An Analysis Pattern for Reservation and Use of Reusable Entities

Eduardo B. Fernandez and Xiaohong Yuan
Dept. of Computer Science and Engineering,
Florida Atlantic University
Boca Raton, FL 33431
ed | xhyuan @cse.fau.edu

Abstract
This analysis pattern describes how to make a reservation for a reusable entity and its subsequent use. Its requirements are expressed by Use Cases and its description uses class models, state diagrams, and sequence diagrams. The pattern corresponds to a minimal semantic unit.

1. Introduction

We present here an analysis pattern that describes the making of a reservation for a reusable entity and its subsequent use. This pattern is in the category of what we have called semantic analysis patterns [Fer98], because they emphasize semantic aspects of the application model, as opposed to improving flexibility.
We consider this type of patterns useful to start the modeling process from the requirements. For example, identifying a few patterns in the requirements produces an initial model that can be used as a guideline for the rest of the design. These patterns can also be used to develop frameworks and components.
This pattern emphasizes fundamental aspects of reservations and leaves out extensions, exceptions, and varieties; these can be added later to define a pattern language for reservations.

2. Problem

A client (person or institution) needs to reserve a reusable entity (hotel room, vehicle, seat in a show) so that subsequently he can make use of it. The entities are limited in availability and can be grouped into equivalence classes or categories.

All of these situations imply a request for some type of entity for some specific date of use. If an entity of the requested type is available, it is booked for the client. A specific entity is assigned at the moment of use and a record of use describes the use of the entity.

The reserved entities are reusable; that is, they are not acquired by the customer, he only has the right to use them for a certain period of time. This aspect implies that the record of

use has also the role (explicit or implicit) of contract, in that it makes the user responsible for returning the entity in good condition after use.

Figure 1 shows a class diagram for a customer booking a hotel room. The association between classes Customer and Room describes each reservation made for a customer. The association class Reservation describes the information normally included in a reservation. The class Availability describes the structure of the rooms and
their occupancy.

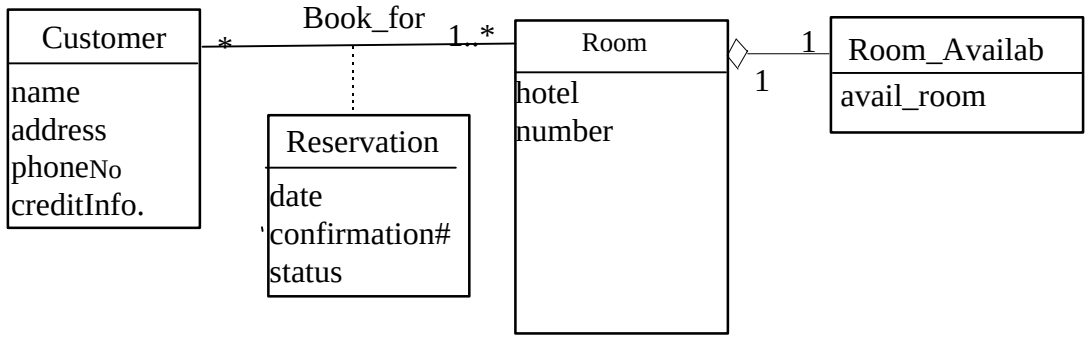

Figure 1. Booking an hotel room

3. Forces

The following forces apply here:

- The normal analysis objectives apply to the solution. An analysis model must be a faithful representation of the requirements and must not include any implementation details.
- Fundamental semantics. The pattern must describe a fundamental semantic unit. This implies simplicity and the ability to describe a variety of situations. This approach has also been used in [Ngu98], where it is called a "minimal" application.
- Model representation of real-life documents. Reservations and use records are normally recorded as paper documents as well as in their software form.

Forces for the pattern language would also include convenient and efficient client service (e.g. by including queueuing facilities), reliability (e.g., by including provisions for exceptional cases).

4. Solution

4.1 Requirements

The solution corresponds to the realization of the following Use Cases:

- **Make a reservation**. The actor is a customer (person or institution). The customer requests an entity of a certain type for given period of time starting at a given date. If available, the entity is booked. The corresponding information is recorded in a reservation document.
- **Use a reserved entity**. The actors are the same as in making a reservation. Before the entity is used, a record of use is created. When the use is completed, the entity is returned and made available again.
- **Modify a reservation**. A change is made to an existing reservation. This implies to determine availability and cancel the old reservation.
- **Cancel a reservation**. An existing reservation is canceled. Some institution policies may apply to define the consequences of the cancellation.

Note that the functionality required to modify a reservation is available in the making and canceling of reservations. Also, the exceptional flows of these Use Cases have been left out.

4.2 Class model

Figure 2 is an analysis class diagram for the realization of these Use Cases. This diagram distinguishes the type of entity being reserved (class Entity_Type), from the individual entity assigned at the moment of use (class Entity). A UseRecord class describes the use of the entity. The Availability class describes the generic idea of checking and keeping track of occupancy. The Reservation class includes the information normally kept in bookings. The association between Reservation and UseRecord indicates the fact that the use record is based on the reservation information.

4.3 Dynamic aspects

Figures 3 and 4 show the state diagrams for the most significant classes. For example, a reservation can be in Created and Confirmed states. Figure 5 shows a sequence diagram showing the making of a reservation and its later use.

5. Consequences

This pattern has the following consequences:

- It can be applied to reservation of rooms in hotels, conference rooms in a building, seats for flights, seats for shows, books and videotapes, etc. This variety of applications indicates that the pattern contains fundamental aspects of this problem. It is also a simple pattern, relatively easy to understand.

- Every aspect of the Use Cases is represented in this pattern, and no implementation aspects are included.

- The class model includes explicitly classes for Reservation and UseRecord, the two basic documents used in these systems. This makes it easy to keep history of reservations and use of the corresponding entities. The role of the Use Record as a legal contract is also explicit.

However, not all the situations described by this pattern are exactly alike:

- One gets a physical ticket when reserving a place in a show or in a flight but not when reserving a vehicle or a room (although one may get a confirmation letter). The ticket represents a contract.
- Reservation of cars are explicitly converted to contracts when the vehicles are picked up, while hotel reservations implicitly become contracts when the guest checks in.
- In the reservation of cars and rooms the specific entity is assigned at use time, while seat reservations for a show or a flight are assigned at reservation time.
- Some entities, e.g., cars, rooms, are reserved for a period of time, expressed as a range of dates, while others, e.g., seat in a flight, are for a specific event.
- A contract may be created without a reservation and a reservation may not become a Use Record.

Also, a good number of aspects are not represented:

- Description of contextual and environmental aspects of the entity. For example, in a flight seat there is the airplane, the airports involved, the number of stops, etc.
- How to keep track of availability of entities.
- How to queue up requests when the entities are not available
- How to deal with preferred customers
- Billing aspects
- History

Finally, exceptional flows were also ignored, e.g., no shows, overbooking, etc.
All these aspects can be considered by defining a pattern language or a family of patterns. We leave that for future work.

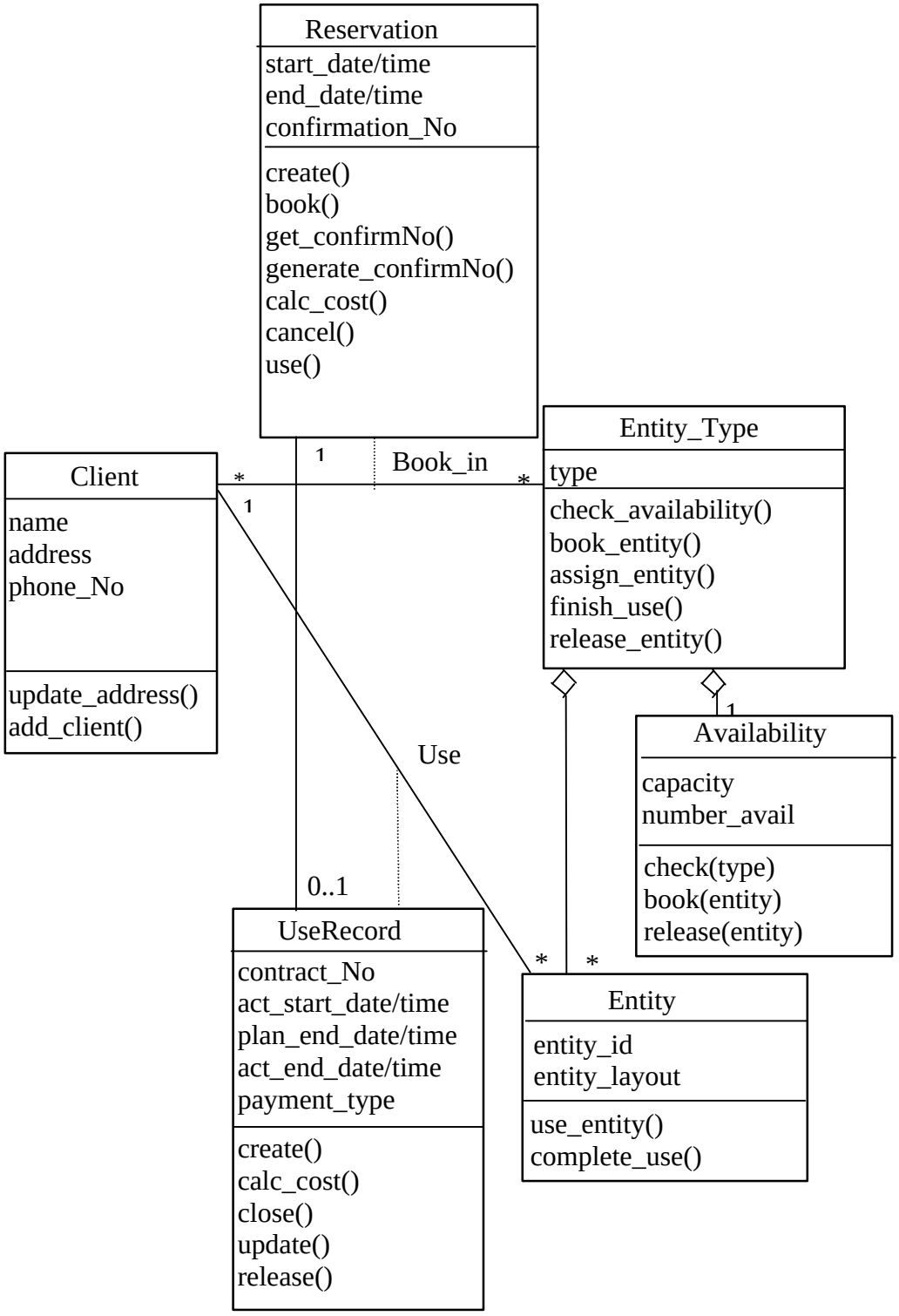

Figure 2. Class Diagram for Reservation/Use Pattern

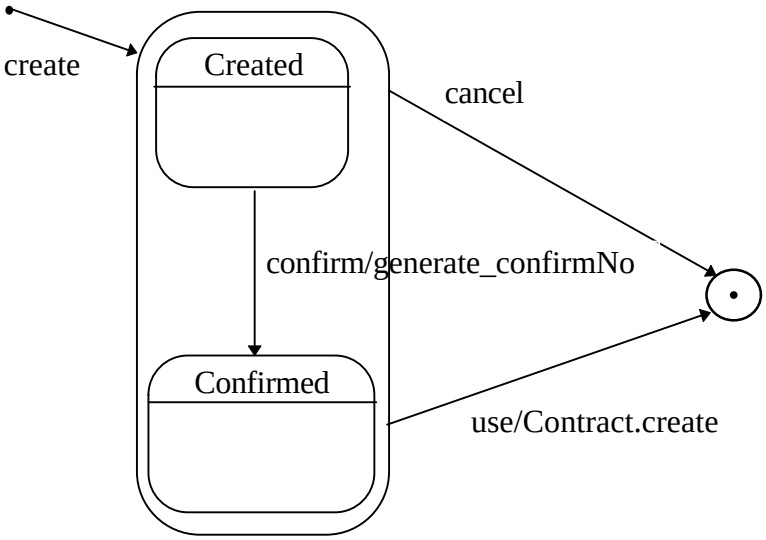

Figure 3. Statechart for class Reservation

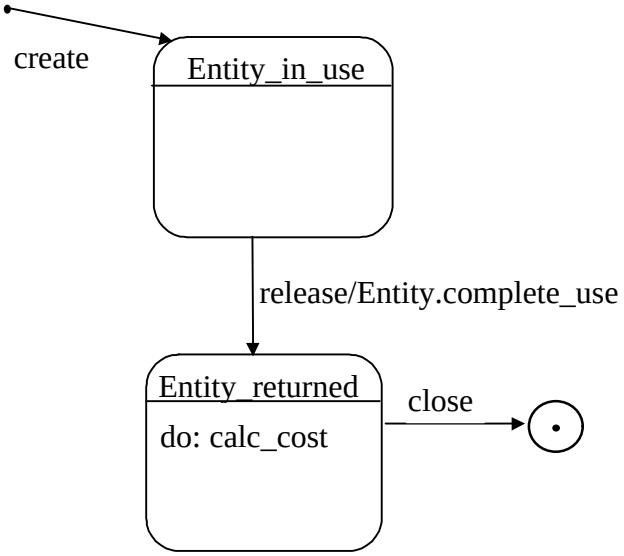

Figure 4. Statechart for class UseRecord

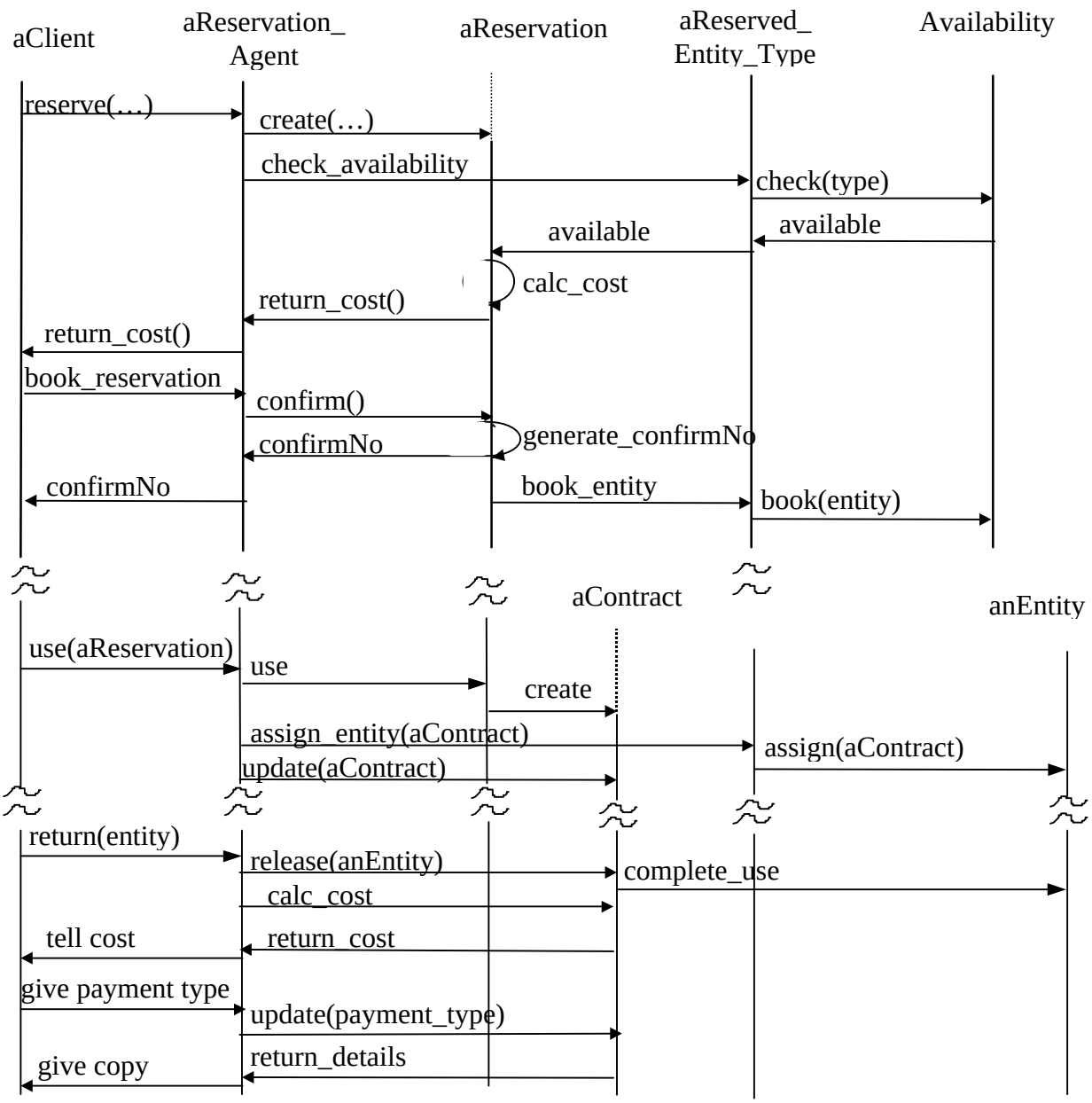

Figure 5. Sequence Diagram for Reservation and Use

6. Known Uses

We have found the following uses in the literature or our experience:

- A person reserving a seat for a show or a flight [Bak97]
- A client reserving a hotel room [Fer98a]
- A client reserving a vehicle [SA98]

- A patron reserving a book in a library [Fer98a]
- A person reserving a videotape in a video store [Fer98a], [Joh96].

6. Related Patterns

The NATURE project [Nat98] has some patterns, e.g. Resource Allocation and Resource Usage, which address some of the same aspects but from a different point of view. Coad [Coa97] has a resource request pattern that has some common aspects. Fowler [Fow97] discusses plans and resource allocation. The reservation and subsequent allocation of manufacturing components also has aspects in common with this pattern [Fer97]. In all these cases the entities are not reused and there is no need for a record of use.
Some aspects of book and video reservation are discussed in [Bra98], but they do not consider the use of the reserved entity.
Our pattern uses as a component the Type Object pattern [Joh96]. In fact, as indicated in [Fer98], these complex patterns usually include subpatterns that are meaningful in their own right.

References

[Bak97] S. Baker, *CORBA: Distributed objects using Orbix*, Addison-Wesley 1997.
[Bra98] R.T.V.Braga, F.S.R.Germano, and P.C.Masiero, "A confederation of patterns for resource management", *Procs. of PLOP'98*, http://jerry.cs.uiuc.edu/~plop/plop98
[Coa97] P. Coad, "*Object models: Strategies, patterns, and applications*" (2nd Edition), Yourdon Press, 1997.
[Fer97] E. B. Fernandez and Z. W. Peng, " An object-oriented model for manufacturing inventory control systems", Tech. Report TR-CSE-97-30, Dept. of Computer Science and Eng., Florida Atlantic University, April 1997.
[Fer98] E. B. Fernandez, "Building systems using analysis patterns", *Procs. 3rd Int. Soft. Architecture Workshop (ISAW3)*, Orlando, FL, November 1998. 37-40.
[Fer98a] E. B. Fernandez, Class Notes for COP5330, Introduction to Objected-Oriented Software Design, Dept. of Computer Science, Florida Atlantic University, Boca Raton, Florida.
[Fow97] M. Fowler, *Analysis patterns – Reusable object models*, Addison-Wesley, 1997.
[Gam95] E. Gamma et al. *Design Patterns – Elements of reusable object-oriented software*, Addison-Wesley 1995.
[Joh98] R. Johnson and B. Woolf, "Type Object", Chapter 4 in *Pattern Languages of Program Design 3,* Addison-Wesley, 1998.
[Nat98] NATURE project, http://www.city.ac.uk/~az533/main.html
[Ngu] K. Nguyen and T. Dillon, "An alternative solution to the observation pattern problem", in *PLOP98*. http://jerry.cs.uiuc.edu/~plop/plop98/final_submissions
[SA98] Software Architects. Exercise in course SA4012, *Testing Object-Oriented Systems,* 1998.