An Analysis Pattern for Invoice Processing

Eduardo B. Fernandez
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431
ed@cse.fau.edu

Xiaohong Yuan
Department of Computer Science
North Carolina A&T State University
Greensboro, NC 27411
xhyuan@ncat.edu

ABSTRACT

We discuss an analysis pattern for invoice processing. The pattern describes events such as the creation and validation of an invoice, followed by the payment process. This pattern is composed of two simpler patterns that describe the creation and payment of the invoice, respectively. The composite 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. The component patterns have value of their own and can be used independently.

Keywords: accounting, analysis patterns, business modeling, invoices, object-oriented design

1. INTRODUCTION

Invoices are used in all places where services or products are provided and they usually contain a list of charges for the services or products rendered. In some environments, e.g. a supermarket, invoices are delivered right away, or they can be delivered via email or post mail. After the invoice has been received, it must be paid, which involves different methods from which one can choose. We present here an analysis pattern that describes the processing of an invoice, including its creation and its payment. This pattern is an example of a S*emantic Analysis Pattern (SAP)* as it expresses semantic aspects of the model. A SAP realizes a few use cases and is composed of a few basic patterns. Following the concepts behind this type of pattern [Fer00a], we describe a set of use cases that together illustrate its idea.

The pattern is a composite pattern and we present first its two components, Invoice Creator and Invoice Payment, which we combine later into an Invoice Processing pattern. The component patterns have value of their own and can be used independently; however, if both aspects are needed the composite pattern presents some advantages over the independent application of each one. Notice that we do not care about the specific contents of the invoice, there are many varieties which are discussed in books such as [Fow97, Hay96, Sil01]; our emphasis is on the needed processing. Our patterns present enough detail to be used as guidelines for implementation and they are directed to analysts, architects, and developers.

The intents of these patterns are:

Invoice Creation. Describe the process of creating and preparing an invoice for a product or service, followed by its validation.

Invoice Payment. Allow a client to make a payment for an invoice corresponding to services used and/or products bought.

Invoice Processing. Describe the conceptual steps of invoice processing, including its creation, preparation, validation, and payment.

2. INVOICE CREATION

Describe the process of creating and preparing an invoice for a product or service, followed by its validation.

2.1 Context

This pattern is valuable for institutions or enterprises of any type that require payment for products or services. In some cases the customers may have registered or opened accounts with these institutions but that is not a necessary condition.

2.2 Problem

In order for a client to pay for the services provided or for the products she bought, an invoice needs to be created that describes what should be paid, including rules and conditions. The process of preparing such an invoice can be rather complex and also time consuming. Different systems deal with this process in a different way, but they all have common aspects; however, we need a systematic and uniform way of preparing invoices.

The following forces affect the solution:

- In the creation of an invoice, there can be more than one actor involved, so some ordering of actions between these parties has to be defined.
- Items in an invoice can have different discounts and can belong to various categories which are managed differently, so the preparation of the invoice needs to be done according to specified procedures and policies.
- The invoice may specify different ways of payment and other conditions, so we must define them explicitly.
- We need to validate that the invoice is complete and in a valid state before we send it.
- Preparation and validation should be done by different people (separation of duty).
- We need to keep track of who created and who validated an invoice.
- There are invoices which are not bills of sale, like for example the 'shipping invoice' which details all the parts that are included in the specific order. So, some information that we include into the invoice has to be optional, while other information has to be mandatory.
- We might want to control or restrict who performs each activity.

2.3 Solution

First, a user creates and prepares the invoice according to predefined institution procedures; then another user validates it. The stakeholders that participate in this process are the creator of the Invoice and the Invoice Validator. The following use cases correspond to these activities:

1) *Create and Prepare Invoice*. After purchasing some items (products or services), a document is issued, which incorporates all the relevant items and their costs. This step can be done automatically or manually either by a person or by a group of persons. After an invoice has been created, we have to add the products or services to it and supply additional information regarding every item from the invoice.

2) *Validate Invoice*. After an invoice has been created and prepared, it has to be validated by either a system or a person. This step ensures that the quality and completeness of the invoice are met.

Figure 1 is a class diagram for the realization of these use cases. Class **InvoiceCreator** defines an interface for creating an invoice. It also provides a way of preparing the invoice by adding or deleting items from it, calculating the amounts to be paid, discounts applicable, and payment deadlines. Class **Invoice** represents the document in which all the goods or services that have been bought are incorporated together with the nature of each item. Class **InvoiceValidator** is used to ensure that the invoice that resulted from the steps described above is in a consistent form that complies with the trade usage [Cro] and possible regulations. Classes **BillingPolicy** and **ValidationRule** include business policies that apply to the preparation and validation of invoices.

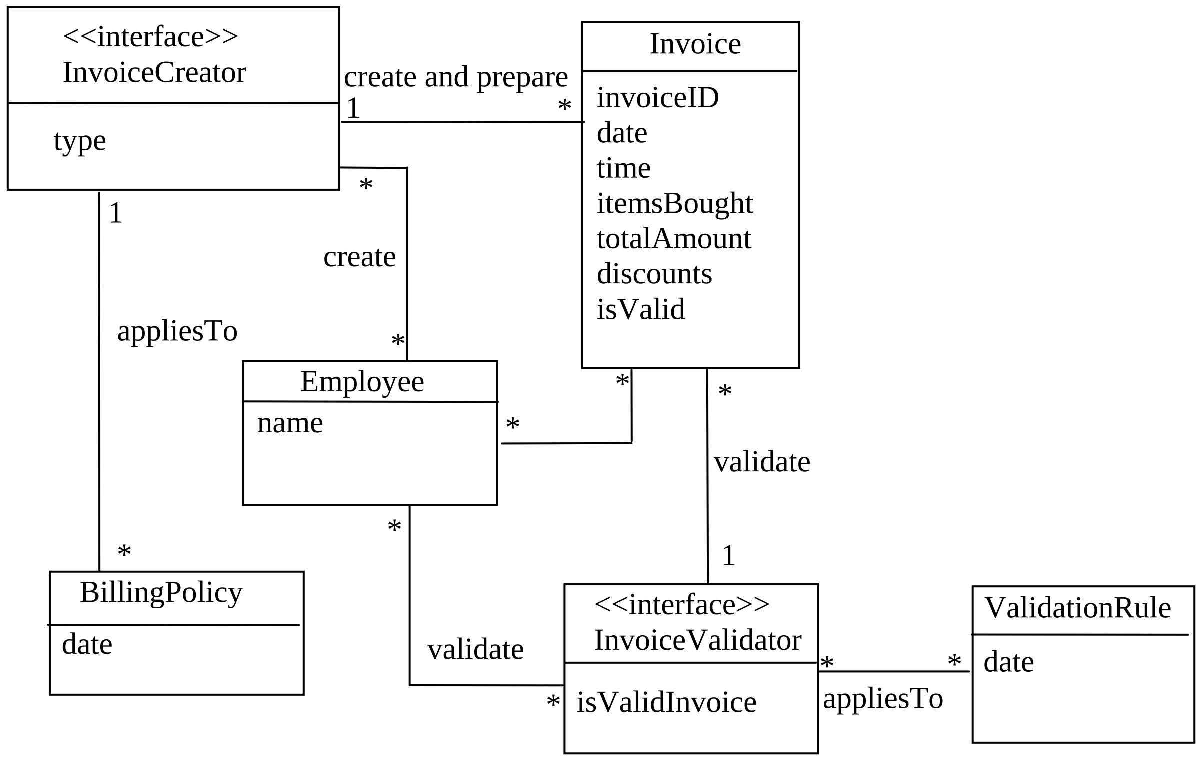

Figure 1: Class diagram for Invoice Creation

Figure 2 shows a sequence diagram for creating, preparing and validating an invoice. After created, the invoice may be initialized. The creator adds items and other specific information until there are no more items to be added to the invoice. After each item is added the cumulative amount is calculated. Once all the items have been supplied, the invoice signals the validator that the invoice is complete. After validation the invoice is ready to be sent.

2.4 Consequences

The pattern provides the following benefits:

- It describes an abstract invoice preparation process that can be tailored to different specific situations.
- We can separate specific stages of preparation along the invoice preparation process and their ordering.
- The creator and the validator of the invoice are roles; that is they can be performed by the same or different individuals. They can also be performed by groups of people.
- We can apply the separation of duty principle by having different users perform creation and validation through different interfaces with access control.
- We can keep track of who prepared and who validated an invoice.
- Every characteristic of the use case is represented in this pattern, without providing any implementation aspects, which leaves it open to different possibilities.
- There are many ways of preparing an invoice, this pattern defines an abstract view of its essential aspects and it can be adapted so that it fits any scenario.
- We can add explicit authorizations to each action; in particular, creation and validation can be restricted to specific actors.

Several liabilities of the pattern can be defined in terms of the aspects that were not covered, such as:

- Different types of creators and validators. This depends on the system, and can range from creators that are individuals to creators that are software programs or institutions (parties).
- The specific contents of the invoice depend on accounting practices. [Hay96] and [Sil01] provide details of these aspects. Because of this we have left out any descriptions of the items bought.

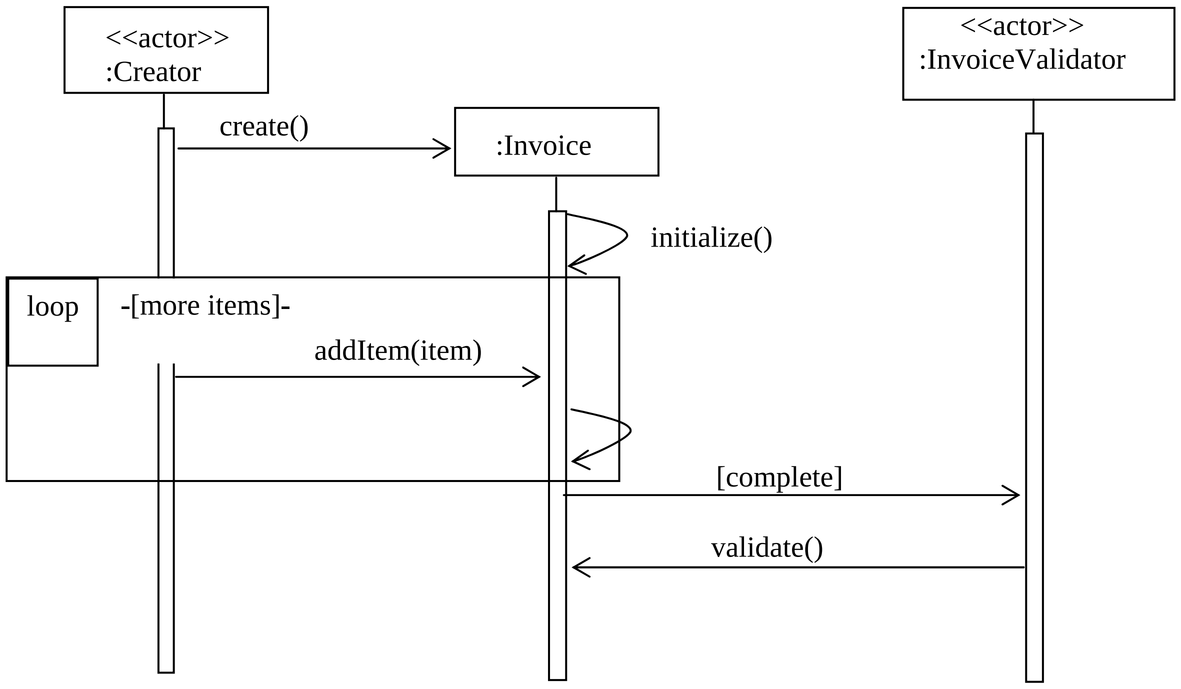

Figure 2: Creating, preparing and validating an invoice

2.5 Known uses

Some examples of use for this pattern are:

- A point-of-sale system in any department store that sells products, such as Macy's.
- An on-line shopping store, where people use the Internet to log onto an on-line shop in order to buy different items, e.g. Amazon.
- Monthly invoicing for telephone or internet service, e.g. Comcast.
- SAP has an Invoice management product where they create and prepare invoices [SAP].

2.6 Related patterns

- Creation of invoices may use a Factory pattern [Gam94] in the design stage.
- The Account pattern [Fer02] defines accounts where the invoice costs may be charged.
- A Strategy pattern [Gam94] can be used to select different ways to calculate prices [Pie].
- Billing policies can be defined using a Business Rule pattern [Ars00].
- Fowler [Fow97], Hay [Hay96], and Silverston [Sil01] deal with accounts and describe how invoices are produced but don't show their handling.
- Order and Shipping [Fer00b] may be used to describe the items bought.

3. INVOICE PAYMENT

Allow a client to make a payment for an invoice corresponding to services used and/or products bought.

3.1 Context

This pattern is valuable for institutions or enterprises of any type that require payment for products or services. In some cases the customers may have registered or opened accounts with these institutions but that is not a necessary condition.

3.2 Problem

A client has to be able to pay an invoice in a way convenient to him; this implies that different payment methods have to be incorporated in the payment process, so that a client can elect the best option that fits his needs. Such a payment has to be validated in order for both the client and the system to have confirmation that the payment has been done in an appropriate way.

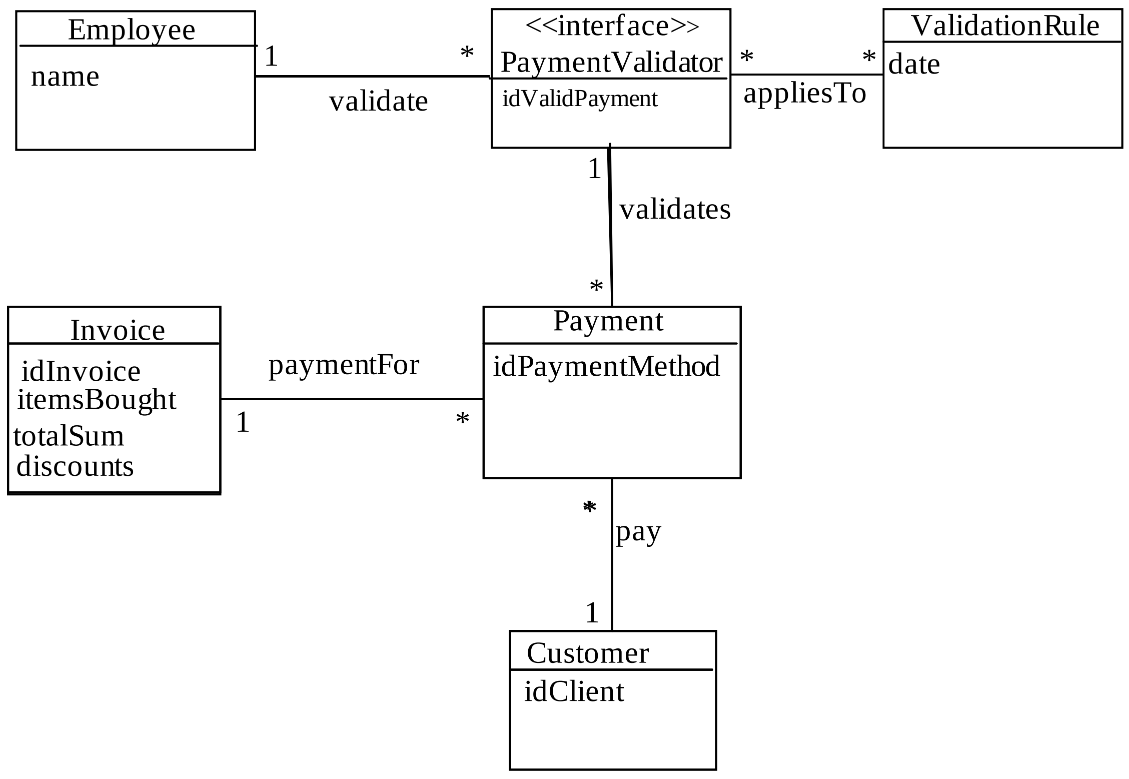

Figure 3: Class Diagram for Invoice Payment

The solution is affected by the following forces:

- We need to provide different possibilities to make a payment for an invoice; otherwise we might lose customers.
- The system and the client need a convenient way of keeping track of the payments made.
- Validation of every received payment has to be made to ensure that the client's information is correct and in accordance to the requirements and regulations of each system.
- We need to keep track of who validated a payment.

3.3 Solution

Separate the validation approach from the payment so we can apply different validation approaches. It should be noticed that an invoice could be sent before or after the actual payment has been made. Such flexibility is incorporated in the design by separating this process.

Figure 3 shows the class diagram for Invoice Payment. The **Invoice** class represents the amount that must be paid by the customer. The **Payment** class represents the payment made by the client for the products and/or services incorporated in the invoice. It should incorporate information related to the invoice for which the payment is being made, the payment method chosen, the id of the client that makes the payment. The **PaymentValidator** validates payments according to **ValidationRules**. **Employee** keeps track of who validated a payment. The **Customer** class represents the customer that makes payments for the given invoice.

Figure 4 shows the sequence diagram for payment of an invoice. A Client initiates the payment process by making a payment according to the total sum in the invoice. The Payment is validated by the PaymentValidator, according to the validation rules, after which the status of the payment is returned to the client.

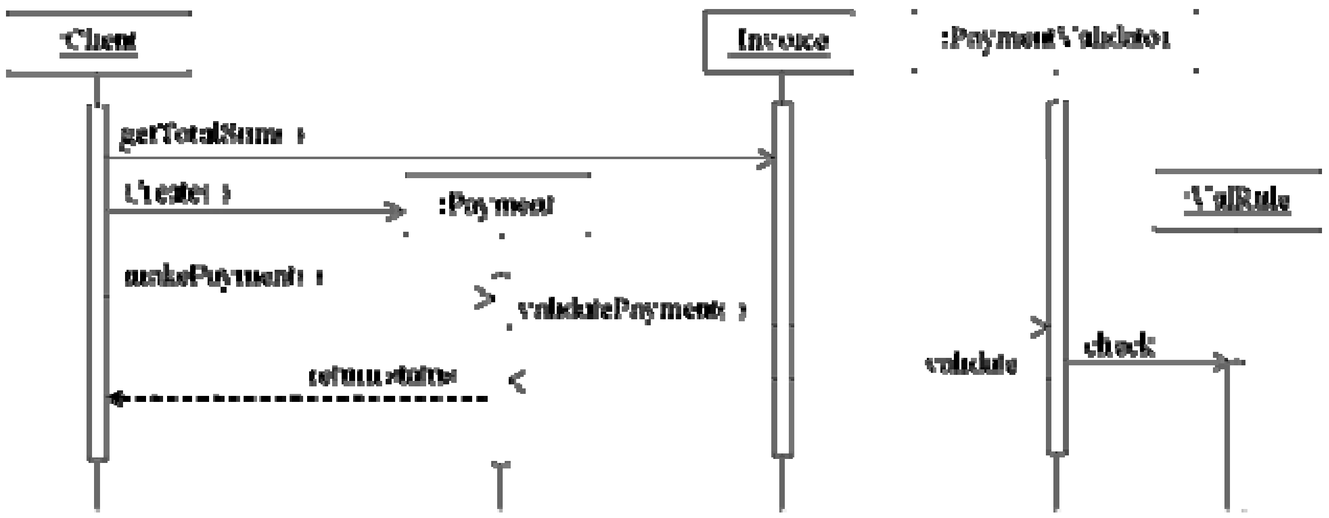

Figure 4: Sequence diagram for Invoice Payment

A Payment can be in the following states: *Active, Paid,* and *Validated*. When the invoice is sent, payment is activated. After the customer has paid (Payment in Paid state), his payment needs to be validated. After the validation process has been completed, the payment is closed.

3.4 Consequences

This pattern has the following advantages:

- It allows one to control and keep track of who validated payments.
- The payments can be related to its corresponding invoice.
- We can apply different validation rules to a payment.
- We can indicate different ways to pay an invoice by adding classes to Payment as in [Hay96] or [Sil01].

In the liabilities we can mention that aspects such as payment in installments or payments by check, have been left out.

3.5 Known uses

Some examples of use for this pattern are:

- A point-of-sale system in any department store that sells products, such as Macy's.
- An on-line shopping store, where people use the Internet to log onto an on-line shop in order to buy different items, e.g. Amazon.
- Monthly invoicing for telephone or internet service, e.g. Comcast.
- SAP has an Invoice management product where they create and prepare invoices [SAP].

3.6 Related patterns

- This pattern may include a Factory pattern [Gam94] to create payments.
- The Validation Rule could be an instance of the Business Rule pattern [Ars00].
- Fowler [Fow97], Hay [Hay96], and Silverston [Sil01] deal with accounts and describe how invoices are produced but don't show their handling.
- A pattern Pay For The Resource Transaction [Bra99], describes the process of paying a bill in installments.

4. INVOICE PROCESSING

Describe the conceptual steps of invoice processing, including its creation, preparation, validation, and payment.

4.1 Context

This pattern is valuable for institutions or enterprises of any type that require payment for products or services. In some cases the customers may have registered or opened accounts with these institutions but that is not a necessary condition.

4.2 Problem

There are many systems where we need to combine the functions of creating and preparing an invoice, and paying that invoice, including the corresponding validations. How do we represent this process in a general and abstract manner?

The solution is affected by the following forces:

- *Workflow*. The creation, preparation, and validation of an invoice requires specific actors, actions in specific sequences, must follow specific rules, and must be easy to change.
- *Separation of duty*. Preparation and validation should be done by different people.
- *Flexibility*. There should be ways to define who is responsible for a payment and the way of payment.
- *Memory*. The system and the client need a convenient way of keeping track of the payments made.
- *Validation*. Every prepared invoice and every received payment has to be validated to ensure that the client's information is correct and in accordance to the requirements and regulations of each system.
- *Accountability*. We need to keep track of who created an invoice, who validated it, and who validated a payment.
- *Authorization*. We may want to control who performs specific actions.

4.3 Solution

Combine the solutions of the two component patterns, Invoice Creator and Invoice Payment. Figure 6 shows the class model of this pattern as a combination of the two previous class models. The sequence diagrams of Figures 2 and 4 still apply. Figure 7 shows the activity diagram of creating and paying for an invoice.

4.4 Consequences

This pattern combines the consequences of its component patterns plus:

- *Workflow*. The actors and the sequence of their actions follow business rules, which can be easily changed.
- *Authorization, separation of duty, and accountability*. We can control who performs specific actions on invoices, we can separate these functions, and we can keep track of who performed these actions.
- *Flexibility*. It is easy to add new payment methods, discount policies, or types of customers.
- *Memory*. Clients and systems can keep track of the payments made.
- *Validation*. We can conveniently validate the preparation of invoices and the reception of payments.
- *Security*. We can define a secure version of this pattern applying instances of the Role-Based Access Control pattern as we have done for analysis patterns describing law applications [Fer07] and others.

The pattern has the following liabilities:

- We left out details of the customers. Those can be found in [Hay96] or [Sil01].
- We did not consider specific ways of payment. They also can be found in [Hay96] or [Sil01].
- We did not consider physical delivery and storage of invoices. They are considered in [Net09].

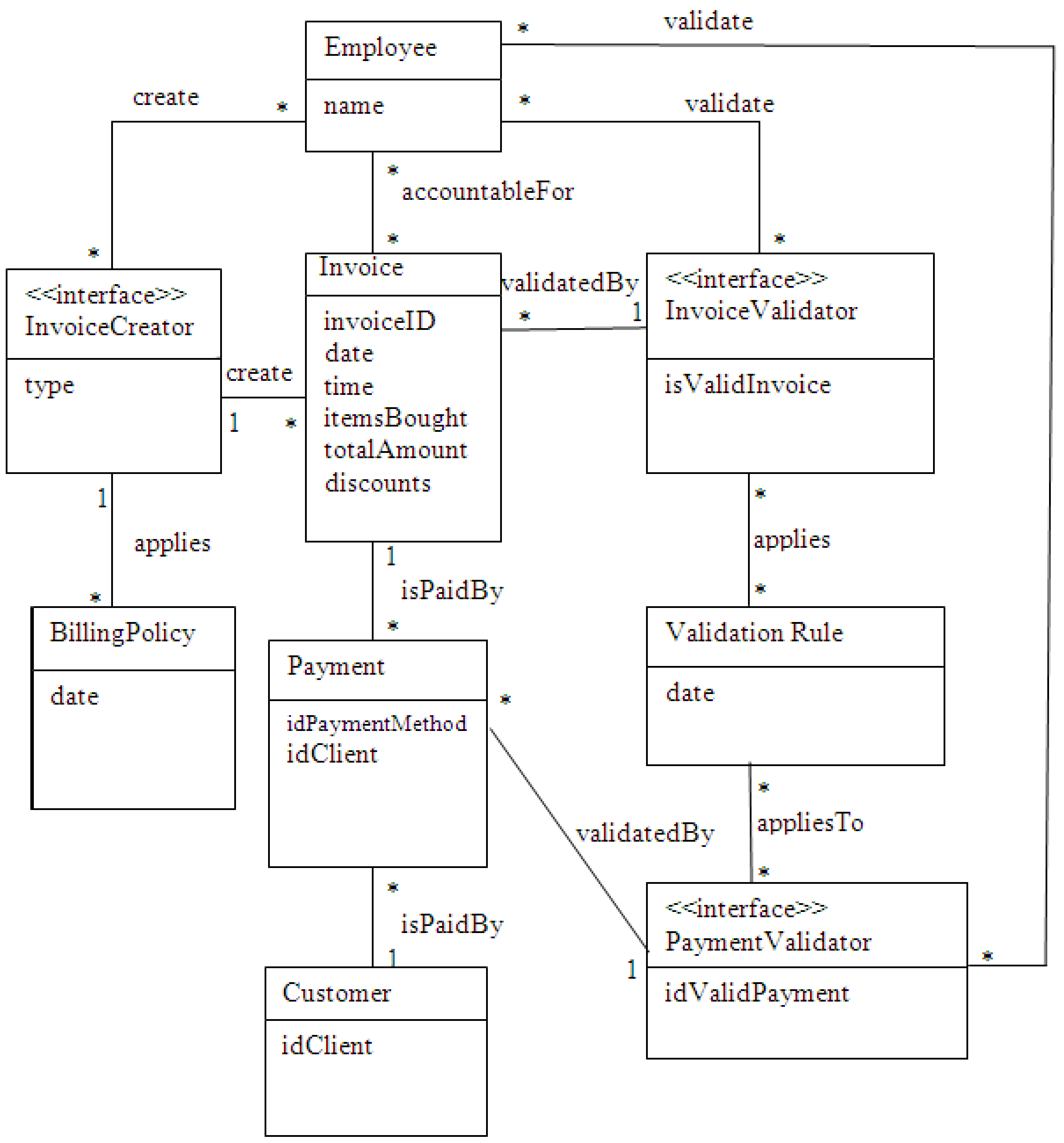

Figure 5: Class diagram for Invoice and Payment

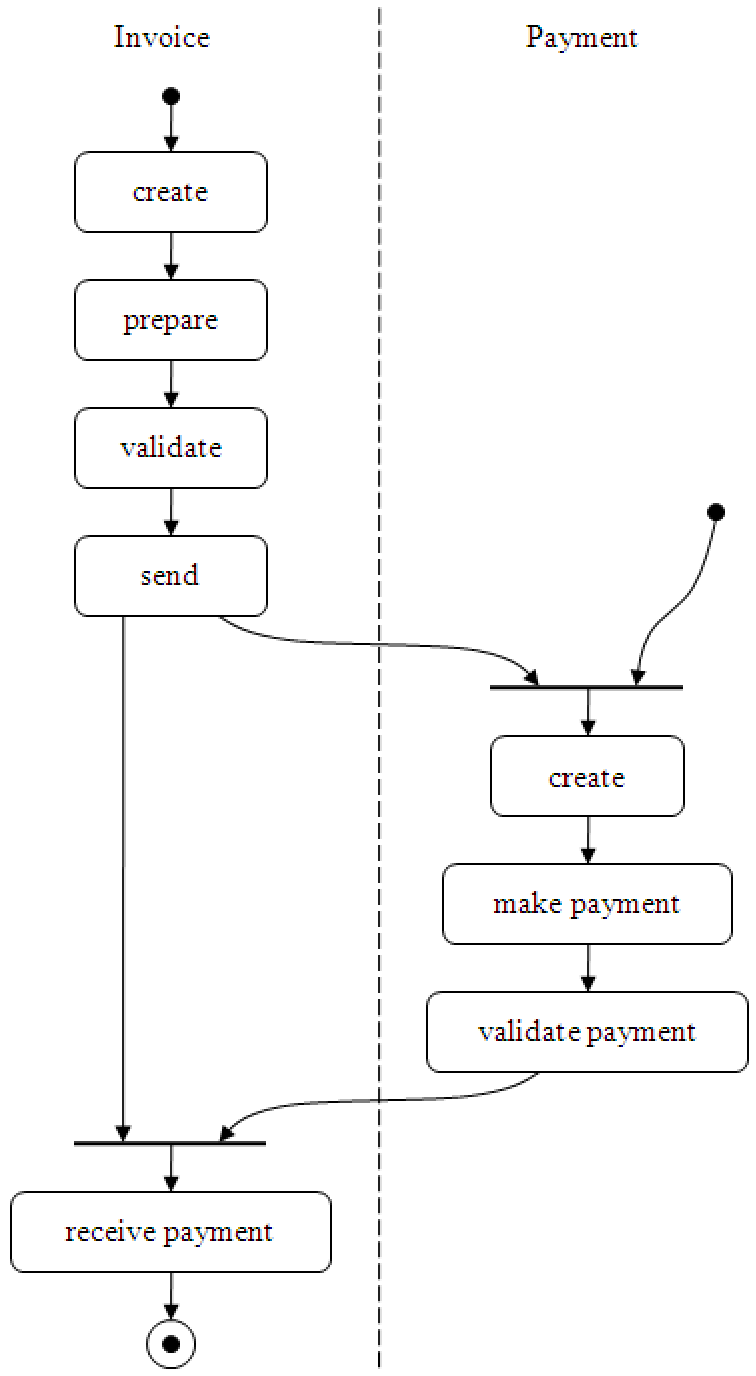

Figure 6: Activity diagram for Invoice and Payment

4.5 Known uses

Some examples of use for this pattern are:

- A point-of-sale system in any department store that sells products, such as Macy's.
- An on-line shopping store, where people use the Internet to log onto an on-line shop in order to buy different items, e.g. Amazon.
- Monthly invoicing for telephone or internet service, e.g. Comcast.
- SAP has an Invoice management product where they create and prepare invoices [SAP].

4.6 Related patterns

- Creation and payment of invoices may use a Factory pattern [Gam94].
- The Account pattern [Fer02, Fow97] defines accounts where the invoice costs may be charged.
- A Strategy pattern [Gam94] can be used to select different ways to calculate prices or discounts.
- Billing and payment policies can be defined using a Business Rule pattern [Ars00].
- The Order and Shipment pattern [Fer00b], describes how a customer can place an order for a product and the subsequent shipment of the product.
- Internet shops may require invoicing [Fer01].
- The contents of invoices can be found in [Fow97], [Hay96], and [Sil01].
- Creation and delivery of invoices is considered in [Net09].

5. ACKNOWLEDGEMENTS

Mihai Fonoage started this pattern as a class project. Our shepherd, Rosana T. Braga provided valuable comments. The participants of the PLoP 2009 Writers Workshop provided valuable improvements.

6. REFERENCES

[Ars00] A. Arsanjani. "*Rule Object Pattern Language*". *Proceedings of PLOP 2000.*

[Bra99] R.T.V. Braga, F.S.R. Germano, and P.C. Masiero, "A Pattern Language for Business Resource Management", *Proceedings of the 6th Annual Conference on Pattern Languages of Programs (PLOP'99),* v7, 1-33, Monticello, Illinois, USA, August 1999.

[Cro] Crowley Maritime Corporation, Definition of the term *Invoice*. http://www.crowley.com/

[Fer00a] E. B. Fernandez and X. Yuan, "Semantic analysis patterns", *Proceedings of the 19th Int. Conf. on Conceptual Modeling, ER2000,* Lecture Notes in Computer Science 1920, Springer 2000, 183-195. Also available from: http://www.cse.fau.edu/~ed/SAPpaper2.pdf

[Fer00b] E.B. Fernandez, X. Yuan, and S. Brey, "An Analysis Pattern for Order and Shipment of a Product," *Procs. of the 7th Pattern Languages of Programs (PLoP) Conference*. http://st-www.cs.uiuc.edu/~plop/plop00, August, 2000.

[Fer01] E. B. Fernandez, Y. Liu, and R.Y. Pan, "Patterns for Internet shops ", *Procs. of PLoP (Pattern Languages of Programs) 2001,* http://jerry.cs.uiuc.edu/~plop/plop2001/accepted_submissions/accepted-papers.html

[Fer02] E.B.Fernandez and Y.Liu, "The Account Analysis Pattern", *Procs. of EuroPLoP (European Pattern Languages of Programs*). http://www.hillside.net/europlop/EuroPLoP2002/

[Fer07] E. B. Fernandez, D. L. laRed M., J. Forneron, V. E. Uribe, and G. Rodriguez G. A secure analysis pattern for handling legal cases", *Procs. of the 6th Latin American Conference on Pattern Languages of Programming (SugarLoafPLoP'2007), 178-187.* http://sugarloafplop.dsc.upe.br/AnaisSugar2007_WEB.pdf

[Fow97] M. Fowler, *Analysis Patterns-Reusable Object Models*, Addison-Wesley, 1997.

[Gam04] Gamma, E., Helm, R., Johnson, R. and Vlissides, J.: *Design patterns –Elements of reusable object-oriented software*, Addison-Wesley 1995.

[Hay96] D.Hay, *Data model patterns-- Conventions of thought*, Dorset House Publ., 1996. Chapter 7: Accounting.

[Net09] M.Netter and G.Pernul, "Integrating security patterns into the electronic invoicing process", *Procs. of the Third Int. Workshop on Secure System Methodologies using Patterns (SPattern 2009).* 150-154.

[Pie] John Pierce, Design patterns, Cal State University, San Jose, CA http://www.cs.sjsu.edu/faculty/pearce/oom/patterns/behavioral/strategy.htm

[SAP] SAP United States, http://www.sap.com/usa/solutions/solutionextensions/invoice-management/index.epx

[Sil01] L. Silverston, *The data model resource book (revised edition),* Vol. 1, Wiley 2001, Chapter 7, vol. 1: Invoicing.