

# An Introduction to UML

# The Use Case Model

by Geoffrey Sparks

*All material (c) Geoffrey Sparks 2000*

[www.sparxsystems.com.au](http://www.sparxsystems.com.au)

## Table of Contents

|  |           |
|--|-----------|
| <b>THE USE CASE MODEL .....</b>                          | <b>3</b>  |
| INTRODUCTION TO UML.....                                 | 3         |
| MODELLING THE SYSTEM FUNCTIONALITY USING USE CASES ..... | 3         |
| USE CASE NOTATION.....                                   | 3         |
| <i>The Use Case</i> .....                                | 4         |
| <i>Actors</i> .....                                      | 5         |
| <i>Constraints, Requirements and Scenarios</i> .....     | 5         |
| <i>Includes and Extends</i> .....                        | 6         |
| <i>Sequence Diagrams</i> .....                           | 7         |
| <i>Implementation Diagram</i> .....                      | 8         |
| <b>AN EXAMPLE .....</b>                                  | <b>9</b>  |
| <i>Browse Book Catalogue</i> .....                       | 9         |
| <i>Locate Book by Title or Author</i> .....              | 10        |
| <i>Request UnListed Book</i> .....                       | 11        |
| <b>RECOMMENDED READING .....</b>                         | <b>13</b> |

## **The Use Case Model**

This paper describes how to model system functionality using UML Use Cases. In the UML, Use Cases are the primary means of capturing system functionality from the user perspective, and often may replace a 'functional requirements' document.

### ***Introduction to UML***

The Unified Modelling Language (UML) is, as its name implies, a modelling language and not a method or process. UML is made up of a very specific notation and the related grammatical rules for constructing software models. UML in itself does not proscribe or advise on how to use that notation in a software development process or as part of an object-oriented design methodology.

UML supports a rich set of graphical notation elements. It describes the notation for classes, components, nodes, activities, work flow, use cases, objects, states and how to model relationships between these elements. UML also supports the notion of custom extensions through stereotyped elements.

The UML provides significant benefits to software engineers and organisations by helping to build rigorous, traceable and maintainable models, which support the full software development lifecycle.

This paper focuses on representing functional requirements in UML using Use Cases.

You can find out more about UML from the books mentioned in the suggested reading section and from the UML specification documents to be found at the Object Management Groups UML resource pages: <http://www.omg.org/technology/uml/> and at <http://www.omg.org/technology/documents/formal/>

### ***Modelling the System Functionality using Use Cases***

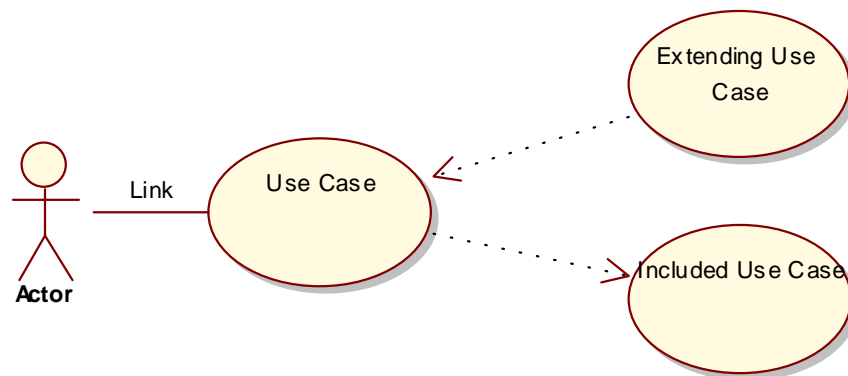
The Use Case Model describes the proposed functionality of the new system. A Use Case represents a discrete unit of interaction between a user (human or machine) and the system. A Use Case is a single unit of meaningful work; for example creating a train, modifying a train and creating orders are all Use Cases.

Each Use Case has a description which describes the functionality that will be built in the proposed system. A Use Case may 'include' another Use Case's functionality or 'extend' another Use Case with its own behaviour.

Use Cases are typically related to 'actors'. An actor is a human or machine entity that interacts with the system to perform meaningful work.

### ***Use Case Notation***

Use Case diagrams are generally made up of one or more Actors linked to one or more Use Cases, as in the diagram below.



A Use Case model describes the functionality of the system - what the system will do for the user in order to get some useful work done. It also helps to layout the actors or users of the system and their role in running the system.

## The Use Case

A Use Case is a representation of a discrete set of work performed by a user (or another system) using the operational system. It is performed as a whole or not at all and returns something of value to the user. Examples of Use Cases are AddOrder, DeleteOrder, ModifyOrder, ListTrains, ListLocations & etc.

A Use Case description will generally include:

1. General comments and notes describing the use case;
2. Requirements: Things that the use case must allow the user to do, such as <ability to update order>, <ability to modify order> & etc.
3. Constraints: Rules about what can and cant be done. Includes Pre-conditions that must be true before the use case is run - e.g. <create order> must precede <modify order>; also includes Post-conditions that must be true once the use case is run e.g. <order is modified and consistent>; Invariants: these are always true - e.g. an order must always have a customer number
4. Scenarios: Sequential descriptions of the steps taken to carry out the use case. May include multiple scenarios, to cater for exceptional circumstances and alternate processing paths;
5. Scenario diagrams: Sequence diagrams to depict the workflow - similar to (4) but graphically portrayed

## Actors

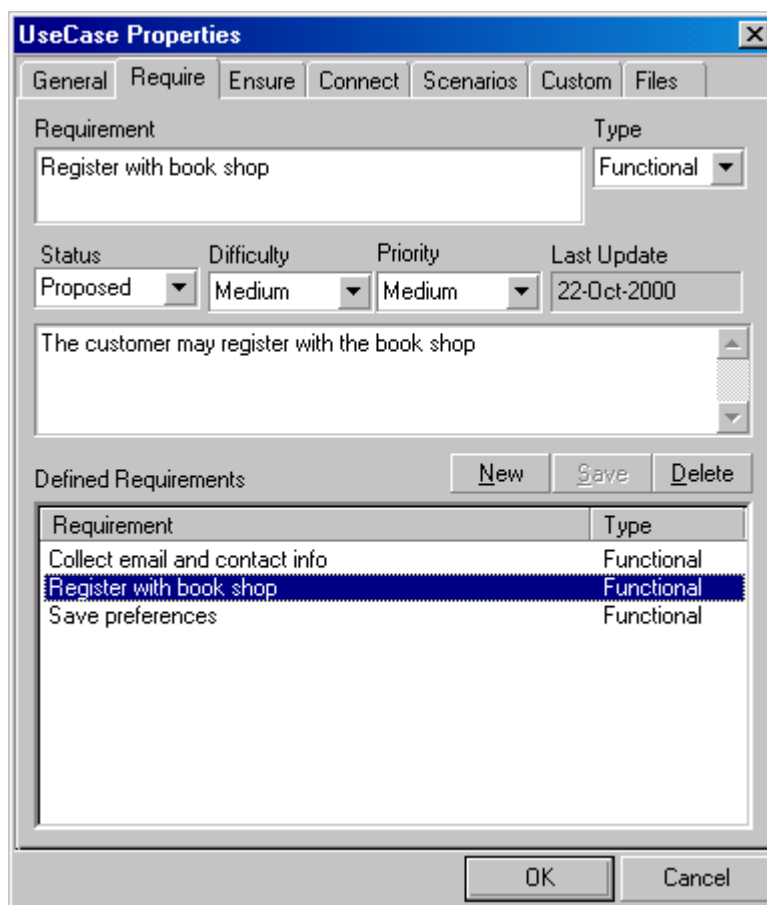


An Actor is a user of the system. This includes both human users and other computer systems. An Actor uses a Use Case to perform some piece of work which is of value to the business. The set of use cases an actor has access to defines their overall role in the system and the scope of their action.

## Constraints, Requirements and Scenarios

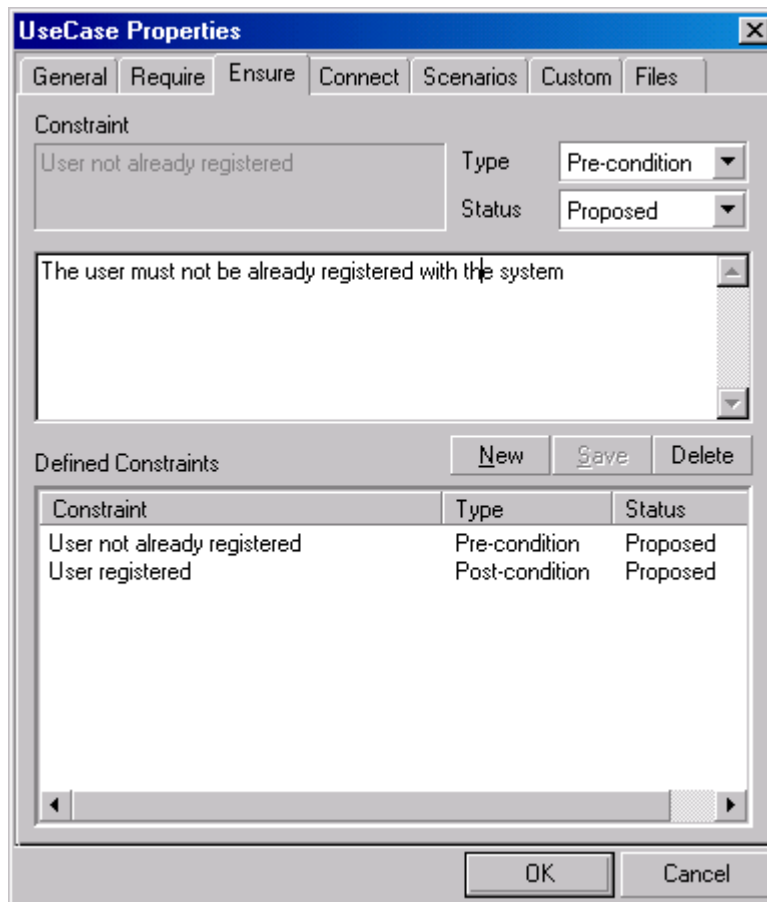
The formal specification of a Use Case includes 3 basic items:

1. Requirements. These are the formal functional requirements that a Use Case must provide to the end user. They correspond to the functional specifications found in structured methodologies. A requirement is a contract that the Use Case will perform some action or provide some value to the system.



| Requirement                    | Type       |
|--------------------------------|------------|
| Collect email and contact info | Functional |
| Register with book shop        | Functional |
| Save preferences               | Functional |

2. Constraints. These are the formal rules and limitations that a Use Case operates under, and includes pre- post- and invariant conditions. A pre-condition specifies what must have already occurred or be in place before the Use Case may start. A post-condition documents what will be true once the Use Case is complete. An invariant specifies what will be true throughout the time the Use Case operates.



Scenarios. Scenarios are formal descriptions of the flow of events that occurs during a Use Case instance. These are usually described in text and correspond to a textual representation of the Sequence Diagram.

## Includes and Extends

One Use Case may include the functionality of another as part of its normal processing. Generally, it is assumed that the included Use Case will be called every time the basic path is run. An example may be to list a set of customer orders to choose from before modifying a selected order - in this case the <list orders> Use Case may be included every time the <modify order> Use Case is run.

A Use Case may be included by one or more Use Cases, so it helps to reduce duplication of functionality by factoring out common behaviour into Use Cases that are re-used many times.

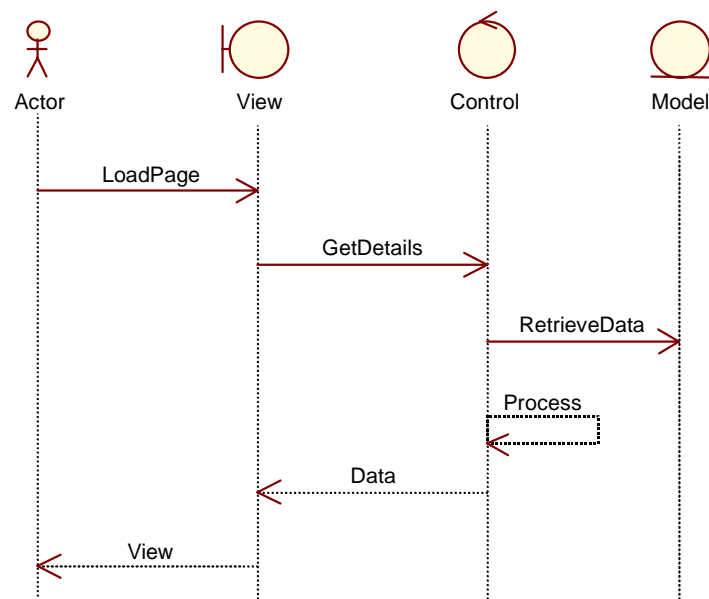
One Use Case may extend the behaviour of another - typically when exceptional circumstances are encountered. For example, if before modifying a particular type of customer order, a user must get approval from some higher authority, then the <get approval> Use Case may optionally extend the regular <modify order> Use Case.

## Sequence Diagrams

UML provides a graphical means of depicting object interactions over time in Sequence Diagrams. These typically show a user or actor, and the objects and components they interact with in the execution of a use case. One sequence diagram typically represents a single Use Case 'scenario' or flow of events.

Sequence diagrams are an excellent way to document usage scenarios and to both capture required objects early in analysis and to verify object usage later in design. Sequence diagrams show the flow of messages from one object to another, and as such correspond to the methods and events supported by a class/object.

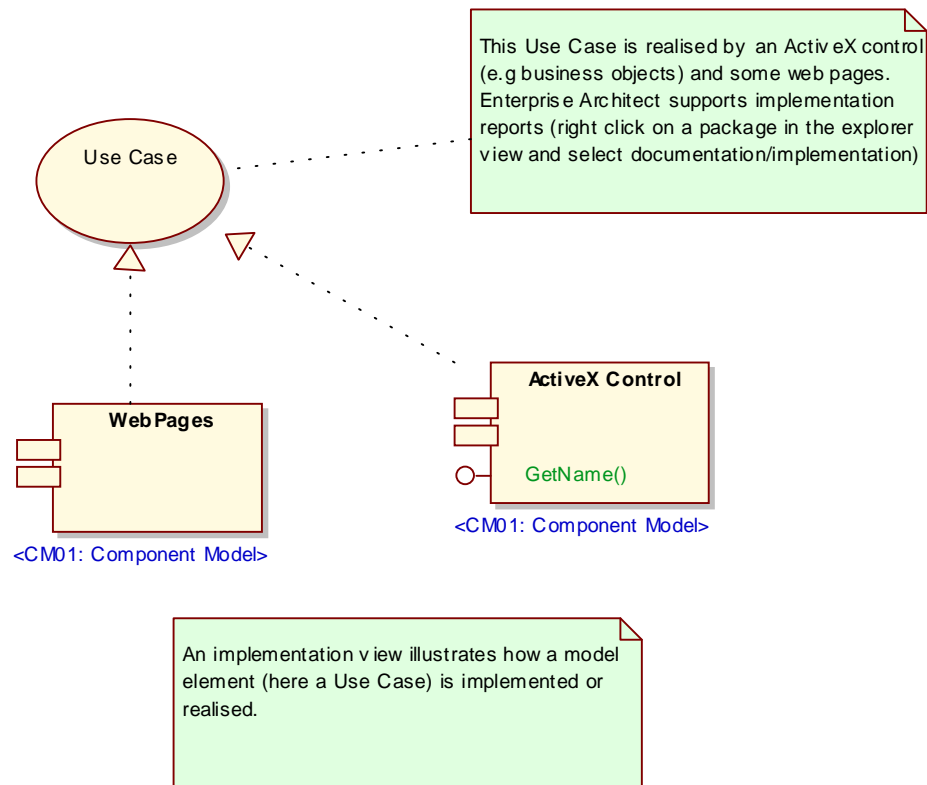
The diagram illustrated below shows an example of a sequence diagram, with the user or actor on the left initiating a flow of events and messages that correspond to the Use Case scenario.



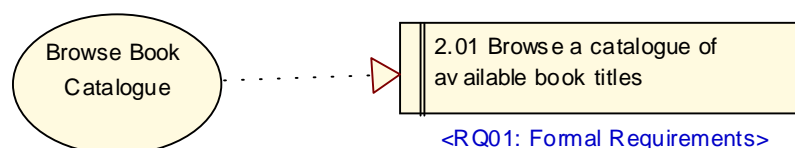
A workflow or sequence diagram may be anchored under a Use Case to further illustrate the sequence of steps taken to realise a use case. The degree of detail here depends on what it is desired to communicate - from the simple interaction of a user and the user interface to the more complicated interaction of objects and controllers within the system and with other systems.

## Implementation Diagram

A Use Case is a formal description of functionality the system will have when constructed. An implementation diagram is typically associated with a Use Case to document what design elements (eg. components and classes) will implement the Use Case functionality in the new system. This provides a high level of traceability for the system designer, the customer and the team that will actually build the system. The list of Use Cases that a component or class is linked to documents the minimum functionality that must be implemented by the component.



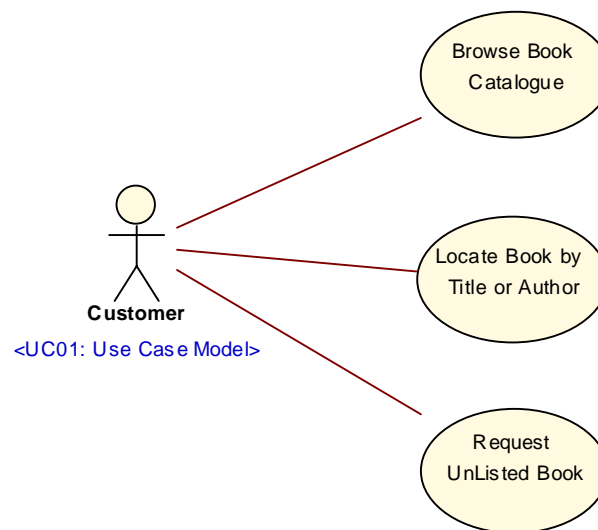
A second kind of Implementation Diagram may be built to illustrate what formal requirements a Use Case implements or realises. The example following shows that the Use Case "Browse Book Catalogue" implements the formal requirement "2.01 Browse a catalogue of available book titles". This second type of diagram is not strictly a UML construct, but may be a very important tool for documenting how the proposed system meets a client's stated specifications.





## An Example

This diagram illustrates the use cases which support searching for a book and browsing a resultant record set. The customer can enter browse criteria and scroll thru the results. If they so wish, the user may select an item for addition to their current shopping cart.



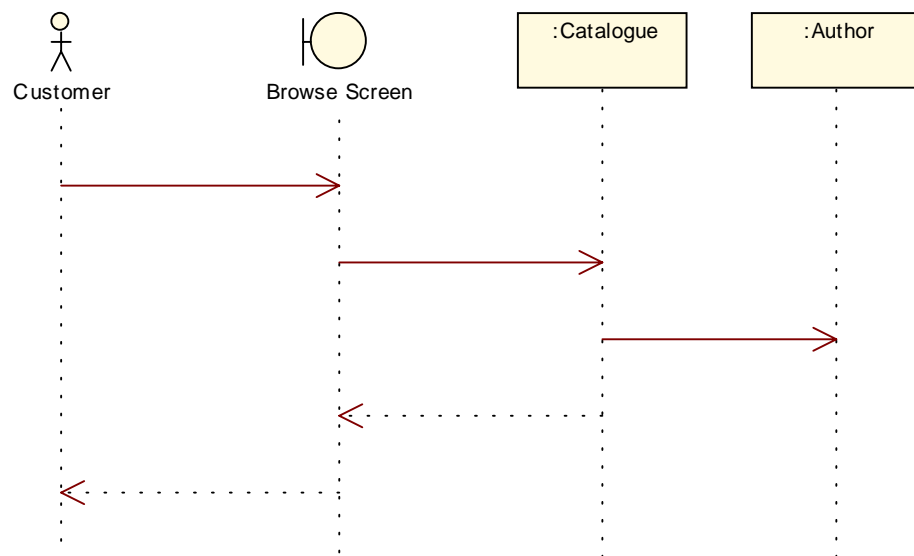
### Browse Book Catalogue

The customer can browse the current book catalogue on-line. This should detail both the book details and the stock details for books. The user should be able to filter by book title, author, book category.

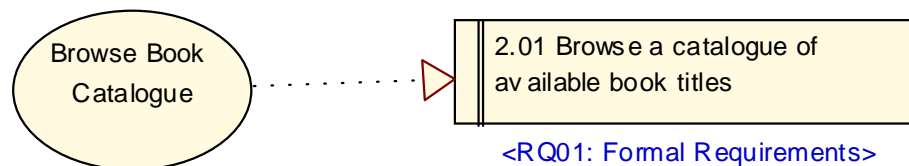
#### **Requirements**

1. *Proposed.* Display a filtered set of books.
2. *Proposed.* Scroll thru list.
3. *Proposed.* Page to next set of results.
4. *Proposed.* Add item to cart.
5. *Proposed.* Get more information on an item.

#### **Browse Catalogue Sequence Diagram**



### Browse Book Catalogue : Implementation



### Locate Book by Title or Author

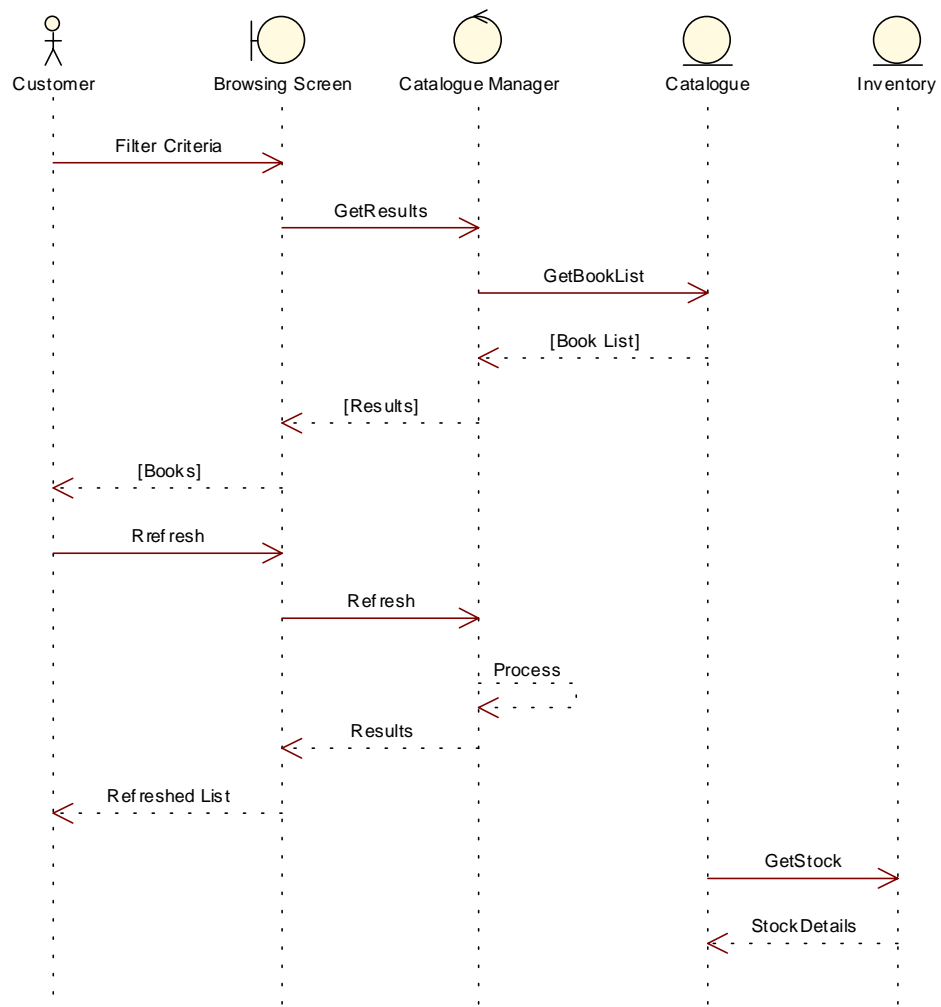
The user enters some filter criteria such as book title or author. This limits the returned set of books to those that match the supplied filter. Wild cards such as '\*' and '?' should be supported.

#### **Requirements**

1. *Proposed.* Enter search criteria for author.
2. *Proposed.* Enter book title search criteria.
3. *Proposed.* Refresh query.

### **Sequence Diagram Locate Book**

This diagram illustrates the flow of events when a user searches for a book on-line. The system will return both catalogue and inventory details for the items selected.



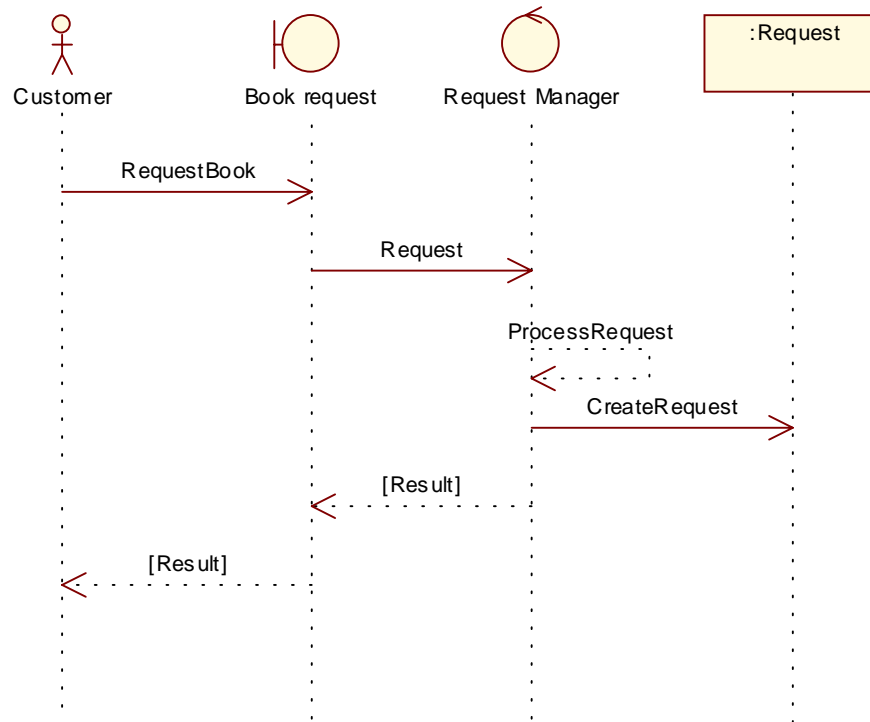
### **Request UnListed Book**

If the user cannot find a book in the current category, they should be able to place a special request. This should include details such as Author, Publisher, Title, ISBN and any other helpful information.

#### **Requirements**

1. *Proposed*. Enter a request for a book not in the catalogue.

**Request UnListed Book : Request Book**



## Recommended Reading

Sinan Si Alhir, *UML in a NutShell*.

ISBN: 1-56592-448-7. Publisher: O'Reilly & Associates, Inc

Doug Rosenberg with Kendall Scott, *Use Case Driven Object Modeling with UML*.

ISBN: 0-201-43289-7. Publisher: Addison-Wesley

Geri Scheider, Jason P. Winters, *Applying Use Cases*

ISBN: 0-201-30981-5. Publisher: Addison-Wesley

Ivar Jacobson, Martin Griss, Patrik Jonsson, *Software Reuse*

ISBN: 0-201-92476-5. Publisher: Addison-Wesley

Hans-Erik Eriksson, Magnus Penker, *Business Modeling with UML*

ISBN: 0-471-29551-5. Publisher: John Wiley & Son, Inc

Peter Herzum, Oliver Sims, *Business Component Factory*

ISBN: 0-471-32760-3 Publisher: John Wiley & Son, Inc