Towards Use Case and Conceptual Models through Business Modeling[1]

J. García Molina, M. José Ortín, Begoña Moros, Joaquín Nicolás, Ambrosio Toval

Software Engineering Research Group[2], Departamento de Informática y Sistemas
Facultad de Informática, Universidad de Murcia
Campus de Espinardo. C.P. 30071. Murcia, Spain
{Jmolina, Mjortin, Bmoros, Jnicolas, Atoval}@um.es

Abstract. A guide to requirements modeling is presented in this paper, in which use cases and the conceptual model are directly obtained from a business modeling based on UML activity diagrams. After determining the business processes of the organization, and describing their workflows by means of activity diagrams, use cases are elicited and structured starting from the activities of each process, while the concepts of the conceptual model are obtained from the data that flow between activities. Furthermore, business rules are identified and included in a glossary, as part of the data and activities specification. One notable aspect of our proposal is that use case and conceptual modeling are performed at the same time, thus making the identification and specification of suitable use cases easier. Both use case and conceptual modeling belong to the requirements analysis phase, which is part of a complete process model on whose definition we are currently working. This process is being experimented in a medium-sized organism of a Regional Public Administration.

1 Introduction

Since UML [1] was adopted as the OMG standard language for modeling, a large number of UML-based process models for object-oriented (OO) development have been proposed. These approaches are usually use-case driven, and therefore capture the functional system requirements as use cases, which provide the foundation for the rest of the development process: iteration planning, analysis, design, and testing.

Nowadays, a lot of research concerning use cases can be found in the literature, and there is an agreement on their usual misunderstanding and on the lack of precise guides to properly organize them. In this sense, several approaches have been published (cf. [3, 7, 8]) dealing with issues such as use case granularity, the level of detail in which use cases should be described, and the suitability of creating a use case hierarchy.

Based on the OOram *three-model architecture* [13] and the IDEA method [2], we are working on the definition of a UML-based process model for application in the information systems domain. This process includes a business modeling phase, aimed

[1] Partially supported by the CICYT (Science and Technology Joint Committee), Spanish Ministry of Education and Ministry of Industry, project MENHIR TIC97-0593-C05-02.
[2] Member of RENOIR (European Requirements Engineering Network of Excellence).

at describing the business processes of the organization, and which then allows the elicitation of the system use cases and the conceptual model in a simple, straightforward way. Inspired by the process view of the enterprise model of the three-model architecture, we describe each business process by a UML activity diagram with swimlanes. Next, we identify the system use cases from the activities, and the *concepts* (domain classes) from data (the information objects that flow among the activities).

In this paper, we describe an approach to business modeling, and how it can serve as a basis for requirements analysis (conceptual and use cases models). This approach is currently being experimented within the framework of a project which aims to provide a specific process model, based on requirements, for the development of information systems with intensive use of data. The scope of the research is the Regional Information Systems and Telecommunications Office (RISTO), of the Ministry of Finance (MF) in the Regional Government of Murcia (RGM)– Spain.

This paper is structured in the following way: some issues related to use cases are presented in section 2; next, a summarized version of our approach can be found in section 3; section 4 deals with business modeling, and our proposal for its realization; the rules that govern the transition from the business model to the use case and the conceptual models are presented in section 5; and finally, in section 6 we present our conclusions.

2 Use Cases in Practice

Most of the process models currently proposed for UML are defined as *use-case driven*. A use case can be defined as a sequence of actions, including variations, that the system can execute and that produce an observable result which has some value for an actor that interacts with the system [1].

Although the success of use cases is usually justified by the simplicity and intuitiveness of the technique, several authors (cf. [3, 7, 8]) have pointed out difficulties in discovering and specifying useful use cases, and finding consensus about how to organize and manage them. These are the reasons why we believe that it is necessary to establish a set of principles to guide the identification, description and organization of use cases.

Some interesting discussions about use case management have been made by T. Korson and A. Cockburn. Korson [7] claims that requirements (and therefore use cases) have to be hierarchically organized, in order to be able to understand them, reason about them, refine them and use them to validate the developed products. He establishes a set of recommendations, including the following statements: i) each level of use cases should conform to a complete set of requirements, i.e. each level does not add any new requirements, but it refines those in the previous level; and ii) the use case hierarchy should not be the result of a functional decomposition and it should be developed in an iterative and incremental way.

Cockburn [3], on the other hand, uses the concept of *goal* to organize use cases hierarchically. He basically distinguishes *strategic goals* (the business processes of the organization) and *user goals* (the system functions). Strategic goals are traced to a set of user goals, and, likewise, a user goal can be subsequently decomposed into a set of user goals. Thus, the concept of *summary goal* arises, which corresponds to either a composite user goal, or to a strategic goal.

Last but not least, another important issue is the allocation of use case modeling in the process model. Use case modeling is usually conceived as a previous step to conceptual modeling. However, Korson [8] claims that it is not possible to create adequate and useful use cases (nor correctly implement them) without understanding the domain, and therefore, use case and conceptual modeling have to be two parallel activities.

Usually, use cases are intuitively elicited from the system specification, an then the entities of the conceptual model are elicited based on the use cases specification. In the following sections, we present an approach to obtain the use cases and the conceptual models from a business model in a systematic way. Inspired in the *OOram Three Model Architecture* [12,13], business modeling is performed by means of UML activity diagrams. After determining the business processes of the organization, and describing their workflows by means of activity diagrams, use cases are elicited and structured starting from the activities of each process, while the concepts of the conceptual model are obtained from the data that flow between activities. Furthermore, business rules are identified and included in a glossary, as part of the data and activities specification. One notable aspect of our proposal is that use case and conceptual modeling are performed at the same time, thus making the identification and specification of suitable use cases easier.

3 Our Proposal in Brief

Our approach can be summarized in the following steps, which are not performed sequentially, but in an iterative and, at times, concurrent way:

1. The identifying and delimiting of the *business processes* within the organization under study, according to the enterprise strategic *goals*. A *business use case* is defined for each business process, and a *business use case diagram* is used to show the context and the boundary of the enterprise.
2. The discovery of the *roles* involved in the business processes, and their description in a *role model* that describes the interactions among the workers in the enterprise during the execution of a business use case. These interactions are represented by UML interaction diagrams (behavioral aspect) and a stereotyped class diagram (structural aspect).
3. The modeling of the *workflow* of every business process by means of activity diagrams, thus showing the interaction among roles to achieve the goal. The business rules that constrain the business processes are also elicited.
4. The extraction of the *system use cases* from the activities making up a business use case (which are included in the corresponding activity diagram).
5. The establishing of the *conceptual model* from the data (*information objects*) in the activity diagrams.

All the elements created during the modeling are specified in a glossary. The overall process outlined before is shown in Fig. 1.

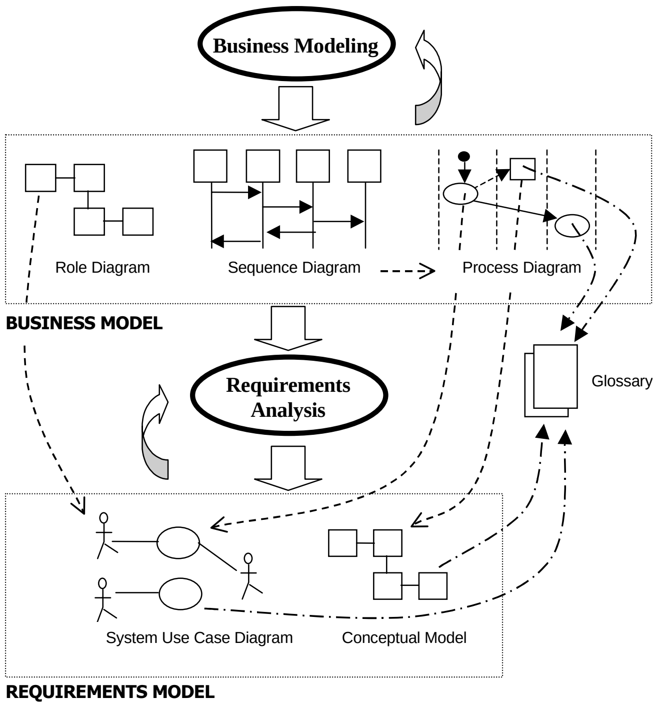

Fig. 1. Traceability relationships between Business and Requirements models

Moreover, use case are organized in two levels: firstly, each business process is associated with a *business use case*, which maps to the Cockburn's strategic goals; secondly, from these business use cases, a collection of *system use cases* is defined, once the activities involved in each business process have been considered.

4 Business Modeling

To achieve its goals, an enterprise organizes its activity through a set of business processes. Each business process is characterized by a collection of data which are produced and manipulated by means of a collection of *tasks* in which certain *agents* (for instance, workers or departments) participate according to a *workflow*. In addition, these business processes are constrained by *business rules*, which determine the policies and structure of the information of the enterprise.

The purpose of business modeling is to describe every business process, specifying the corresponding data, activities (tasks), roles (agents) and business rules. At this stage, our purpose is to understand the activity of the organization related to the system to build, considering "what" the system is supposed to do, instead of "how" it will support its goals.

The first step of business modeling is to capture the business processes of the organization under study. The elicitation of an adequate set of business processes is a crucial task, since it establishes the boundaries of the later modeling process. Following the concept of strategic goal given by Cockburn [3], we capture the business processes from the main goals of the enterprise. Firstly, we consider the strategic goals of the organization. Since these objectives are extremely complex, they are decomposed into a set of a few subgoals, which are more specific and which have to be accomplished to achieve the strategic goal. These subgoals can be subsequently divided into some more subgoals, and therefore a hierarchy of goals arises. In our research, we have experienced that two (or a maximum of three) levels of decomposition are enough. For every one of those subgoals we define a business process, whose purpose is to achieve that goal. We represent every business process as a business use case, which is initially specified by using a textual description.

We will use as running example the case study of a company that manufactures products by demand (following a *just in time* scheme). The strategic goals of that company might include *Satisfy a customer order*, *Increment sales by 25%*, or *Improve the manufacturing time by 15%*. Thus, the goal *Satisfy an order* can be divided in the following subgoals: *Register order*, *Manufacture product*, *Stock management* and *Generate orders to providers*. These are the subgoals that we use to discover the business processes.

4.1 Role Identification in the Business Context

Once business use cases are identified, we must discover the agents that are involved in their realization. Every agent plays a certain role when it collaborates with other agents to carry out the activities making up that business use case. In fact, we identify roles which are played by enterprise agents (including workers, departments, and devices) or external agents (as customers or other systems). For the moment, we only pay attention to those roles with which the organization interacts to carry out its business use cases. In our example, we have two roles which are clearly external to the organization: *Customer* and *Provider*.

To have a general view of the collection of business processes of the organization, we can create a business use case diagram, in which every business process is represented as a use case (see Fig. 2).

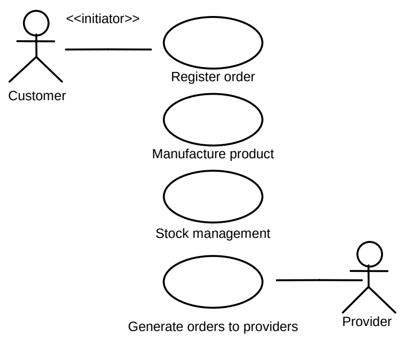

Fig. 2. Business use case diagram for the *Just in Time Manufacturing System*

The business use case diagram allows us to show the boundary and environment of the organization under study. This is the reason why only the business actors that correspond to external roles are shown in this diagram. In this way, the business use cases only involving roles which are internal to the organization are not connected to any actor. The business use case diagram shown in Fig. 2 is a UML use case diagram that consists of business use cases and actors. The diagram also specifies that the agent *Customer* initiates the realization of its related use case, while *Provider* is an actor that just participates in the associated use case.

4.2 Describing the Business Use Cases

The following step consists of describing in detail every business use case previously identified. We will focus on one of the business use cases of our example, namely *Register Order*, whose description is shown in Fig. 3. This description can be easily validated by the users.

1. A customer submits an order, which has to include the order date, the customer data and the desired products. A clerk of the sales department might also introduce the order on request of a customer who has placed their order by phone, or has sent it by fax or ordinary mail to the sales department of the company.
2. The clerk revises the order (and completes it, if necessary), and begins its processing by sending it to the catalog manager, who is in charge of its analysis.
3. The catalog manager analyses the viability of each product of the order separately:
   - if the ordered product is in the catalog, its manufacturing is accepted.
   - otherwise, it is considered as a *special product*, and the catalog manager studies its manufacturing:
     - if it is viable, the manufacturing of the special product is accepted;
     - if it is not viable, the product is not going to be manufactured.
4. Once the whole order has been studied, the catalog manager...
   - informs the sales department if every ordered product is accepted or rejected;
   - in the case that all the products of an order have been accepted, a work order for every product is created, starting from a manufacturing template (the standard one, if the product was in the catalog, or a new one, specifically designed for the product, if it was not present in the catalog). Every work order is sent to the manufacturing manager, and its launching is considered pending.
5. The clerk informs the customer about the final result of the analysis of his or her order.

Fig. 3. Description of the *Register Order* business use case

Now we have to determine the internal agents that play a role in each business use case. We have identified the roles that belong to the business environment and now, we have to study the description of each business use case, and observe the complete set of involved roles, both external as well as internal to the organization. For instance, the roles in the business use case example are *Customer, Clerk, Catalog manager* and *Manufacturing manager* (where the last three are internal to the system).

The static (or structural) aspect of the collaboration among the roles to perform the business process, can be represented in a *role diagram*, in which each role (a stereotyped UML class) appears linked to the roles with which it can collaborate (see Fig. 4). Therefore, this diagram allows us to express the knowledge that some roles have about the others, as well as the characteristics of the relationships between roles (such as multiplicity). In addition, this diagram can serve to define some characteristics of the identified roles, such as their attributes and responsibilities. Ortín and García Molina [11] discuss role modeling in UML in more detail.

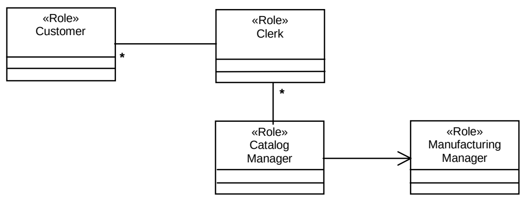

Fig. 4. Role diagram for the *Register Order* business process

Next, we create *scenarios* to show the behavioral aspect of the role collaboration. Here, we use UML sequence diagrams (see Fig. 5), in which the objects denote the instances of the roles that participate in the interaction.

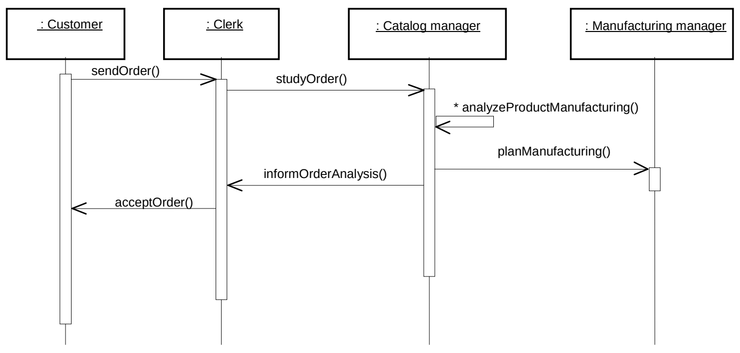

Fig. 5. Sequence diagram for the *Register Order* business use case

In every business use case we must distinguish the basic path of the interaction (in our example, request of an order that is finally accepted), and the existing alternative paths (for instance, canceling or rejecting an order). To improve legibility, it is convenient to associate several scenarios to the same business use case, instead of including all the possibilities in the same sequence.

The OOram three-model architecture [13] includes a business model represented through *process views* based on the standard IDEF0 [5], showing the workflow performed to obtain some goal of the organization, indicating the roles that are in charge of each activity, and the data required and produced by each activity. We consider these kind of diagrams very suitable for model business use cases, since they are very expressive and simple, thus facilitating discussions with users. These diagrams can be easily adapted to UML, by using activity diagrams with swimlanes. Thus, we use this type of UML diagram, which we have called *process diagrams*, to show the workflow that makes up the business use case in more detail.

A process diagram that includes the scenario of Fig. 5 is shown in Fig. 6. There exists a swimlane for every role participating in the scenario, including the activities performed by that role. The diagram also shows the data needed and produced by each activity, as well as the synchronization required between different activities. Data appear as objects that flow between activities and can have a state. For instance, the activity *Pass on* order receives a proposed order and initiates its revision (see Fig. 6). We refer to these objects as *information objects*.

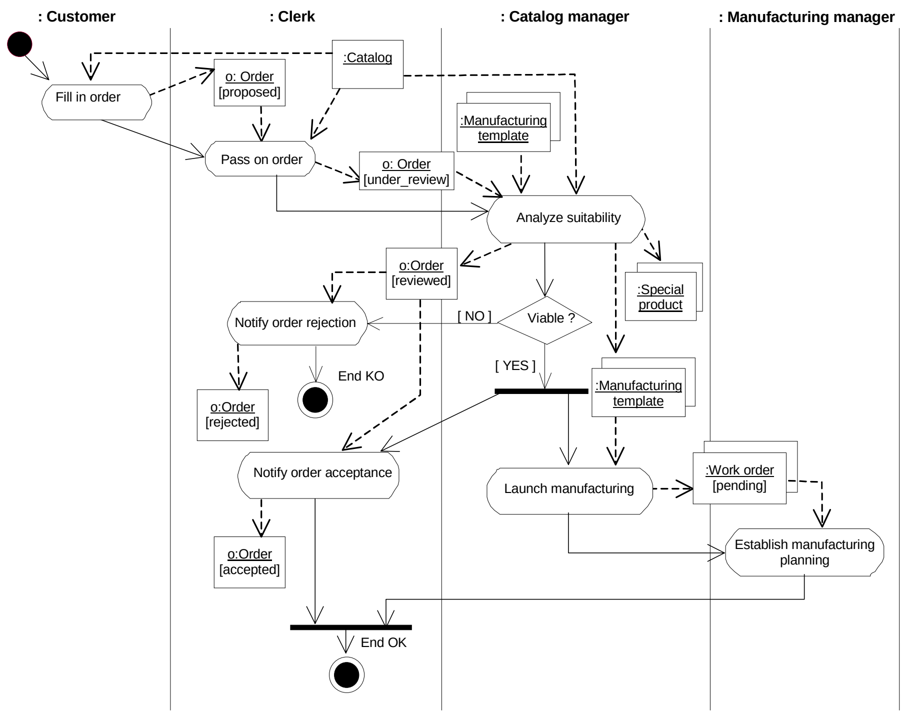

Fig. 6. Process diagram for the *Register Order* business use case

During the description of a business use case by means of a process diagram, it is possible to find out an activity which is complex enough to be described in another activity diagram. Thus, this new activity diagram will describe a subgoal in relation to the goal related to the original business process. In this fashion, business processes can be hierarchically organized.

4.3 Business Rules Specification

In an organization, both business processes and the data that they manage are constrained by business rules. As Whitenack [14] states, business rules are seldom explicitly captured during product development, regardless of the fact that they are often important constraints on system behavior. Because there is no well-defined framework in which to plug rules, and because there is a variety of rules types that are not well understood, rules are often ignored until the implementation phase.

With the aim of understanding the various types of rules that we have to take into account in a requirement specification, we use the taxonomy described by Odell [9]. This classification is simple but comprehensive and it covers every kind of business rule. Business rules are divided into two categories: constraint and derivation rules.

- *Constraint rules* specify policies or conditions that constrain the structure and behavior of the objects. Moreover, these rules can be subdivided into *stimulus-response rules* (they constrain the behavior and specify the conditions that must hold to activate an operation), *operation constraints rules* (they specify conditions that must be true before and after an operation is performed) and *structural rules* (they specify constraints about object types and associations, and these rules must always hold true).
- *Derivation rules* specify policies and conditions to infer or calculate facts from other facts in the business.

According to this classification, we explicitly collect each type of rule in the business model by means of the specification of the activities and information objects shown in the process diagrams. These specifications are gathered in a glossary.

Each information object is described by a set of attributes and their integrity constraints (if any exist). Therefore, we explicitly state structural and derivation rules. On the other hand, the semantics of each activity is described by its *source* (that is, the previous activities), *agent* (who is the responsible for doing the activity), *pre* and *post conditions* (stating what has to hold before and after the activity). The latter establish the operation rules, whereas stimulus-response rules are represented in the source part, where we express the order between the activities. Fig. 7 shows how the information object *Order* and the activities *Launch Manufacturing* and *Notify Order Acceptance* could be specified.

... **Information Object:** ***<u>Order</u>*** **Attributes** Order code Submission date Maximum delivery date Set of {Products} Customer Total price Current state **Constraints** - Order code uniquely identifies the order, and has to be assigned automatically by the system - Submission date has to be previous to the Maximum delivery date. - An order must contain at least one product, but there is no maximum number. - An order is always processed for one (and only one) customer. - The total price is calculated starting from the price of each product in the order. ... **Domain Class**: *-to be specified-*	... **Activity:** ***<u>Launch manufacturing</u>*** **Source:** Analyze suitability **Agent**: Catalog manager **Precondition**: All ordered products are viable and a manufacturing template exists for all of them. **Postcondition**: A work order for each product has been created, and has been sent to the Manufacturing manager for planning. **Use Case**: - *to be specified-* **Activity:** ***<u>Notify Order Acceptance</u>*** **Source**: Analyze suitability **Agent**: Clerk **Precondition**: All ordered products are viable and have been accepted. **Postcondition**: The customer is informed that his or her order has been accepted. Order state is *Accepted.* **Use Case:** - *to be specified-* **...**

Fig. 7. Glossary: information objects (left) and activities (right)

The glossary will have a hypertext (cross-references) structure, in order to maintain the traceability relationships from the business processes to the classes and use cases that specify the functionality of the system. During the development life cycle, suitable links will be established. For example, each activity will be connected to the system use case where it is performed, and each information object will be linked to its related domain class, as we will see in the next section.

5 Requirements Analysis: Use Case and Conceptual Models

Starting from the business model described in the previous section, it is possible to obtain both the initial collection of system use cases and the earliest conceptual model in a systematic and straightforward way. Next, we are going to describe separately how to obtain each model.

The requirements which are elicited and specified in this phase will be included in a Software Requirements Specification (SRS) document. We recommend the use of a SRS standard template, such as the IEEE 830-1998.

5.1 Transition to the Initial System Use Case Model

We believe that the activities in a process diagram have the appropriate level of granularity to be associated to a single system use case. In this manner, we create a use case for each activity of a process diagram that will be supported by the software system. Thus, the role performing the activity will be the primary actor of the use case. Note that, according to the use case definition, not all the activities in a process diagram will be considered as use cases, but only those which have some value for an actor.

For instance, consider that the *Customer* role could not fill in the order on their own (through a web form, for example). Then, he or she would have to send all the data by fax or by telephone or some other way, as the result of the activity *Fill in Order*. As this activity would be performed outside the software system, neither the *Customer* role (since he or she will not interact with the software system) nor the activity *Fill in Order* (see Fig. 6) would be created in the system use case diagram. Fig. 8 shows the *system use cases diagram* for the business process *Register order*, whose process diagram was illustrated in Fig. 6, with the consideration that all the activities will be supported by software.

The use case diagram shown in Fig. 8 contains the architecturally most important use cases. We have to remark that some use cases could not be directly obtained from the process diagrams, but would be detected when the elicited use cases were described, thus gaining more knowledge about the requirements to be supported. These new use cases represent functions that the system must perform in order to achieve the goal related to some existing system use case. For instance, in our running example, to *Analyze suitability* it is necessary to look up in the products catalog whether an ordered product exists and so this catalog must be up to date. Thus, we would have to add the use case *Maintain product catalog*. Another example of a new use case could be *Maintain manufacturing templates*.

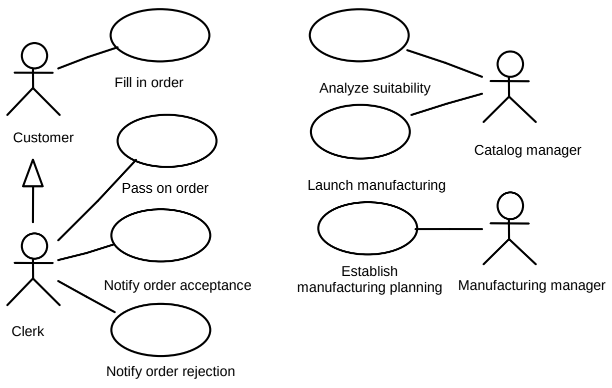

Fig. 8. Initial system use cases diagram

Moreover, the use cases could be organized into levels (two o three levels as maximum) according to the hierarchical decomposition proposed in the business modeling.

Each use case will be described by means of a template which can be filled in starting from the description of the associated activity, which is specified in the glossary as we saw before. We have chosen the template proposed by Coleman [4] because it combines simplicity and completeness, as shown in Fig. 9.

Use Case	Launch manufacturing.
Description	Work orders for every ordered product will be created, and will be sent to the Manufacturing manager so as to be planned.
Actors	Catalog manager.
Assumptions	- All ordered products are viable. - There exist manufacturing templates for all of them.
Steps	1. REPEAT 1.1. Obtain a product from the order. 1.2. Look for the manufacturing template of this product. 1.3. Create the work order. 1.4. Store the work order with *pending* state.
Variations	--
Non-Functional	--
Issues	--

Fig. 9. *Launch manufacturing* use case description

Once we have described the use case, it will be linked to the related activity in the glossary, with the aim of keeping traceability between business use cases and system use cases.

Relationships between uses cases could also be found, such as *include*, if common aspects to various use cases are found, and *extend*, to express an optional or alternative path in a use case. Nevertheless, we agree with the recommendations about not overusing these relationships and not showing them in the use case diagrams.

In order to complete this phase, non-functional requirements should be stated. If they are related to a specific use case, they will be specified in the proposed use case template [4]. If they are global to the system, they will be gathered in a section of the chosen SRS.

5.2 Transition to the Initial Conceptual Model

The information objects that flow between the activities of a business use case represent domain data and therefore they are a good base to create the initial conceptual model. This conceptual model will include the concepts and their relationships, and will be represented by a UML class diagram, where concepts are represented by classes (domain clases).

Each information object in the glossary will become a concept. In the design phase, this concept will become a class if the software system is going to manipulate that information. From the specification of an information object in the process diagram, we will obtain the definition of the concept, that is, its attributes, relationships with other classes, and constraints. For example, from the specification of *Order* shown in Fig. 7, we could obtain i) the attributes *code, submissionDate, maximumDeliveryDate, totalPrice, state*, ii) the associations *Customer-Order* and *Order-Product*, and iii) the constraints that could be expressed textually or by means of OCL (*Object Constraint Language*) as {*maximumDeliveryDate* > *submissionDate*}.

Furthermore, it should be noticed that when a conceptual model evolves to a class diagram, responsibilities can be obtained from certain constraints already specified in the glossary. For example, the Order class could have responsibilities such as *getProducts, calculateTotalPrice, calculateMaximumDeliveryDate*, or *changeState*.

In the same way as we connected the activities with the use cases in the glossary, we will link each information object to the domain class that represents it in the system. The class diagram depicting the first conceptual model for our running example is shown in Fig. 10.

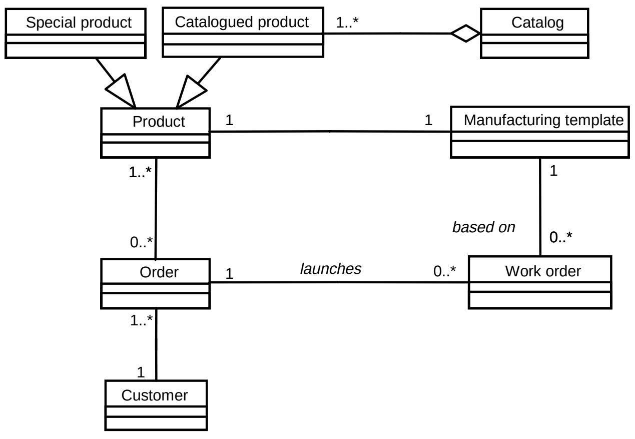

Fig. 10. Initial conceptual model from the *Register Order* business process

At this stage of the development, it is worth spending time on identifying the concepts rather than the relationships between them. We should concentrate on the *has to know* relationships. For example, from the glossary, we can state that an order *has to know* the related customer and the products being ordered (see Fig. 7).

Thus, some of the roles detected in the business model, and therefore specified in the role model, could be included as a class in the conceptual model. This is the case of the class *Customer* in our example.

Starting from the business model, it is also possible to identify some classes whose behavior depends on a rich set of reachable states. In this case, it could be of interest to define a state machine for them, represented by means of a UML *statechart diagram*. These classes are easily detected in the process diagrams, since they correspond to information objects labeled with several states. In our running example, *Order* would be a candidate for building a state machine that shows the states of an order (*proposed, under_review, reviewed, accepted* and *rejected*) and the events to change from one state to another.

6 Conclusions

In this paper, we have presented an approach for business modeling and requirements analysis, in which the use cases and the conceptual model are obtained in a straightforward using the business model as a starting point. Business modeling is centered on the use of UML activity diagrams.

With this guide, the modeler has available a systematic way to identify and organize use cases, and to identify and define the classes of the conceptual model. The business processes of the organization are identified from the goals proposed by Cockburn [3], and they are described by means of a flow of activities represented by a UML activity diagram. In this manner, system use cases are obtained from the activities of the business processes and they are organized into a hierarchy of levels, as proposed by Korson [7].

The classes of the conceptual model are obtained from the information objects flowing among the activities. We would like to highlight as an important feature of our approach that use cases are modeled at the same time that the conceptual modeling is done, in agreement with Korson [8], who states that this is crucial to get correct use cases, since understanding the domain is necessary to write useful use cases.

During the business and requirements modeling, the activities specification and the associated use cases, as well as the information objects and the corresponding domain classes, are gathered in a single glossary, which allows us to keep traceability between the different modeling artifacts.

In the Rational Unified Process (RUP) [6], defined by Rational for UML, business modeling is also included as a step within the iterations making up the process model. Jacobson et al. [6] present some steps that are similar to ours, but the hierarchical decomposition of the highest-level use cases is not considered, nor is a clear guide to discover the system use cases provided. Our approach to business modeling is a complete guide as opposed to the general sketch presented there.

References

1. Booch, G., Rumbaugh, J., Jacobson, I.: The Unified Modeling Language User Guide. Addison-Wesley (1999)
2. Ceri, S., Fraternalli, P.: Designing Database Applications With Objects and Rules. The IDEA Methodology. Addison-Wesley (1997)
3. Cockburn, A.: Using Goal-Based Use Cases" JOOP vol. 10 no.7. (Nov/Dec 1997) 56-62
4. Coleman, D.: A Use Case Template: Draft for discussion (1998) `http://www.bredemeyer.com/use_case.pdf`
5. Integration Definition for Function Modeling. Computer Systems Laboratory, National Institute of Standards and Technology, FIPS Pub. 183. (December 1993)
6. Jacobson, I., Booch, G. Rumbaugh, J.: The Unified Software Development Process. Addison-Wesley Longman, Inc. (1999)
7. Korson, T.: Misuse of Use Cases. (1998) `http://software-architects.com/publications/korson/korson9803om.htm.`
8. Korson, T.: Constructing Useful Use Cases (1999) `http://software-architects.com/publications/korson/usecase3`
9. Martin, J. Odell, J.J.: Object-Oriented Methods: A Foundation. Prentice Hall. (1997)
10. Ortín, M.J., García Molina, J., Martínez, A., Pellicer, A.: Combining OOram and IDEA for Information Systems Modeling. Technical Report TR-01-00. (December 1998)
11. Ortín, M.J., García Molina, J.: Role-Based Modeling with UML. IV Jornadas de Ingeniería del Software y Bases de Datos. Cáceres, Spain (1999)
12. Reenskaug, T.: Working with Objects: the OOram Software Engineering Method. Addison-Wesley / Manning Publications (1996)
13. Reenskaug, T.: Working with Objects: a Three-Model Architecture for the Analysis of Information Systems. JOOP vol. 10 no. 2 (May 1997) 22-30
14. Whitenack, B.: RAPPeL: A Requirements Analysis Process Pattern Language for Object-Oriented Development. In: Coplien, J.O., Schmidt, D.C. (eds.): Pattern Languages of Program Design. Addison-Wesley (1995) 259-291