Extending UML to Specify and Verify E-commerce Systems.

Conference Paper · January 2003

Source: DBLP

CITATION
1

READS
262

6 authors, including:

Mark A. J. Song
Pontifícia Universidade Católica de Minas Gerais
66 PUBLICATIONS **166** CITATIONS
SEE PROFILE

Adriano C. M. Pereira
Federal University of Minas Gerais
161 PUBLICATIONS **1,983** CITATIONS
SEE PROFILE

Sergio Campos
Federal University of Minas Gerais
123 PUBLICATIONS **2,911** CITATIONS
SEE PROFILE

Wagner Meira Jr.
Federal University of Minas Gerais
584 PUBLICATIONS **10,249** CITATIONS
SEE PROFILE

Extending UML to Specify and Verify E-commerce Systems

M. Song, A. Pereira, F. Lima, G. Gorgulho, S. Campos, W. Meira Jr.
Department of Computer Science - Federal University of Minas Gerais
Av. Antônio Carlos, 6627 - ICEx - room 4010 - Pampulha - CEP 31270-010
Belo Horizonte - Minas Gerais - Brazil
Phone: (55-31) 3499-5860 Fax: (55-31) 3499-5858
{song, adrianoc, fernanda, gorgulho, scampos, meira}@dcc.ufmg.br

Abstract

There are many software engineering processes, but they are usually generic, not focused in e-commerce systems. E-commerce applications are complex and difficult to be correctly designed. Currently, most approaches are ad-hoc, *and frequently lead to expensive and unreliable systems. Moreover, guaranteeing the correctness of an e-commerce application is not an easy task and it is quite hard and laborious if only tests and simulations, common techniques of system validation, are used. A solution to this problem is the adoption of formal methods.* Formal-CAFE *is a methodology that uses formal-method techniques to specifically design electronic commerce applications. It is used to model e-commerce systems and identify errors in earlier stages, when it is cheaper and easier to correct them. Nevertheless, it is not a simple task to apply this methodology since it demands specific knowledge of formal methods. To overcome this obstacle we propose a new process, based on this methodology, an unified modeling language (UML) and a set of design patterns to specify and automatically verify e-commerce systems.*

1. Introduction

Electronic Commerce (e-commerce) has become a popular topic for business and academic research since the early 1990's. In this context, different researchers work on different areas: applications, services, marketing, strategy, the Internet, extranets, technologies in e-commerce, and so on. E-commerce has changed the ways organizations perform their activities. It makes easy the access to goods and services and has made a revolution in the economy as a whole. However, e-commerce applications tend to generate complex systems. As new services are created, new kinds of errors appear, some unacceptable. Actually exists a consensus that the occurrence of errors in the sites is a barrier to the growth of e-commerce because it can cause damages for the users and for the site, depending on its nature.

New methodologies can be used in order to improve the quality of the software/hardware and to guarantee the integrity of critical systems. One such approach, named Formal Methods, consists of the use of mathematical techniques to assist in the documentation, specification, design, analysis and certification of computational systems.

The main objective of this work is to present a process based on a formal method technique (model checking [1]), a methodology (the Formal-CAFE Methodology [10]), and a standard notation (the Unified Modeling Language - UML [4]), to design and verify properties of e-commerce systems. Although UML in its current state is found useful by modeling practices, it has to be improved to describe further concurrent aspects of the systems such as isolation. So we decide to extend UML to be capable to model and produce robust e-commerce systems.

This paper is organized as follows. The Section 2 defines some important concepts about model checking. Section 3 gives an overview of the Formal-CAFE methodology and introduces our model checking patterns. Section 4 analyzes the related works, and Section 5 presents our Process. Section 6 illustrates the application of our process through a case study, and Section 7 presents some conclusions and ongoing work.

2. Model Checking

In this section we present a brief background on model checking [1]. Applying model checking to design computational systems consists of several task, that

can be classified in three main steps, as follows:
Modeling: converts a design into a formalism accepted by a model checking tool.
Specification: states the properties that the design must satisfy. The specification is usually given in some logical formalism, such as CTL [1].
Verification: ideally the verification is completely automatic. However, in practice it often involves human assistance. One such manual activity is the analysis of the verification results.

The system being verified is represented as a *state-transition graph* (the model) and the *properties* (the behaviours) are depicted as formulas in some temporal logic. Formally, the model is a labeled state-transition graph. The labels correspond to the values of the variables in the program, while the transitions correspond to the passage of time in the model.

In the next section we briefly describe the Formal-CAFE methodology and the model checking pattern system used in our process, the UML-CAFE.

3. The Formal-CAFE Methodology

The *Formal-CAFE* methodology [10] consists of a way to design e-commerce systems applying model checking. This methodology explains how to specify an e-commerce system, but it considers that the user knows some formal language, such as SMV [7], to build the model.

The methodology is incremental and divided into four levels. The first level, defined as conceptual, embodies the business rules (norms that specify some functioning of e-commerce applications) and the definition of the e-commerce system to be designed. The second level, called application, models the life cycle of the item that is negotiated, identifying the types of operations, called actions, that are performed by agents, changing item's state. The third one, named functional, models the services provided by the system. The last level contemplates the components of the system and the user's interaction with them. It completes the scope of the system, modeling its architecture, so it is called the execution or architectural level. Further details can be obtained in [8].

3.1 Properties of an E-Commerce System

The Formal-CAFE methodology describes properties as formulas in CTL. CTL-formulas are built from atomic propositions, boolean connectives and temporal operators. For example, a rule can specify that an item can only be reserved if it is available. To specify this property, a developer would have to translate this informal requirement into the following CTL-formula: AG (((item_state = Available) & (service = Reserve) & (product_inventory > 0) & (next(product_inventory) = product_inventory - 1)) → AX ((item_state = Reserved))).

As it is usually difficult to read and understand formulas written in CTL, and even more difficult to write them correctly without the knowledge of the syntax of the specification language, it was proposed a specification pattern system to overcome such problems. It was defined a set of patterns to describe properties such as completeness, invariant, and properties relate to transactions.

Such patterns are used to map business rules to a formal verification language. Some properties are useful to build the application's model and others to validate its behaviours. Details on model checking patterns can be obtained in [12].

4. Related Work

There are many software engineering processes, such as Rational Unified Process (RUP [5]) and Enterprise Unified Process (EUP [6]), but these processes are generic, not focused in e-commerce systems. Moreover, guaranteeing the correctness of an e-commerce application is not an easy task and it is quite hard and laborious if only tests and simulations, common techniques of system validation, are used. To design robust e-commerce systems, we conceive a new process, based on RUP, that uses formal methods.

However it is not trivial to apply formal methods since it demands specific knowledge. Most software engineers adopts a high level modeling language, such as UML, to general-purpose software design. Nevertheless, there are application, such as embedded, real-time and e-business, that tends to be highly event-driven, concurrent, and often distributed, not fully represented by UML.

In order to solve this problem, there are other modeling languages, such as UML-RT(an UML extension [3]) and ROOM [11]. Our experience pointed out that these languages are focused in real-time systems and do not fit some e-commerce requirements. Stringent requirements must be met for atomicity, isolation, and so on [12]. To describe them we propose an extension to UML-RT. Actually we define an innovative process to produce an accurate software design, that aggregates concepts of formal methods, model checking patterns and this modeling language extension.

5. The UML-CAFE Process

In this section we describe the UML-CAFE, a process to design and verify properties of e-commerce systems. It is based on formal method techniques, the Formal-CAFE methodology [9], and a standard notation, the Unified Modeling Language (UML).

UML [2] is a language which can be used to visualize, to specify, and to document object oriented systems. It is a standard modeling language used in the software development process and visual modeling which provides a smooth transition between the business domain and the computer domain. We will describe each UML component as they appear in our process.

5.1 The Process

A successful development project satisfies or exceeds the customer's expectation, is developed in a timely and economical fashion, and is resilient to change and adaptation. The development, in general, proceeds as a series of iterations that evolve into the final system. Each iteration consists of one or more of the following process components: requirements capture, analysis, design, implementation, and test.

The UML-CAFE Process is an incremental process which can be used to design more reliable e-commerce applications. It is divided into four phases: conceptual, application, functional, and execution. Each of them is subdivided into stages in order to help developers to manage each step during the system's design.

5.1.1 The Conceptual Phase

The first phase which captures the requirements of the system is divided into three stages:

Stage 1: In the first stage, the system is described as a set of business rules. Remember that a business rule is a norm specifying some functioning of an e-commerce application. Rules can describe situations such as: if there are two items available in a virtual store and two buyers reserve these items simultaneously then the inventory must remains zero.

Stage 2: Based on the business rules presented in stage 1, the designer has to describe the actors, their actions, the negotiated item and its states. Each actor represents an agent of the Formal-CAFE methodology.

Stage 3: Once the actors and the main negotiated object were identified, the system's behaviours should be described. In UML, this behaviours are represented by use cases which illustrate the system's functionalities and relationships between the use cases and actors. The collection of use cases for a system constitutes the context diagram.

5.1.2 The Application Phase

The second phase defines the system's behaviour, its interactions with the actors, actions, and negotiated object. Moreover, the states of the system are modeled, the system's functionality is described and properties, such as completeness, are verified. At this phase, all elements are modeled through use cases, as illustrated by the following stages:

Stage 4: Each use case is documented with a flow of events required to accomplish its behaviours. A flow of events is a sequence of transactions, or events, performed by the system. It should contain detailed information written in terms of what the system should do, regardless of how the system accomplishes the task. Table 1 describes the UML-CAFE template to create the flow of events. The template is based on RUP Process, and is used to model information presented in the application phase. In this stage only the preconditions and the flow of events are depicted for each use case, letting the statechart diagram to the next stage.

Table 1. The UML-CAFE Template

Element	Description
<Name>	Use Case Name
Preconditions	Conditions that must be obeyed to allow the execution of the use case
Main Flow	Normal sequence of events for the use case
Alternative Flows	Alternate or exceptional flows
Statechart	Diagram that represents the sequence of states that the system goes through
Observations	Additional information about the use cases

Stage 5: Here, the statechart diagram is used to model the discrete stages of a system's lifetime. Statechart diagram shows the sequence of states that the system goes through, the events that cause a transition from one state to another, and the actions that result from a state change. It typically contains one start state and one or more final states, and transitions connecting the various states on the diagram. Each state represents a snapshot during the life of a system which satisfies some condition or waits for some event. The statechart is related to the item's life cycle as specified in the application level of Formal-CAFE methodology. Transitions in the diagram is represented by events which indicate actions executed by an agent, as

defined in this methodology. There is also a guard condition that must be met before the transition is taken. As long as the guard condition remains false, the transition will not occur.

Stage 6: In this stage the developer reviews in detail the precedent stages and collects, for each use case, additional information using the observation field of the template.

5.1.3 The Functional Phase

This phase models the services provided by the system. It uses the class diagram, interaction diagram and activity diagram to model features, such as concurrency and synchronization. This phase is divided into four stages, explained as follow.

Stage 7: In this stage the class diagram is built, defining the classes, methods, attributes and relationships. Class diagrams show the static structure of the model, in particular, classes and types, their internal structure, and their relationships to other classes.

Stage 8: Here, a set of services is defined for the use cases previously identified. Each service consists of a sequence of actions. The designer identify all sequences that must be executed isolated or as an atomic transaction. Note that although actions are atomic by definition, not every sequence of actions is atomic. So, in this stage it is described the services and their concurrent aspects.

Stage 9: Once the class diagram and services are defined, the designer describes the interaction among instances. In UML interaction come in two forms: collaboration diagrams and sequence diagrams. Collaboration diagrams show the relationships among instances and are better for understanding all the effects for procedural design. Sequence diagrams depict interactions arranged in time sequence and the explicit sequence of stimuli. In this phase we use them and introduce some extensions in order to describe concurrency. For example, an event flag is used to describe an atomic block.

Stage 10: Our experience pointed out that some concurrent activities can not be fully described by sequence diagrams. To overcome this obstacle, it was decided to extend UML, adding new features to the activity diagram. An activity diagram is a variation of a state machine in which the states represent the performance of activities and the transitions are trigged by their completion. Its purpose is to focus on flows driven by events. Again, some extensions are introduced in order to describe concurrency. For example, a semaphore is used to isolate conflicting transactions.

5.1.4 The Execution Phase

The last phase, named execution, contemplates the components of the system and the user's interaction with them. It completes the scope of the system. To describe it we use the UML component or deployment diagrams. The component diagram models the development view of a system's components and their relationships. The deployment diagram shows the organization of the hardware and the binding of the software to the physical devices.

This phase finalizes the description of our process. In the next section we illustrate the use of the UML-CAFE Process in practice.

6. Applying UML-CAFE Process

In this section we illustrate our process through a buyer group application. This is a typical e-commerce application where a group is created to aggregate similar demands in order to buy goods at low cost. The first step is to publish the buyer group, enabling buyers to participate of it. The buyer group is published by an Association Administrator agent. In this example, buyers join a group only once and should inform the quantity and maximum value each of them is willing to pay for the good being traded.

Once the group is published if the deadline to join a group is met and there is no adhesion, then the group should be cancelled; otherwise the group changes to a confirmation phase. Once the group is in confirmation stage, the association administrator can cancel or confirm the group. If the group is confirmed the group changes to the negotiation phase where the administrator chooses the negotiation modality and waits for sellers send their proposals.

If the deadline to receive proposals is achieved and none of the proposals is the winner, then the group is cancelled; otherwise the group changes to a confirm adhesion phase. Each adhesion is checked against the winner proposal. For each adhesion, if the winner proposal price is higher than the maximum value informed in the adhesion, then this adhesion is get confirmed, otherwise it is cancelled.

As soon as the deadline to confirm adhesion is achieved and at least one adhesion is confirmed, the buyer group changes to confirm proposal stage. If the seller confirms its proposal, the buyer group goes to a closed state; otherwise it is cancelled. Once the group is closed it should remain closed.

In the next subsections we illustrate each phase of UML-CAFE in practice.

6.1 The Conceptual Phase

The first step in the design, as defined by our process in stage 1, is to describe the system as a set of business rules. Following, we present some of them:

1. The minimum quantity in an adhesion must be greater than zero.
2. The maximum value specified in an adhesion must be greater than zero.
3. If the buyer group is in the *Adhesion* state and the buyer agent makes an adhesion to this buyer group, then the buyer group will continue in the *Adhesion* state.
4. If the buyer group is in the *Adhesion* state and the deadline to make adhesion is achieved and there is at least one adhesion, then buyer group will change to *Confirmation* state.
5. If the buyer group is in the *Adhesion* state and the deadline to make adhesion is achieved and there is no adhesion, then buyer group will change to *Cancelled* state.
6. Once the buyer group is cancelled it must remain in this state.

The next step, according to stage 2, is the identification of actors and their actions as shown in Table 2. The object being negotiated and its states are defined as illustrated in Table 3.

Table 2. Actors and Actions

Actor	Action
Automatic Agent	cancel group, confirm group, process adhesions, process proposal, generate orders
Administrator	create buyer group, confirm buyer group, cancel buyer group, choose modality of negotiation
Seller	create proposal, confirm proposal, cancel proposal
Buyer	join group, confirm adhesion, cancel adhesion

Based on the information described, the designer is able to define the behaviours of the system under development. At this point the use cases for the system are identified. Figure 1 shows the buyer group context diagram specified in stage 3.

Table 3. Negotiated Item and its States

Item	State
Buyer Group	Unpublished, Adhesion, Confirmation, Confirmed, Cancelled, Negotiation, Closed, Confirm Adhesion, Confirm Proposal

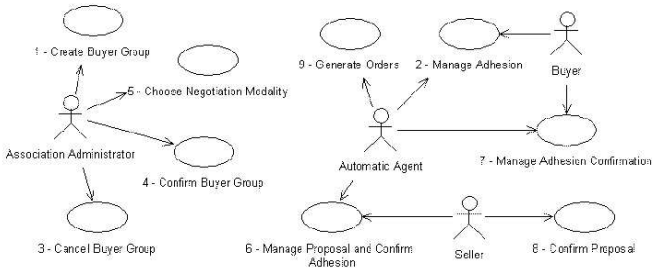

Figure 1. Context Diagram

6.2 The Application Phase

This second phase defines the system's behaviour, its interactions with the actors, actions, and negotiated object. In stage 4, each use case is documented. Considering the first five presented business rules, the designer could generate the *Manage Adhesion Use Case*. A full description applying the UML-CAFE template is illustrated as follow:

1. Manage Adhesion Use Case
2. Preconditions: (group state = adhesion)
3. Main Flow:

   if (agent=buyer and action=join and adhesion_quantity>0 and max_value>0 and not adhesion deadline)
   then (state=adhesion)
4. Alternative Flows:
   - Preconditions: Adhesion deadline
   - Steps:

     if (agent=automatic and action=cancel group and !has_adhesion)
     then (state=cancelled)

     if (agent=automatic and action=confirm and has_adhesion)
     then (state=confirmation)

     if (agent=buyer and action=join)
     then adhesion not accomplished

5. Statechart Diagram: as defined in stage 5 the item's life cycle related to the use case is specified as a state transition diagram (Figure 2).

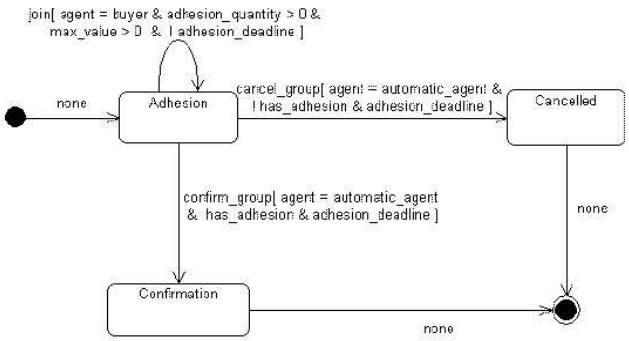

Figure 2. Manage Adhesion Diagram

6. Observations: In stage 6 after reviewing the use cases, details such as *Cancelled group is a final state* are registered.

Note that transitions in the diagram is represented by events, such as cancel group, that indicate actions executed by the agent. There are also guard conditions, such as $(agent = automatic\_agent \ \& \ !has\_adhesion \ \& \ adhesion\_deadline)$, which define a necessary condition to change the state of the system when the correspondent event happens.

6.3 The Functional Phase

This phase models the services provided by the system. First, as described in stage 7, the class diagram is built as shown in Figure 3.

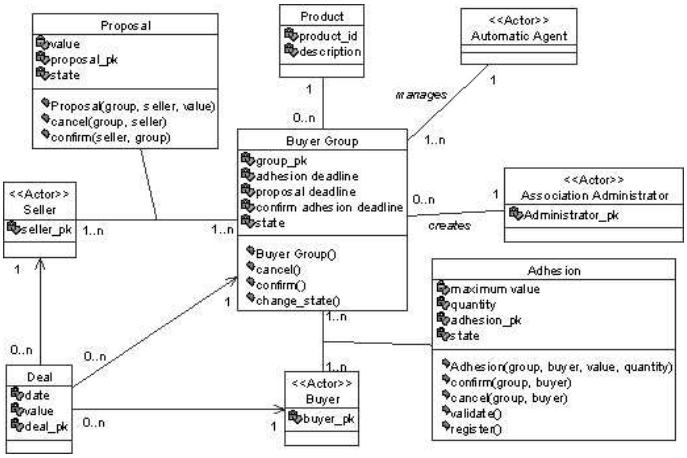

Figure 3. The Class Diagram

The next step, according to stage 8, is the definition of services for the use cases. For example, the following service is defined to the Manage Adhesion Use Case:

- Receive buyer adhesion; Reject adhesion, if deadline for adhesion is achieved; The automatic agent manages deadline for adhesion;
- Atomic actions: Validate adhesion; Register valid adhesion;

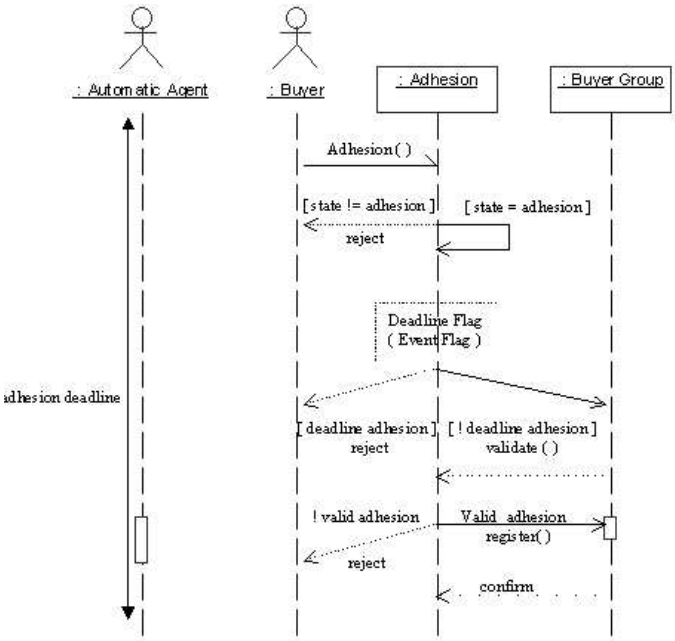

Figure 4. The Sequence Diagram

Note that an atomic sequence is identified: once adhesion is received on time and is a valid one, it must be registered. Now, as defined in stage 9 the service is depicted in a sequence diagram as shown in Figure 4. We introduced some extensions in the diagram. An event flag, such as deadline flag, is used to represent an atomic block. If a deadline for adhesion is achieved any adhesion is reject; otherwise it should be validated and, if it is a valid adhesion, it must be registered no matter if the deadline for adhesion occurs - these two actions form an atomic transaction.

Not all concurrent activities can be totally depicted by a sequence diagram. In the example above, a deadline can occurs at any time while the buyer is making an adhesion to the group. To describe such fact we use the activity diagram, as defined in stage 10 of our process. As depicted in Figure 5, actions executed by the automatic agent and any adhesion instances are concurrent. Also note that both actions, validate adhesion

and register adhesion, are treated as an atomic block as defined in the sequence diagram.

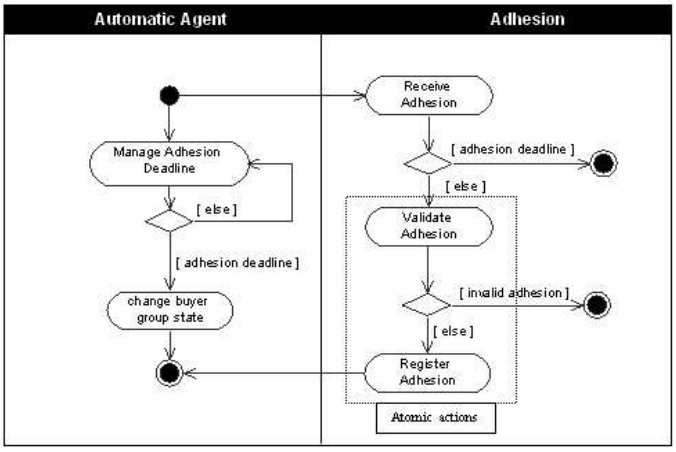

Figure 5. Activity Diagram Revised

In this stage another important aspect of services should be captured and described: the isolation of conflicting services. Consider a virtual store where the following service is implemented: a buyer selects an item; the system validate the selection (item must be available); if the selection is valid, then the inventory should be changed; and finally the item, reserved. Now, consider two buyers reserving the same item. It is evident that, not only the validation of the selected item and the changing in the inventory must be treated as an atomic block, but also they can not be concurrently executed (if two or more buyers, for example, are reserving the same item - this problem is described in [10]). To overcome such problem it is introduced a new element in the activity diagram: a synchronization component. Figure 6 shows its use to control two conflicting concurrent activities.

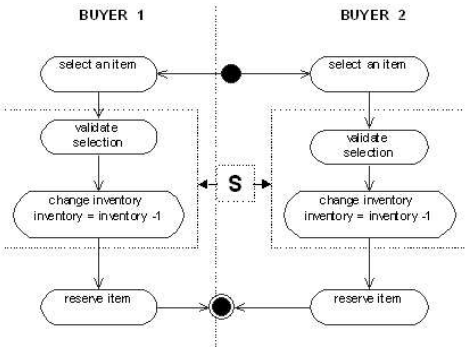

Figure 6. Two Conflicting Sequences

6.4 The Execution Phase

In this phase we model the components of our e-commerce application: the client servers, the web server, the transaction server, and the database server. This phase is important because the architecture of the system is defined, but we prefer not to detail it as we consider much more relevant to present part of the verification model and some verified properties.

So we present here the NuSMV code generated according to UML-CAFE Process. It is related to the business rules, the Formal-CAFE methodology, the model checking patterns, and templates describing each use case of the system under construction. Based on table 2 the following NuSMV processes are defined:

MODULE main
var
– buyer agent process
buyer: process buyer_agent(1);
...
– buyer group process
bgwf: process buyerGroup(buyer.action, ...);
...
MODULE auto_agent(id)
var
action: cancel_group, ..., none ;
...
According to table 3, the following NuSMV module describing the item's life cycle is implemented:

MODULE buyerGroup(ba_action, ..., auto_action)
var
state : Unpublished, Adhesion, ..., Closed ;
has_adhesion : boolean;
...
ASSIGN
init(state) := Unpublished;
init(has_adhesion) := FALSE;
...
– Life cycle Pattern: AG(q − > AX(p))
next(state) := *case*
...
— Manage Adhesion Use Case
state=Adhesion & ba_action=join & ba_quant>0
& ba_maxval>0 & !adhesion_deadline:*Adhesion*;
state=Adhesion & auto_action=cancel_group &
!has_adhesion & adhesion_deadline: *Cancelled*;
...
1: state;
esac;
— Completeness Pattern: EF(STATE = < *A* >)
SPEC EF(state = Unpublished)

...
SPEC EF(state = Confirm_Proposal)
— end buyer group module

Note that we use the model checking patterns and use cases to construct the item's life cycle. Also, note that some patterns are used to check important properties of the system as the completeness property above or the consistence and invariant properties below:

- Consistence Model Checking Pattern: AG(p→q)
AG ((state = Closed) → AX(state = Closed))
AG((state = Cancelled)→AX(state = Cancelled))

– Invariant Model Checking Pattern: AG(p)
AG (buyer.adhesion_quantity > 0);
AG (buyer.max_value > 0);

The next section presents our conclusion.

7. Conclusion

The currently adopted methods for design validation of most e-commerce applications are still the traditional techniques of simulation and testing. Although effective in the very early stages of debugging, when the design is still infested with multiple bugs, their effectiveness drops quickly as the design becomes cleaner. A very attractive alternative to simulation and testing is the approach of model checking.

In this paper we propose a process, based on RUP, Formal-CAFE methodology [10], and a standard notation (UML), to design and verify e-commerce systems. This process, denoted UML-CAFE, tends to increase the efficiency of the design of electronic commerce applications using a notation commonly adopted by software engineers. It uses formal methods to formalize the specification of the system, through a UML notation, and also to automatically verify requirements that it must satisfy. We have modeled and verified a buyer group, an interesting e-commerce application, to demonstrate how the process works. We were able to construct the model through our model checking patterns and check important properties of this system such as completeness, invariant, and consistence.

References

[1] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking.* The MIT Press, Cambridge, Massachusetts, 1999.

[2] Rational Company. Uml resource center. http://www.rational.com, 2003.

[3] R. Grosu, M. Broy, B. Selic, and G. Stefanescu. *Behavioral specifications of businesses and systems - Chapter 6: What is Behind UML-RT?* Kluwer Academic Publishers, 1999.

[4] Object Management Group. Uml resource page. http://www.omg.org/uml, 2003.

[5] Wolfgang Hesse. RUP: A process model for working with UML. In Keng Siau and Terry Halpin, editors, *Unified Modeling Language: Systems Analysis, Design and Development Issues*, chapter 4, pages 61–74. Idea Publishing Group, 2001.

[6] Ivar Jacobson, Grady Booch, and James Rumbaugh. *The Unified Software Development Process.* Addison-Wesley, 1999.

[7] K.L. McMillan. *Symbolic Model Checking: An Approach to the State Explosion Problem.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 1992.

[8] A. Pereira. A Model Checking Methodology to E-commerce Systems. Master's thesis, Department of Computer Science – Federal University of Minas Gerais (DCC/UFMG), 2002.

[9] A. Pereira, M. Song, G. Gorgulho, W. Meira Jr., and S. Campos. Formal-cafe methodology: an e-commerce system's case study. In *Proceedings of the Fifth International Conference on Electronic Commerce Research (ICECR-5)*, Montreal, Canada, October 2002.

[10] A. Pereira, M. Song, G. Gorgulho, W. Meira Jr., and S. Campos. A formal methodology to specify e-commerce systems. In *Proceedings of the 4th International Conference on Formal Engineering Methods*, Lecture Notes in Computer Science, Shanghai, China, October 2002. Springer-Verlag.

[11] Bran Selic, Garth Gullekson, and Paul T. Ward. *Real-time Object-oriented Modeling.* Wiley Professional Computing. John Wiley & Sons, Inc., New York, 1994.

[12] M. Song, A.Pereira, G. Gorgulho, W. Meira Jr., and S. Campos. Model checking patterns for e-commerce systems. In *Proceedings of the First Seminar on Advanced Research in Electronic Business*, Lecture Notes in Computer Science, Rio de Janeiro, RJ, Brazil, November 2002.