An Analysis Pattern for Repair of an Entity

Eduardo B. Fernandez[1] and Xiaohong Yuan[2]

[1]Dept. of Computer Science and Engineering,
Florida Atlantic University,
Boca Raton, FL 33431
ed@cse.fau.edu

[2]Dept. of Computer Science,
North Carolina A & T State University,
Greensboro, NC 27411
xhyuan@ncat.edu

Abstract

This analysis pattern models repair events in a repair shop. This pattern focuses on the basic aspects of the repair events in a repair shop, from the initial estimates until completion of the repair. Details of the repair shop and of the type of entities it repairs are left for the specific application or for complementary patterns. This pattern represents a minimum application that can be applied to a variety of situations and it can be combined with other related patterns to describe more complex applications. It can also be further abstracted to describe other situations, not just repair.

1. Introduction

We present an analysis pattern that describes performing repairs in a repair shop. This pattern is in the category of what we have called Semantic Analysis patterns [Fer00a], thus the pattern focuses on the basic aspects of the repair events in a repair shop, without detailing the specific type of repair shop, entity or customer. Details of the repair shop and the type of entities it repairs are left for the specific application or for complementary patterns. The purpose of this type of pattern is to serve as a starting point when translating requirements into an actual design. This pattern represents a minimum application so that it can be applied to a variety of situations and it can be combined with other related patterns to describe more complex applications. It can also be further abstracted to apply to other situations.

Sending a broken entity to a repair shop for repair is a common real-life problem. Customers bring broken entities, for example, computers or cars, to a repair shop and a reception technician makes an estimate of their repair. If the customer agrees, the entity is assigned for repair to some repair technician, who keeps a Repair Event document. All the Repair Event

documents for an entity are collected in its repair log. A repair event may be suspended because of a lack of parts or other reasons. Repairs can also be cancelled.

2. Intent

This pattern describes the repair events in a repair shop, including making an estimate for repairing a broken entity, assigning the broken entity for repair to some repair technician, and logging the repair event of the broken entity.

3. Context

Customers bring computers to a computer repair shop that is part of a chain of similar shops (Figure 1). A customer may take a computer to several computer repair shops for estimates (class **Estimate**), and choose one repair shop to repair the computer. A reception technician makes estimates, and a repair technician is in charge of repairing computers and keeping a log of the repairs (class **RepairEvent**). Each computer has a repair log which records all the repair events of this computer in these shops (class **RepairLog**).

This example is a particular case of a more general problem of repairing broken entities, which appears in a variety of contexts, e.g., repairing a car, a watch, a house appliance, a copy machine, etc. To make this an analysis pattern useful to develop conceptual models we should abstract those aspects and define a generic pattern that subsumes all these cases. The generic pattern is then tailored to the specific situation.

4. Problem

How can we model the repair events in a repair shop? We need to develop a starting model that includes making an estimate for repairing a broken entity, assigning the broken entity for repair to some repair technician, and logging the repair event of this broken entity.

5. Forces

- Conceptual models are hard to build. In particular, modeling the events in repair shops is not a simple problem. An initial generic model can be a great help.
- We want to model repair events in a repair shop. However, there is a variety of repair situations that have similar structure but differ in details.
- A user may apply to several places for estimates of repair services. However, only one of the estimates may be selected to receive the required services.

- The model must include representations of real-life documents, e.g., Estimate, Repair Event, Repair Log. Otherwise, it would be complex to create these necessary documents.

6. Solution

6.1 Requirements

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

(1) Get an estimate for a repair. The reception technician, according to the problems reported by the customer, checks the broken entity and generates an estimate for the cost of the repair.

(2) Repair a broken entity. The customer decided to repair the broken entity at this repair shop. A repair technician is assigned to repair this broken entity. The repair can be cancelled by different reasons.

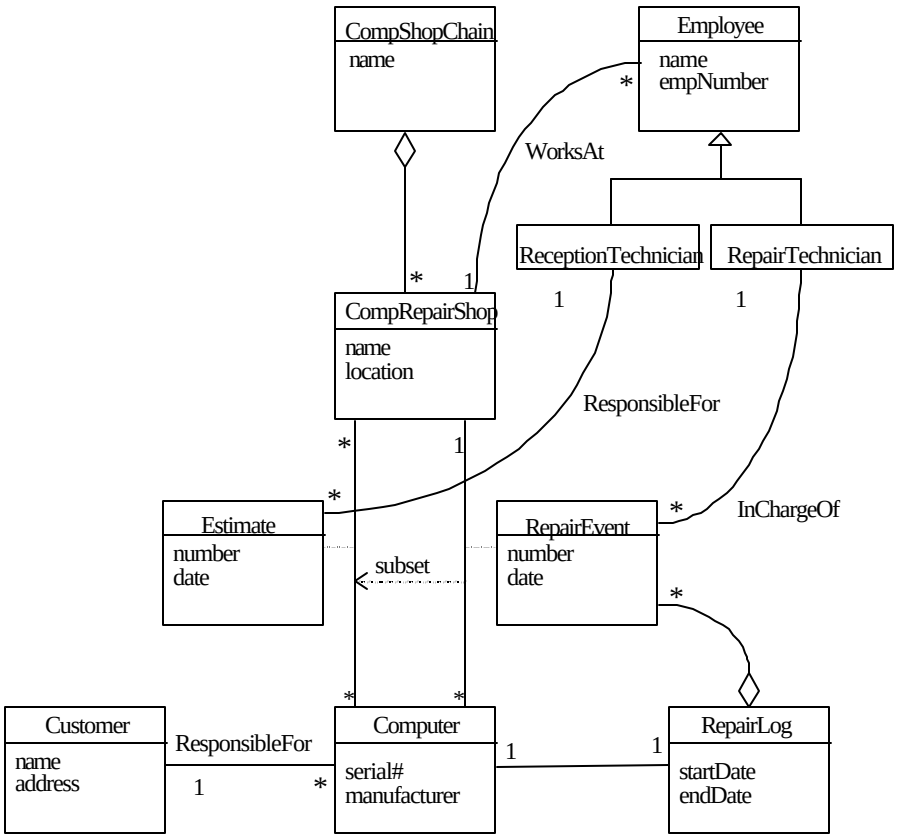

Figure 1. Class diagram for the computer repair shop

6.2 Class model

Figure 2 is a class diagram for the realization of these Use Cases. This diagram is an abstraction and extension of the diagram of Figure 1. A **BrokenEntity** is considered for repair at a **RepairShop.** A **Customer** is responsible for the **BrokenEntity** (he will pay the costs of repair). The many-to-many association between **BrokenEntity** and **RepairShop** reflect the facts that a broken entity can be estimated by different shops in the chain. The many-to-one association between these same classes describes that one of these estimates may become an actual repair. A computer that has been repaired at least once has a repair log that collects all its repair events. The collection of repair shops is described by class **RepairShopChain**. Typically, two types (or roles) of employees are required: a **ReceptionTechnician**, who prepares estimates, and a **RepairTechnician**, who performs repairs.

6.3 Dynamic aspects

Figure 3 shows the state diagram for class **RepairEvent**. A RepairEvent could be in the states of *AssignRepairTechnician*, *InRepair*, *Suspended*, and *Completed*. The superstate around *InRepair* and *Suspended* implies that cancellation of repairs can be performed from these two states. Figure 4 shows the sequence diagram for getting an estimate for a repair, it assumes that the customer is a new customer and it adds it to the list of customers. Figure 5 shows the sequence diagram for assigning repair jobs to technicians.

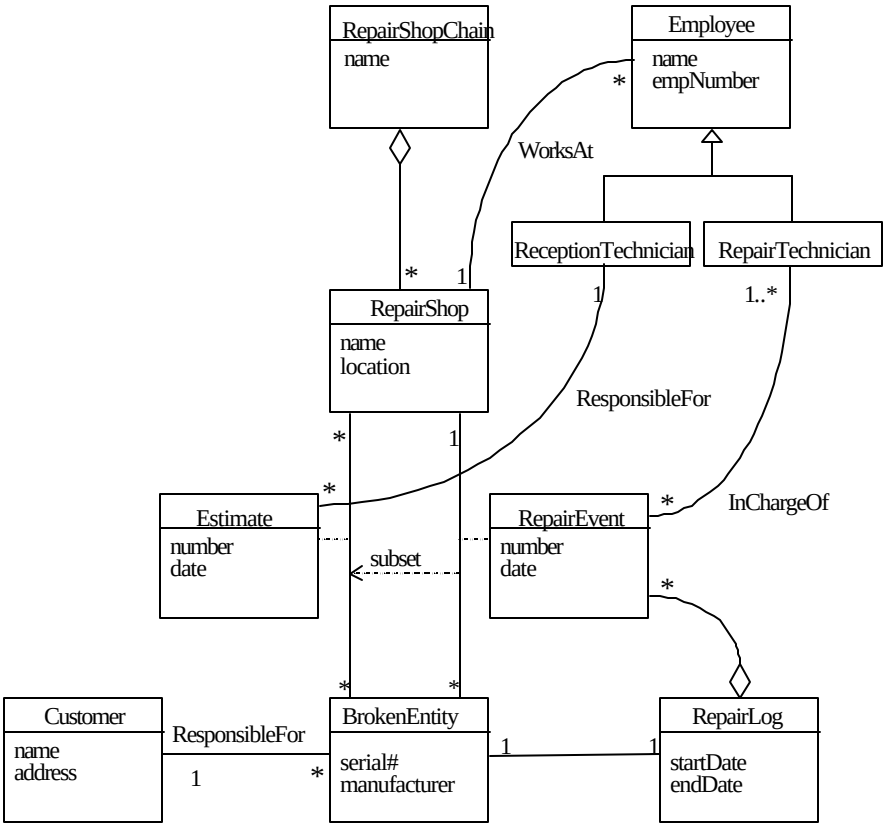

Figure 2. Class diagram for the repair shop pattern

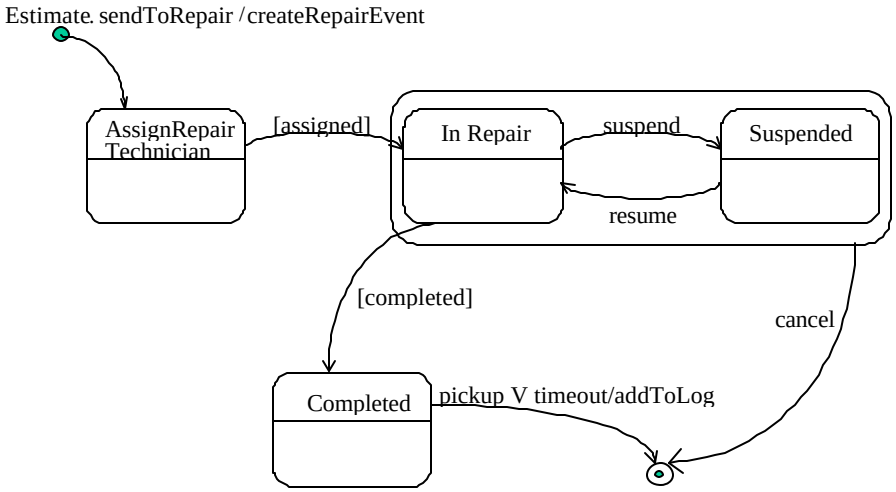

Figure 3. State diagram for class RepairEvent

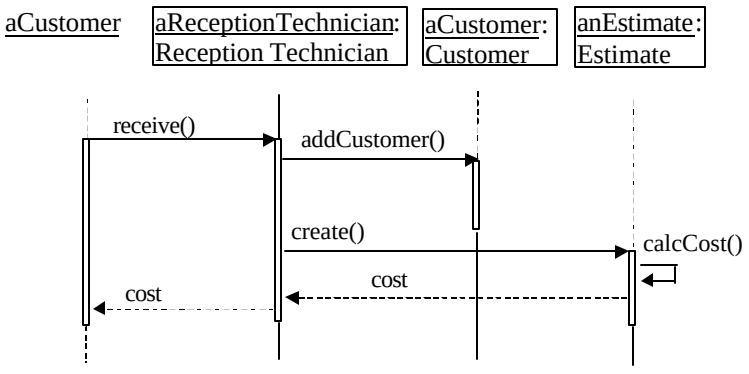

Figure 4. Sequence diagram for getting an estimate for a repair

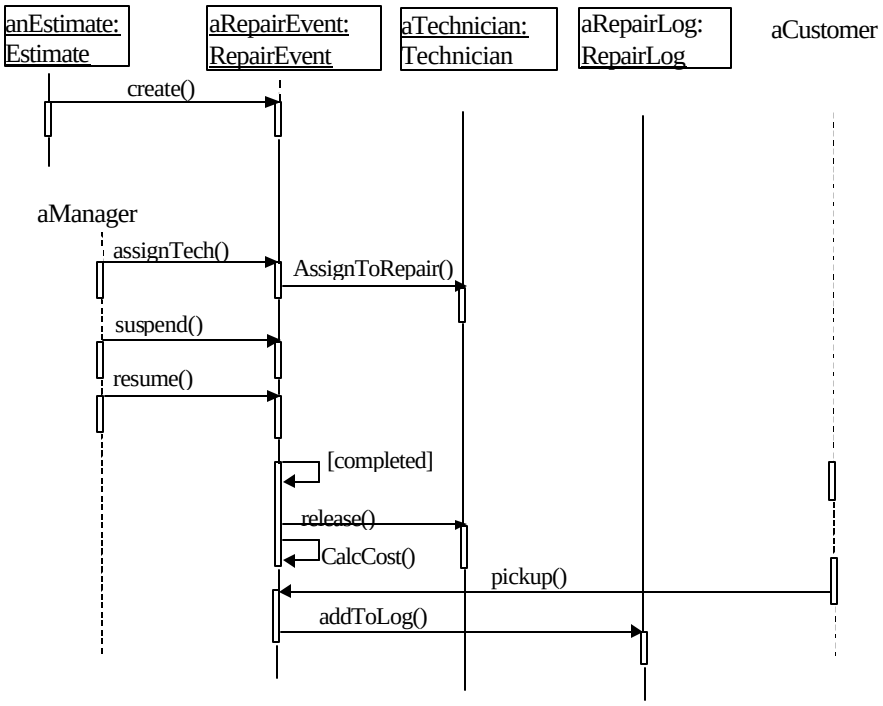

Figure 5. Sequence diagram for assigning repair jobs to technicians

7. Consequences

This pattern presents the following advantages:

- It describes in a generic way a variety of real-life situations (see Section 8). This means that one can use it in different applications.
- Typical documents are represented by classes, e.g., Estimate.
- It provides a systematic structure for the activities needed in repair activities and as such it optimizes the handling of repairs in a shop.

A disadvantage is that not all the situations described by this pattern are exactly alike:

- In situations such as repairing a watch or repairing a car, the customer brings the broken entity to the repair shop. The customer may also ship the broken entity to the broken shop. In situations such as repairing a refrigerator, the repair technician has to come to the house of the customer to perform the repair.
- Sometimes the reception technician is able to estimate the repair cost directly according to the report of the problems from the customer. In other situations, the reception technician (or the manager) has to assign a diagnosis technician to examine the broken entity and report the problems, and then give the estimate of repair cost. The customer may be charged for the diagnosis cost or the examination cost whether he decides to repair the

broken entity at this repair shop or not. The diagnosis technician often turns out to be the repair technician if the customer decides to repair the broken entity at this repair shop.

- Some repair shops specialize on repairing a specific type of entity, even an entity from a specific manufacturer, or a specific model. Other repair shops provide repair services for many types of entities, for example, refrigerators, dishwashers, air-conditioning units, and other house appliances.
- The customer may have a contract with some repair service provider, so that the customer does not have to pay for each repair service. Instead, the customer pays certain amount each year according to the contract. In this case, the repair cost is not estimated.
- In the case that the broken entity is under warranty (within some time period), the customer does not need to pay for the repair service. Therefore the repair cost is not estimated.
- Depending on whether free repair technicians are available, the customer may have to wait for hours, or even days to get a repair technician assigned to the broken entity. The duration of the repair varies depending on the problems with the broken entity.

All this means that the pattern needs to be tailored for specific applications.

Some aspects not represented in the pattern are:

- Description of contextual and environmental aspects of the broken entity
- Billing and payment policies
- How to deal with varieties of customers, e.g., individual, corporate, preferred
- Exceptions, e.g., bringing back the repaired entity (repair not satisfactory)
- How to order parts when the parts required to repair the broken entity are not available
- How to keep track of the parts in stock
- How to queue up customer requests when no free technicians are available
- How to ship the repaired entity to the customer when shipping is required
- The customer makes an appointment with the repair technician to bring in the broken entity, or an appointment for the repair technician to come to the house (or other location) where repair is needed

These aspects have been left out to make the pattern more general. They can be described by complementary patterns (see Section 8) or directly in an application model.

8. Known uses

The following are examples of uses of this pattern:

- A customer takes a car to a car repair shop (or car dealer) for repair
- A customer takes a broken computer to a computer shop for repair
- A company orders repair for a copy machine
- A customer orders repair for a refrigerator in her house

9. Related patterns

When the parts needed to repair the broken entity are not available, the repair shop needs to order the parts. Therefore the Order/Shipment [Fer00b] pattern complements this pattern. The customer may need to make an appointment with the repair technician, part of the Reservation and Use pattern [Fer99] can be used to capture this aspect. The Stock Manager pattern [Fer00c] is also complementary, it can be used to keep track of the parts in stock in the repair shop. To consider payment of orders a detailed treatment of money aspects can be found in [Hay96] and [Fow97].

The pattern includes two other patterns: a Collection pattern (A repair shop is a collection of repair shops), and a Role pattern [Bau00] (Employees appear in the roles of reception and repair technicians).

The pattern can be made even more general [Fer00a]. The same structure needed to model repairs can also describe the action of applying for admission or service to several institutions and selecting one of them. Examples include hospital admissions and student application and registration. This generalization is useful when one is building conceptual models; however, the list of aspects that need to be tailored becomes longer in this case.

Acknowledgements

We thank our shepherd Mary Lynn Manns for her valuable suggestions that have significantly improved this paper.

References

[Bau00] D.Baumer, D. Riehle, W. Siberski, and M. Wolf, "Role Object", Chapter 2 in *Pattern Languages of Program Design 4* (N. Harrison, B. Foote, and H. Rohnert, Eds.). Also in *Procs. of PLoP'97*, http://jerry.cs.uiuc.edu/~plop/plop97

[Fer00a] E.B. Fernandez and X. Yuan. "Semantic analysis patterns", *Procs. of 19th Int. Conf. on Conceptual Modeling, ER2000,* 183-195.

[Fer00b] E. B. Fernandez and X. Yuan. "Analysis patterns for the order and shipment of a product", *Procs. of PloP 2000*. http://jerry.cs.uiuc.edu/~plop/plop2k

[Fer00c] E. B. Fernandez. "Stock Manager: an analysis pattern for inventories ", *Procs. of PloP 2000*. http://jerry.cs.uiuc.edu/~plop/plop2k

[Fow97] M. Fowler, *Analysis patterns -- Reusable object models*, Addison- Wesley, 1997.

[Hay96] D.Hay, *Data model patterns-- Conventions of thought*, Dorset House Publ., 1996.