with a particular derivation method being assembled from the supplied parameters.[1]

[1] For example, an FX-specific derivation method that always constructs cross-curves for non-USD currency pairs.

### 5.2.1 Relationships

| | Class | Description | Notes |
|---|---|---|---|
| ↑ | RateScenarioRule §3.2 | | |
| ⇓ | RateScenarioRuleSimpleModel §5.3 | | |
| ↔ | ActualRateParameter | matching set 0..n | → |

⇓:Inherited by ↑:Realizes ↔:Association →:Navigable ◊:Aggregate ♦:Composite

### 5.2.2 Operations

**Boolean matches(RateFunctionSpecifier specifier)** matches

**specifier: RateFunctionSpecifier** The logical rate specifier to test.

Test to see if the supplied specifier triggers the rule.

If the type of the supplied specifier does not match the type of this rule then return false.

Otherwise, form the union of the actual parameters from the supplied specifier and the associated parameters.

## 5.3 RateScenarioRuleSimpleModel

The most simple form of rule: a pre-defined derivation method is returned when this rule is matched. The type of rate being tested for is stored in an attribute.

### 5.3.1 Relationships

| | Class | Description | Notes |
|---|---|---|---|
| ⇑ | RateScenarioRuleMatching-Model §5.2 | | |
| ↔ | RateDerivationSpecifier | derivation method 1..1 | → |

⇑:Inherits ↔:Association →:Navigable ◊:Aggregate ♦:Composite

### 5.3.2 Attributes

**type: String** The type of rate that this rule specifies.

### 5.3.3 Operations

**RateDerivationSpecifier derivationMethod(RateFunctionSpecifier specifier, LogicalRateParameterD deriver)** derivation-Method

**specifier: RateFunctionSpecifier** The logical rate specifier to test.
**deriver: LogicalRateParameterD** The parameter deriver to use when reducing constraints.
**Raises:** RateScenarioException

Return the associated derivation.

## 5.4 StandardRateScenarioModel

A standard rate scenario model allocates derivation methods by attempting to match logical rate specifiers against a series of rules. The first matching rule is used to build the derivation method.

Standard rate scenario models tend to be complex. They also tend to reflect minor variations on a base-case. These conditions indicate that the use of inheritance is essential. Since groups of rates can be grouped together to form instrument-specific valuation methods (for example, an interest rate group, an FX group, etc.) multiple inheritance can also be used. Multiple inheritance tends to create difficulties whenever different super-scenarios clash; clashes are resolved by a priority mechanism.

### 5.4.1 Relationships

| | Class | Description | Notes |
|---|---|---|---|
| ↑ | RateScenario §3.1 | | |
| ↔ | RateScenario §3.1 | parents 0..n | → |
| ↔ | RateScenarioRule §3.2 | rules 0..n | → |

↑:Realizes ↔:Association →:Navigable ◊:Aggregate ⧫:Composite

### 5.4.2 Attributes

**isComplete: Boolean = false** Is this a complete rate scenario?

**identifier: String** The unique identifier for the rate scenario.

### 5.4.3 Operations

**RateDerivationSpecifier derivationMethod(RateFunctionSpecifier logicalRate)** derivation-Method
**logicalRate: RateFunctionSpecifier** The logical rate specifier that needs to be allocated a derivation method.

**Raises:** RateScenarioException

The derivation method for a logical rate.

Test each rule, in order, to find one that matches this specifier, using the associated deriver. If a matching rule is found, return the derivation method from that rule.

If no explicit rule is found, test each parent scenario until a derivation method is found and return that method.

If no method has been found and this scenario is complete, then raise a RateScenarioException §6.2. Otherwise, if no method has been found, return nil.

# 6 Exceptions

## 6.1 RateManagerException

An exception raised when a rate manager is unable to respond to a request for a rate for structural reasons.

Rate managers, normally, are often unable to respond to a request for a rate, simply because the feed required has not been established. This exception indicates that an impossible request has been made.

## 6.2 RateScenarioException

This exception is raised when a rate scenario — or rate scenario rule — fails in some unexpected way. This exception is usually raised when a supposedly complete rate scenario is unable to supply a derivation method for a rate.

# 7 Associations

Table 1: Rate Scenarios— Associations

| Association<br>Role | Class | Card. | Notes |
|---|---|---|---|
| parents | | | |
| parent | RateScenario §3.1 | 0..n | → |
| scenario | StandardRateScenarioModel §5.4 | 0..n | |
| rules | | | |

Table 1: . . . continued

| Association Role | Class | Card. | Notes |
|---|---|---|---|
| rule | RateScenarioRule §3.2 | 0..n | → |
| scenario | StandardRateScenarioModel §5.4 | 0..n | |
| model | | | |
| model | RateScenario §3.1 | 1..1 | → |
| reference data | RateScenarioReferenceData-Model §5.1 | 0..1 | |
| matching set | | | |
| parameter | ActualRateParameter | 0..n | → |
| scenario rule | RateScenarioRuleMatching-Model §5.2 | 0..n | |
| derivation method | | | |
| derivation | RateDerivationSpecifier | 1..1 | → |
| scenario rule | RateScenarioRuleSimpleModel §5.3 | 0..n | |

→:Navigable ◊:Aggregate ⧫:Composite

### 7.1 parents

**Role: parent** *Navigable* RateScenario, 0..n.
**Role: scenario** StandardRateScenarioModel, 0..n.
The parent rate scenarios for a standard rate scenario.

### 7.2 rules

**Role: rule** *Navigable* RateScenarioRule, 0..n.
**Role: scenario** StandardRateScenarioModel, 0..n.
The rules that make up the rate scenario.

### 7.3 model

**Role: model** *Navigable* RateScenario, 1..1.
**Role: reference data** RateScenarioReferenceDataModel, 0..1.
The wrapped model for this piece of reference data.

### 7.4 matching set

**Role: parameter** *Navigable* ActualRateParameter, 0..n.

**Role: scenario rule** RateScenarioRuleMatchingModel, 0..n.
The set of parameters that a logical rate must match to fire a rate scenario rule.

## 7.5 derivation method

**Role: derivation** *Navigable* RateDerivationSpecifier, 1..1.
**Role: scenario rule** RateScenarioRuleSimpleModel, 0..n.
The derivation method to use for this rate.

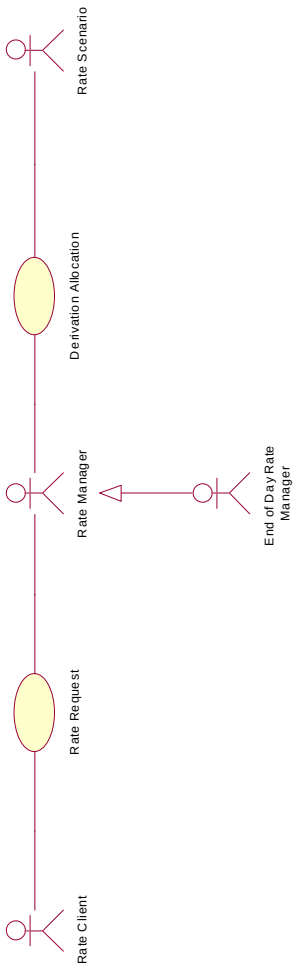



Figure 1: Class Diagram— Rate Requests

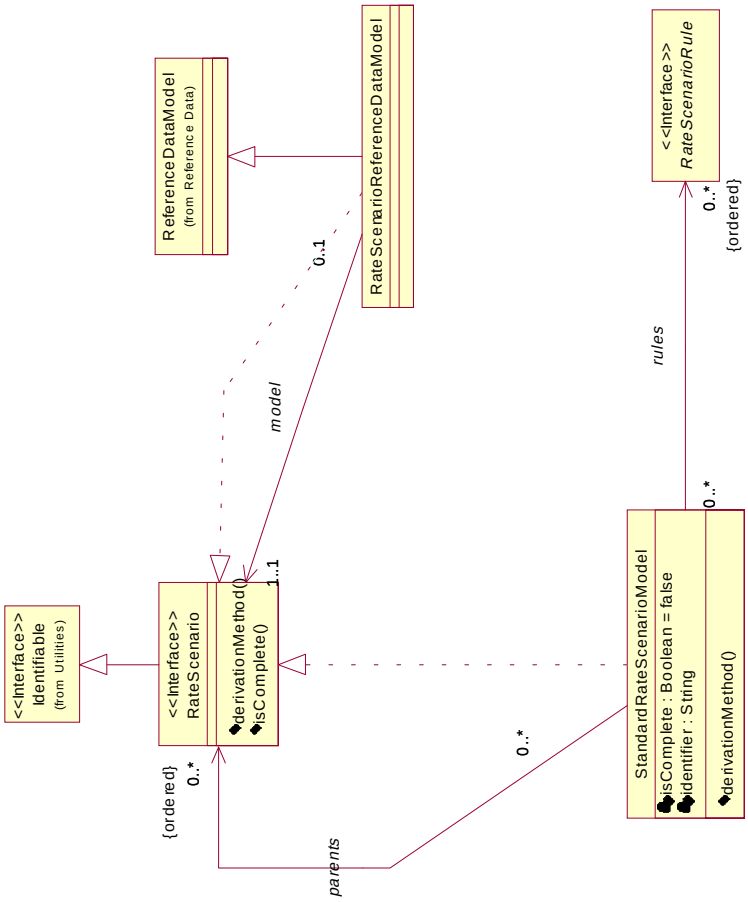



Figure 2: Class Diagram— Rate Scenarios

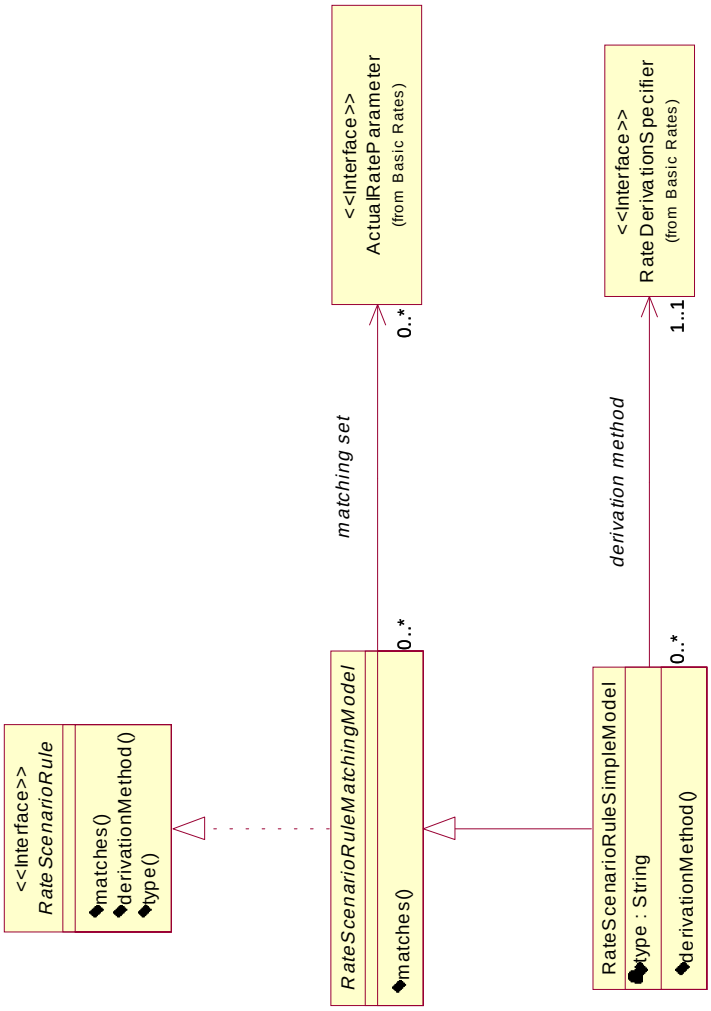



Figure 3: Class Diagram— Rate Scenario Rules

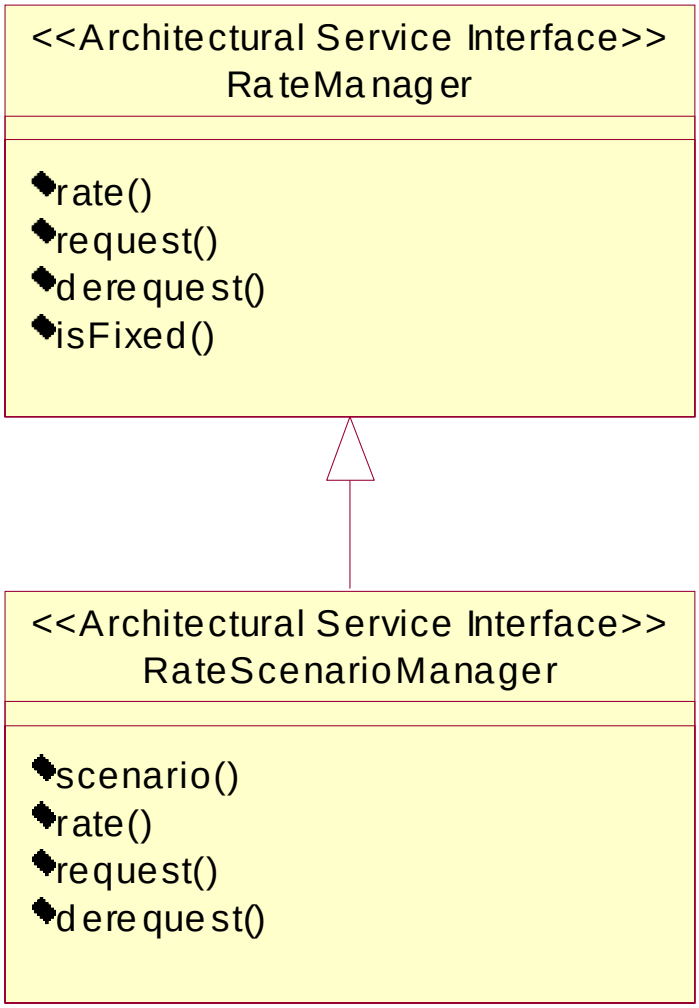



Figure 4: Class Diagram— Rate Managers

## References