An Experimental Validation of the ADORA Language

Stefan Berner, Nancy Schett, Yong Xia, Martin Glinz

Technical Report 99-07
Institut für Informatik, Universität Zurich
Winterthurerstrasse 190
CH-8057 Zurich, Switzerland
+41-1-63 54570
http://www.ifi.unizh.ch/~glinz

Abstract

ADORA (Analysis and Description of Requirements and Architecture) is an approach to object oriented modeling that is based on object modeling and hierarchical decomposition, using an integrated model. The ADORA language is intended to be used for requirements specifications and high-level, logical views of software architectures.

In order to assess the comprehensibility and the appeal of specifications written in ADORA we conducted a controlled experiment with students. We wrote two specifications of the same problem in ADORA and in UML and let the students study them. Then we tested the students' comprehension by asking questions about the contents of the specification and tested how they liked ADORA in comparison to UML by asking questions about personal preferences.

In this report we describe the experiment and report the results.

1. Introduction

ADORA (Analysis and Description of Requirements and Architecture) is an approach to object-oriented modeling that is based on object modeling and hierarchical decomposition, using an integrated model. The ADORA language is intended to be used primarily for requirements specifications and also for logical-level architectural design.

The principal ideas of ADORA are listed as follows.

- **Using an integrated model.** Unlike UML, in which a collection of different models with nearly no semantics in between is used to specify a whole system, an integrated model is adopted in ADORA. This makes it possible for us to check the consistency of the whole system**.**
- **The system being modeled with hierarchical decomposition**. Decomposition ensures large specification being manageable and comprehensible. ADORA uses abstract, prototypical objects, instead of classes, as the core of the ADORA model. This is a most distinctive feature of the ADORA model, which allows that objects can be recursively decomposed into objects (or elements that may be part of an object, like states). Therefore, the full power of object modeling on all levels of the hierarchy can be exploited with this decomposition mechanism, and only the degree of abstractness is varied: objects on lower

levels of the decomposition model small parts of a system in detail, whereas objects on higher levels model large parts or the whole system on an abstract level.

- **Visualizing models in context**. The integrated ADORA model is visualized by presenting details of a model always with an abstraction of its surrounding context. Hierarchical structures can be viewed at any level of detail. A fisheye view concept [1] is used to realize these features.
- **Expressing different parts of a specification with varying degree of formality.** The ADORA language contains elements that - together with an open and flexible modeling process - allow tailoring the formality of ADORA models to the problem at hand.

The detailed introduction to ADORA is given in [2,3].

2. Goals of the experiment

In our opinion, there are two fundamental qualities that a specification language should have:

- When people set up a model using a language (or a set of notation), the language should help users to interpret the meanings of that model correctly.
- The users must like it. In particular, the language must be easy to read (a specification has much more readers than writers).

Therefore, we set up our experiment with the following goals.

(i) Determine the *comprehensibility* of an ADORA specification both on its own and in comparison with an equivalent specification written in UML – today's standard modeling language – from the viewpoint of a reader of the specification.

(ii) Determine the *acceptance* of the fundamental concepts of ADORA (using abstract objects, hierarchical decomposition, integrated model...) both on its own and in comparison with UML from the viewpoint of a reader/writer of models.

3. Setup of the experiment

The basic idea of the experiment was

- to write two specifications of the same problem, one in ADORA and one in UML
- let students read these specifications
- test the comprehensibility and acceptance of the two specification languages by asking questions (a) about the contents of the specifications and (b) about the personal impressions and preferences of the students.

3.1 Preparation of the experiment.

As a sample application we chose a distributed ticketing system. The system consists of geographically distributed vending stations (POS) where users can buy tickets for events (concerts, films, musicals...) that are being offered on several event servers. The vending stations and the event servers shall be connected by an existing network. From another project, we had a detailed specification of this system written in natural language (in German) [5].

Because of the limited experiment time, we only wrote partial specifications of the ticketing system in ADORA and in UML. We also prepared two introductory tutorials on UML and on ADORA.

The UML specification was written by a new research assistant who was familiar with UML, but had nearly no knowledge of ADORA before. The UML tutorial was written by an experienced research assistant, who works on another project of our group which has few relations with the ADORA project. The ADORA specification and tutorial were written jointly by a research assistant who had worked in the ADORA project for several years and a graduate student.

Both teams jointly prepared the questionnaires. The two teams worked separately and independently at most of the time. At the end of modeling the system, they collaborated in order to get two models nearly equal in semantics (We purposely made some small differences in the two models and asked the participants in the questionnaires to tell the differences).

The questionnaire consisted of two parts. The original questionnaire was written in German. In Appendix A4 both the original German version as well as an English translation are given. In the first part of the questionnaire, the “objective” one, we aimed at measuring the comprehensibility of an ADORA model. We created 30 questions about the contents of the specification to test whether the participants understood the execution of the Ticketing System correctly. 25 questions were yes/no questions; the rest were open questions. We also prepared a sheet with the correct answers for all questions.

For every question, we additionally asked

- whether the answering person was sure or unsure about her or his answer;
- how difficult it was to answer the question (effort: easy, moderate, difficult, impossible).

In the second part, the “subjective” one, we tested the acceptance of ADORA vs. UML. We asked 14 questions about the personal opinion of the answering person concerning distinctive features of both ADORA and UML.

3.2 Participants

We ran the experiment with fifteen persons who were not members of our research group. Most participants were Diploma students[1] or Ph.D. students from our department. A few participants came from industry. We ensured that all participants had sufficient fundamental knowledge in Computer Science and particularly in software specification.

The members of our research group had nearly no personal contacts with any of the participants.

Most participants had some knowledge in UML. None of them had been exposed to ADORA before.

3.3 Process of the experiment

Figure 1 shows our experiment process.

[1] These students are at the level of M.Sc. students.

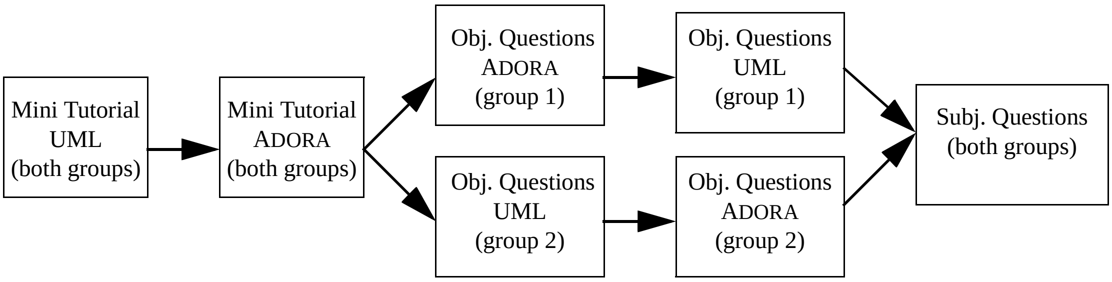


Figure 1. The process of the experiment

The participants were first given an introduction both to ADORA and to UML.

According to the experiment process, we gave them one hour training on UML and one hour training on ADORA.

Then the participants worked in two groups. The members of Group 1 answered the objective part of the questionnaire first for the ADORA specification and then for the UML specification; Group 2 members did it vice-versa. Finally, both groups answered the subjective part of the questionnaire.

We did not assign participants to a certain group but asked the participants to form two groups (equal or almost equal in size).

While participants were working on the questionnaires, only questions concerning syntax/ semantics of language elements or understanding the questions/questionnaires were answered by the members of our group. The participants were also strongly advised to refer only to the given specification when trying to answer a question. The students were asked *not* to sign their name on the questionnaires.

At the end of experiment, they were asked to hand in their questionnaires to us, after they finished the work.

3.4 Validity of the experiment

Fairness is the major concern of this experiment, because any bias to our own specification language, ADORA, would make the whole experiment meaningless. Therefore, we took the following measures to avoid bias towards ADORA and to make the experiment fair.

1. *Selecting an example system which had not been used in the ADORA project before*. The ticketing system was first used in another project "Simulation Tools for Requirement Engineering" which has no direct relation with the ADORA project.
2. *Equal and good quality of the models both in ADORA and in UML for that system.* As we described above, we divided our group into two teams. The UML team consisted of research assistants who were familiar with UML and were not directly involved in the ADORA project. This team wrote the UML specification of the ticketing system and prepared the UML tutorial.
3. *Fairness of the Questionnaires.* During the preparation, our research group members had been strongly advised to make the questionnaires neutral. It was the team working on the UML part that contributed to most of questions in the questionnaires.

4. *Participants with the knowledge of computer science, but no ideas on ADORA.* The ADORA project is still in the research phase, and has never been disseminated to the participants (mainly our students) in any software engineering courses.
5. *Equal training for the both specification languages (UML and ADORA).* As mentioned in the experiment process, the participants got the same amount of training both on UML and ADORA (1 hour for each).
6. *Two randomly divided groups, working on opposite sequences (UML/ADORA and ADORA/UML).* We ensured this according to our experiment process (c.f. Section 3.3).
7. *Anonymity of the filled questionnaires.* This was fulfilled in our experiment process (c.f. Section 3.3).

Through these efforts, we think we have adopted a neutral stance in the experiment.

4. Results

Two participants did not finish the experiment; another person's answers could not be scored because his answers revealed insufficient basic knowledge of object technology. So we finally had twelve complete sets of answers.

The results are presented in the following diagrams and tables.

4.1 Evaluation of the "objective" questionnaire

Table 1 summarizes the result of the evaluation of the "objective" questionnaire for the two groups. As the differences between Groups 1 and 2 are marginal, we consolidated the results of the two groups. The consolidated figures are also shown in Table 1.

For each model, we should have a total of 360 answers (30 questions times 12 participants). For every answer, we determined whether the answer was objectively right or wrong according to our answer sheet. The answers were further subdivided into those where the answering person was sure about her or his answer and those where she or he was not (the confidences of the participants on answering questions). Those values again are subdivided, indicating how difficult it was to answer the questions in the participants' opinion (the efforts of the participants on answering question). For some questions, as participants forgot to give the ranks of confidence or effort, we excludes those answer from our final statistical data.

			easy	moderate	difficult	impossible (guess)		
Group 1 - ADORA	right	sure	50 (34.5%)	48 (33.1%)	14 (9.7%)	0 (0%)	112 (77.2%)	132 (91.0%)
		unsure	0 (0%)	6 (4.1%)	14 (9.7%)	0 (0%)	20 (13.8%)	
	wrong	sure	1 (0.7%)	0 (0%)	3 (2.1%)	0 (0%)	4 (2.8%)	13 (9.0%)
		unsure	3 (2.1%)	6 (4.1%)	0 (0%)	0 (0%)	9 (6.2%)	

Table 1: Evaluation results of "objective" questionnaire

			easy	moderate	difficult	impossible (guess)		
Group 1 - UML	right	sure	66 (43.4%)	28 (18.4%)	6 (3.9%)	0 (0%)	100 (65.8%)	131 (86.2%)
		unsure	3 (2.0%)	6 (3.9%)	22 (14.5%)	0 (0%)	31 (20.4%)	
	wrong	sure	1 (0.7%)	2 (1.3%)	6 (3.9%)	0 (0%)	9 (5.9%)	21 (13.8%)
		unsure	8 (5.3%)	3 (2.0%)	1 (0.7%)	0 (0%)	12(7.9%)	
Group 2 - ADORA	right	sure	57 (44.9%)	39 (30.7%)	7 (5.5%)	0 (0%)	103 (81.1%)	113 (89.0%)
		unsure	1 (0.8%)	4 (3.1%)	5 (3.9%)	0 (0%)	10 (7.9%)	
	wrong	sure	0 (0%)	3 (2.4%)	1 (0.8%)	1 (0.8%)	5 (3.9%)	14 (11.0%)
		unsure	6 (4.7%)	1 (0.8%)	0 (0%)	2 (1.6%)	9 (7.1%)	
Group 2 - UML	right	sure	40 (31.0%)	24 (18.6%)	3 (2.3%)	8 (6.2%)	75 (58.1%)	99 (76.7%)
		unsure	2 (1.6%)	8 (6.2%)	14 (10.9%)	0 (0%)	24 (18.6%)	
	wrong	sure	0 (0%)	2 (1.6%)	7 (5.4%)	0 (0%)	9 (7.0%)	30 (23.3%)
		unsure	13 (10.1%)	6 (4.7%)	1 (0.8%)	1 (0.8%)	21 (16.3)	
Groups 1+2 - ADORA	right	sure	107(39.3%)	87 (32.0%)	21 (7.7%)	0 (0%)	215 (79.0%)	245 (90.1%)
		unsure	1 (0.4%)	10 (3.7%)	19 (7.0%)	0 (0%)	30 (11.0%)	
	wrong	sure	1 (0.4%)	3 (1.1%)	4 (1.5%)	1 (0.4%)	9 (3.3%)	27 (9.9%)
		unsure	9 (3.3%)	7 (2.6%)	0 (0%)	2 (0.7%)	18 (6.6%)	
Groups 1+2 - UML	right	sure	106(37.7%)	52 (18.5%)	9 (3.2%)	8 (2.8%)	175 (62.3%)	230 (81.9%)
		unsure	5 (1.8%)	14 (5.0%)	36 (12.8%)	0 (0%)	55 (19.6%)	
	wrong	sure	1 (0.4%)	4 (1.4%)	13 (4.6%)	0 (0%)	18 (6.4%)	51 (18.1%)
		unsure	21 (7.5%)	9 (3.2%)	2 (0.7%)	1 (0.4%)	33 (11.7%)	

Table 1: Evaluation results of "objective" questionnaire

In Figures 2 and 3, the results are visualized graphically. For example, in the Figure 2, we calculated the percentages (shown on the vertical coordinate axis) by sorting the participants' answers by groups (Group 1, Group 2), models (UML, ADORA), correctness (right, wrong), effort (easy, moderate, difficult, impossible) and confidence (sure, unsure).

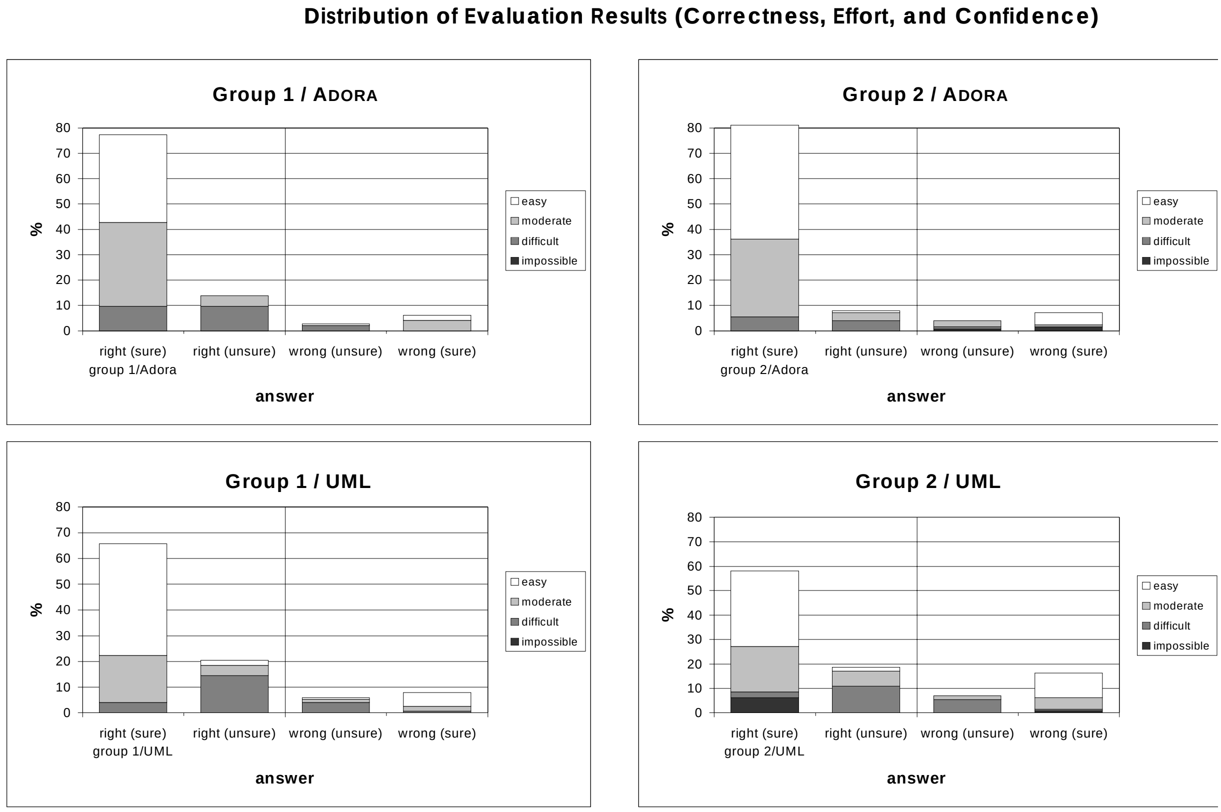


Figure 2. Comprehensibility of models (Groups 1 and 2 shown seperately)

Figure 2 should be read as follows. For example, in Group 1, about 77% of the questions about the ADORA model were answered correctly and the participants were sure about their answer. For about 45% of these answers, the participants judged the answer to be easy to give.

Figure 3 shows the overall results for the first goal of our evaluation, indicating the comprehensibility of ADORA models vs. UML models.

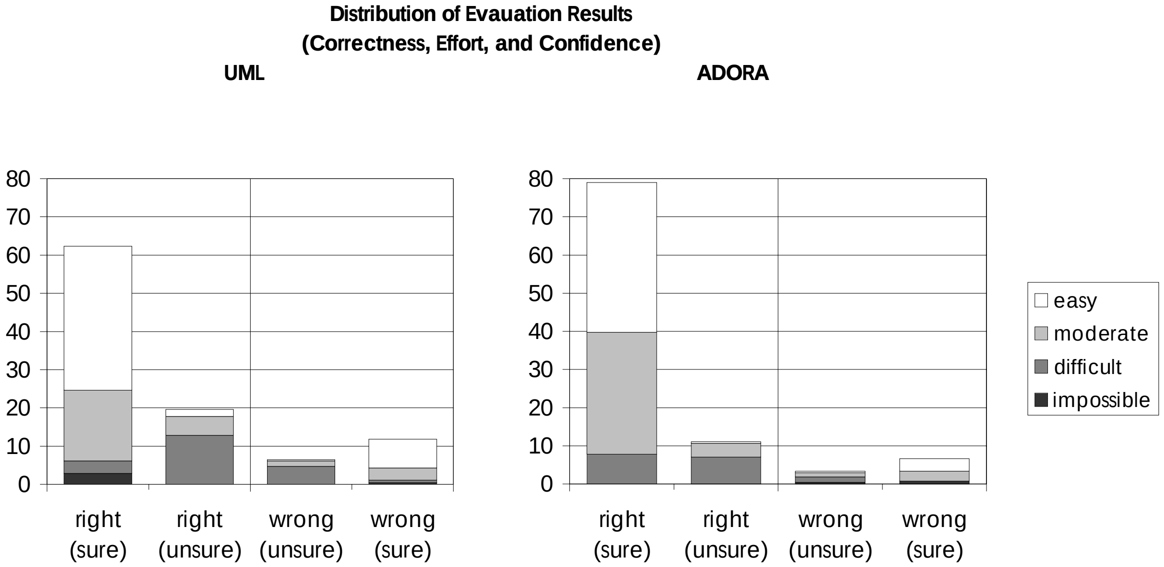


Figure 3. Comprehensibilty of models (two groups consolidated)

4.2 Evaluation the "subjective" questionnaire

Table 2 summarizes the results of the subjective part of the questionnaire.

Statement		strongly agree	mostly agree	mostly disagree	strongly disagree
The specification gives the reader a precise idea about the system components and relationships	ADORA	23 %	62 %	8 %	8 %
	UML	8 %	46 %	31 %	15 %
The structure of the system can be determined easily	ADORA	54 %	31 %	8 %	8 %
	UML	8 %	38 %	23 %	31 %
The specification is an appropriate basis for design and implementation	ADORA	25 %	75 %	0 %	0 %
	UML	0 %	50 %	33 %	17 %
Using an integrated model (ADORA) makes sense		42 %	25 %	33 %	0 %
Using a set of loosely coupled diagrams (UML) makes sense		8 %	17 %	67 %	8 %
Hierarchical decomposiition eases description of large systems		15 %	69 %	15 %	0 %
ADORA eases focusing on parts without losing context		38 %	46 %	15 %	0 %
Decomposition in ADORA eases finding information		46 %	38 %	15 %	0 %
Integrating information from different diagrams is easy in UML		15 %	15 %	46 %	23 %
Specifying objects with their roles and context is adequate		31 %	54 %	15 %	0 %
Describing classes is sufficient		0 %	15 %	62 %	23 %

Table 2: Acceptance of distinct features: ADORA vs. UML

The above table can be read as follows. For example, consider the statement of "the specification gives the reader a precise idea about the system components and relationship". For the

ADORA model, 23% of the participants strongly agree with this statement; 62% of the participants mostly agree this statement; 8% of the participants mostly disagree this statement; and 8% of the participants strongly disagree[1].

4.3 Analysis of the results

A qualitative analysis of the evaluation results yields the following tendencies for both groups.

- *Reading ADORA models is less prone to errors than reading UML models.*
  When we analyze the errors that the participants made when reading the models, we find that the participants made fewer errors with ADORA than with UML. In Group 1, there is a difference of 4.8%, while in Group 2 the difference is 12.3%. In the consolidated results, we have a difference of 8.3%[2]. Using Hypothesis Testing [6], we found that the result is statistically significant at the 0.5% level, which is a very strong result (for details, see Appendix 5).

- *Both groups of participants strongly support our hypothesis that users like the fundamental concepts (abstract objects, hierarchical decomposition, integrated models, etc.) of ADORA and that they prefer them to those of UML.*
  From the Table 4, we can easily get the above conclusion. As the size of the sample is small, we do no statistical analysis here.

From the confidence analysis of the results, we can see that

- *when participants correctly answer a question, they are more confident of themselves after reading the ADORA model*
  Let us study the consolidated results of Group 1 and Group 2. For those correctly answered questions, which were got after participants worked on the ADORA model, 87.8% (215/245) of them were answered by the students with confidence (instead of random guess). For those correctly answered questions, which were got after participants worked on the UML model, 76.1% (175/230) of them were answered by the students with confidence.
  We can see the same tendency, when we study Group 1 (84.8% vs. 76.3%) and Group 2 (91.2% vs. 75.8%) separately. I.e. comparatively speaking, when participants read the ADORA model, they got clearer information. Therefore, their selections of answers were more based on their interpretations of the model rather than by random guess.

  Again, the result is statistically significant at the level of 0.5%.

- *Though participants make a mistake, they are more confident that they are "correct", when reading the UML model.*
  From Table 1 and Table 2, we find a contradiction: Group 1 is against the above statement, and Group 2 is for the statement.

[1] The percentages have been rounded properly, therefore the sums in the rows sometimes yield 99% or 101%.

[2] The possible reasons are: the participants in Group 1 did the ADORA part first. Comparatively, it was easier for them to answer nearly the same questions in UML notations again.

From the consolidated results of Group 1 and Group 2, using the method of Hypothesis Testing again, we calculate the statistics and conclude that there is not a significant difference between the proportions of answering questions wrong with random guess reading two models at a 40% level of significance.
Therefore, we think this statement can **not** be proved from our experiment.

- *From the efforts analysis, we can draw no special statements in favor of either UML or ADORA.*

5. Conclusions

Despite the fact that the number of participants is fairly small, these results strongly support the comprehensibility hypothesis and also show a clear trend that an ADORA specification is easier to comprehend than an UML specification. The results also strongly support our hypothesis that users like the fundamental concepts of ADORA and that they prefer them to those of UML.

Our students are just the people who will soon use those specification languages in their industrial careers. Thus they represent the future software engineers in the industrial field. In this sense, working with students in the ADORA validation experiment is reasonable.

Due to our explicit measures for ensuring a fair, unbiased experiment we are confident that our results are not inadvertently biased in favor of ADORA.

The experiment and its encouraging results give us the confidence that the ADORA approach will meet our expectations concerning the comprehensibility of ADORA models and the acceptance of the ADORA concepts. We think that our validation approach can also be applied for doing similar validation work on other modeling languages.

References

1. Berner, S., Joos, S., Glinz, M. Arnold, M. (1998). *A Visualization Concept for Hierarchical Object Models*. **Proceedings 13th IEEE International Conference on Automated Software Engineering (ASE-98)**. 225-228.
2. Glinz, M., Berner, S., Joos, S., Ryser, J., Schett, N., Schmid, R., Xia, Y. (2000). *The ADORA Approach to Object-Oriented Modeling of Software*. Submitted for publication. http://www.ifi.unizh.ch/groups/req/ftp/papers/ADORA_approach.pdf
3. Joos, S. (1999). **ADORA-L – Eine Modellierungssprache zur Spezifikation von Software-Anforderungen [ADORA-L – A modeling language for specifying software requirements. [In German]**. Ph.D. Thesis, University of Zurich.
4. Rumbaugh, J., Jacobson, I., Booch, G. (1999). **The Unified Modeling Language Reference Manual**. Reading, Mass., etc.: Addison-Wesley.
5. Schmid, R., Berner, S., Joos, S., Reutemann R., Ryser, J. (1998): *Spezifikation des Referenzmodells. Arbeitsdokument des AK's Simulation für das RE*, GI-Fachgruppe 2-1-6 .
6. Fogiel, M.(1984). **The Statistics Problem Solver**. Research and Education Association, New York, N.Y. 10018

Appendices

A1. Overview of the Ticketing System

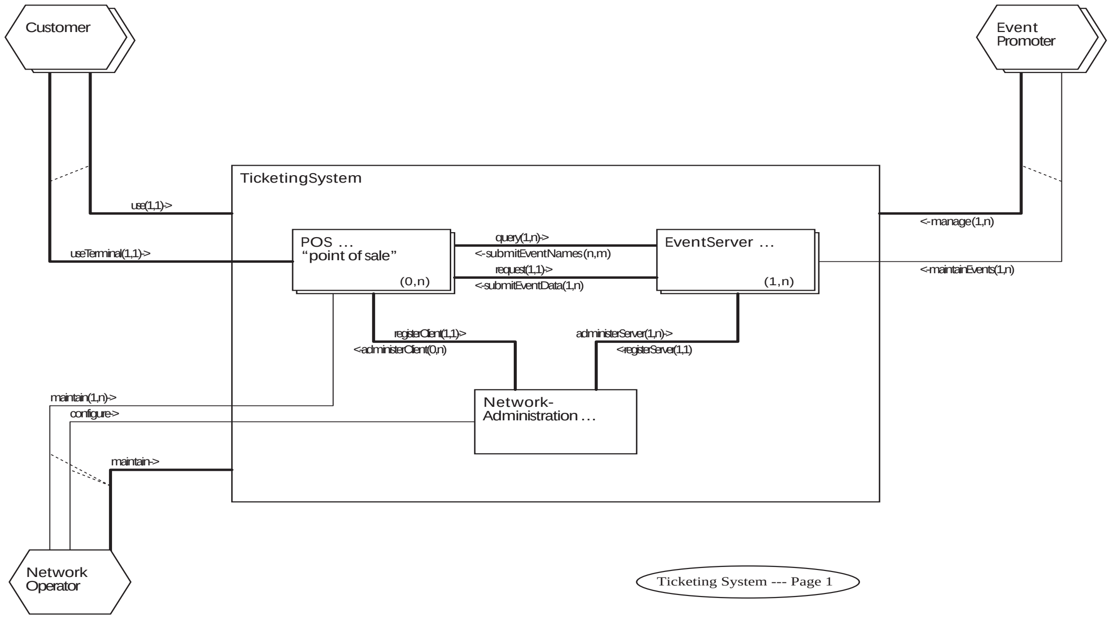
The ticketing system, which is used in this experiment, is roughly introduced here in a natural language. A more detailed description of this system can be read in [5][3].

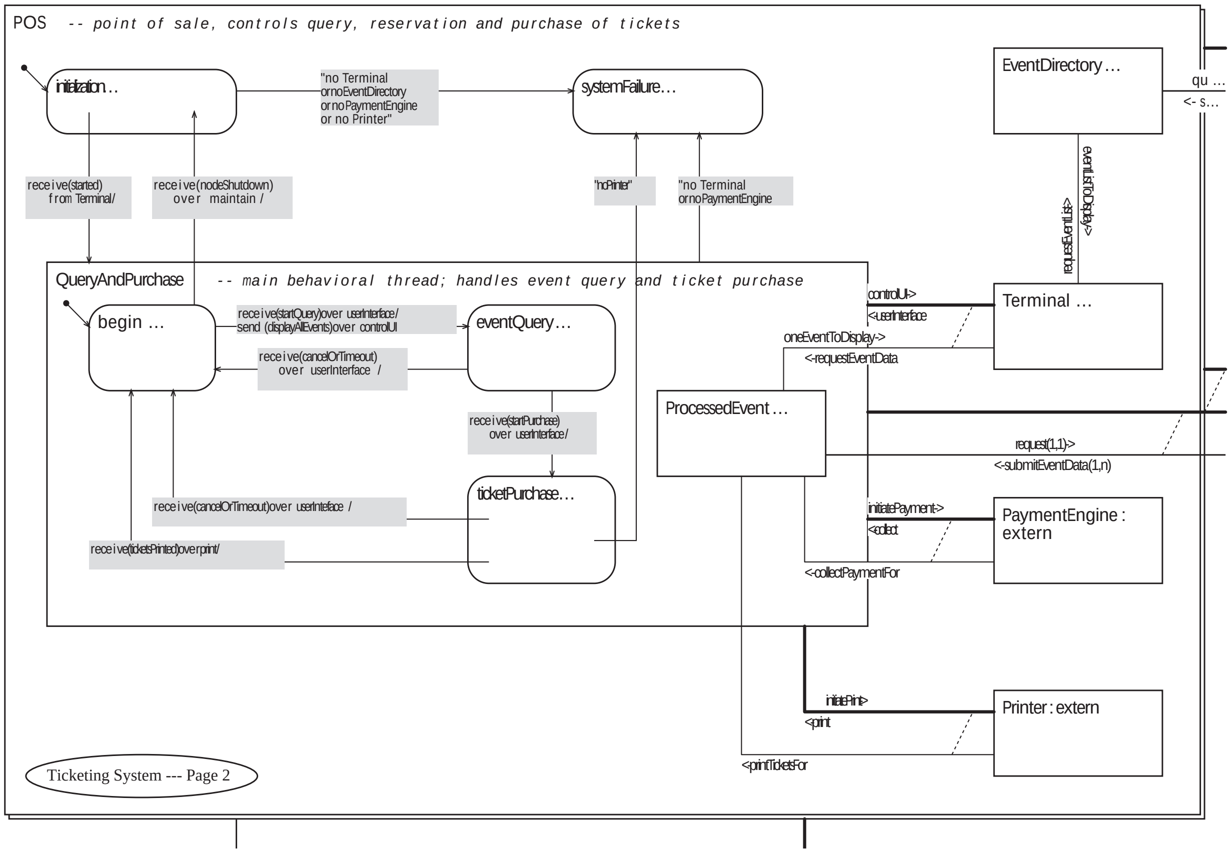
This system is an information management system, whose main functions are
1) Management of the events (films, concerts, etc) taking place in the region. This includes the events information being added, being modified, and being deleted.
2) Handle of the ticket-purchase of those events.

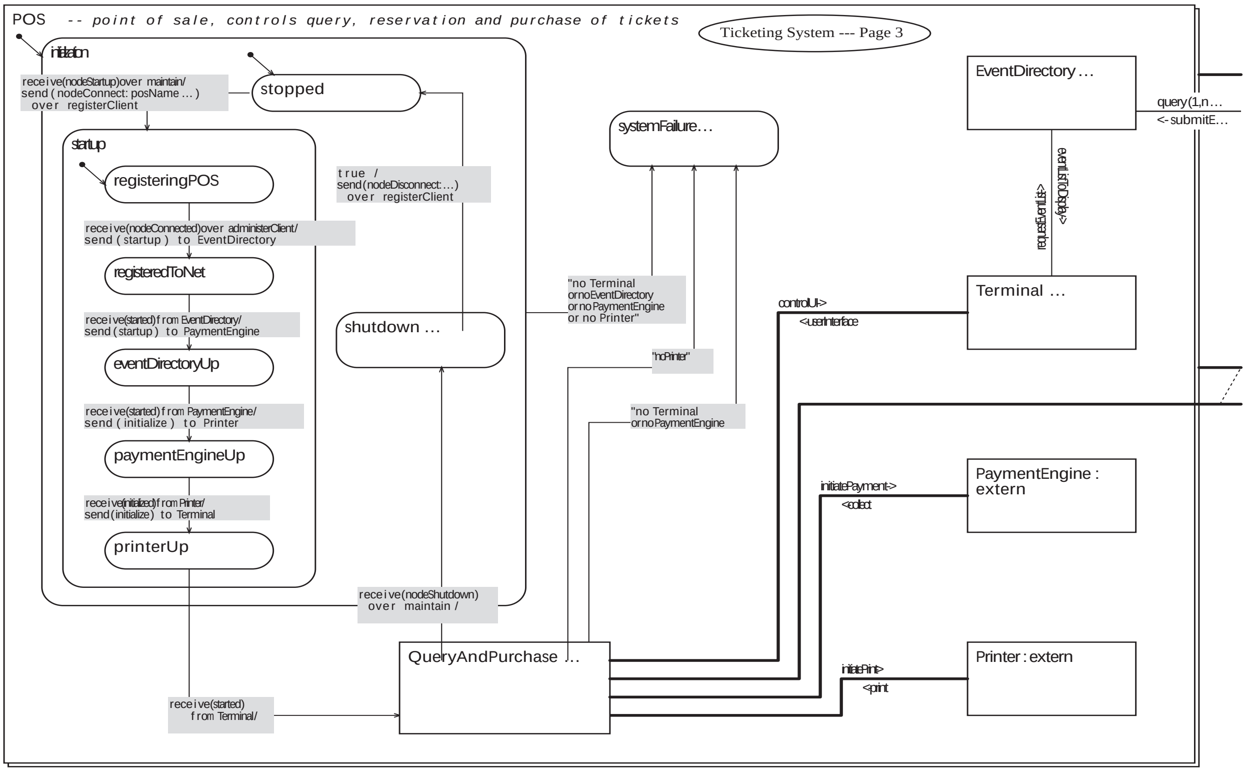
The system is consisted mainly three parts:

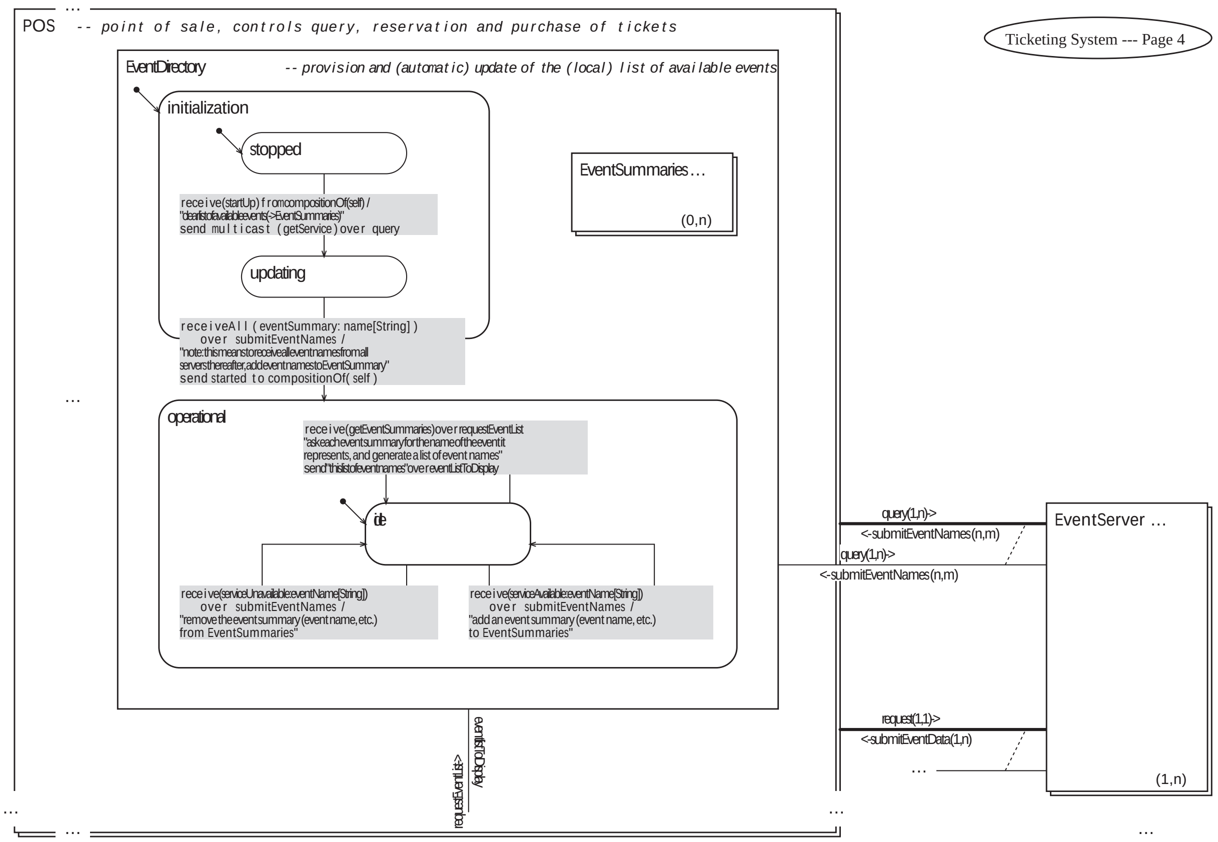
- Event-Server: All the information of events and the information relating to them (e.g. the information of seats, etc.)are processed in the Event Server. One Event-Server may stores the information of one or more events. There are several Enevt-Servers in the system.
- POS: Customers will query all the events in each POS. They can reserve and pay for the tickets of those events in the POS. There are a lot of POSs in the system. After receiving the requests from the customers, POS will communicate with Event-Server, transfer the user commands to the Server, and transfer the processed results from the Server back to the users.
- Network: It connects the Event-Servers and POSs in the system. For the limit of the experiment time, we do not give detailed specification for this part.

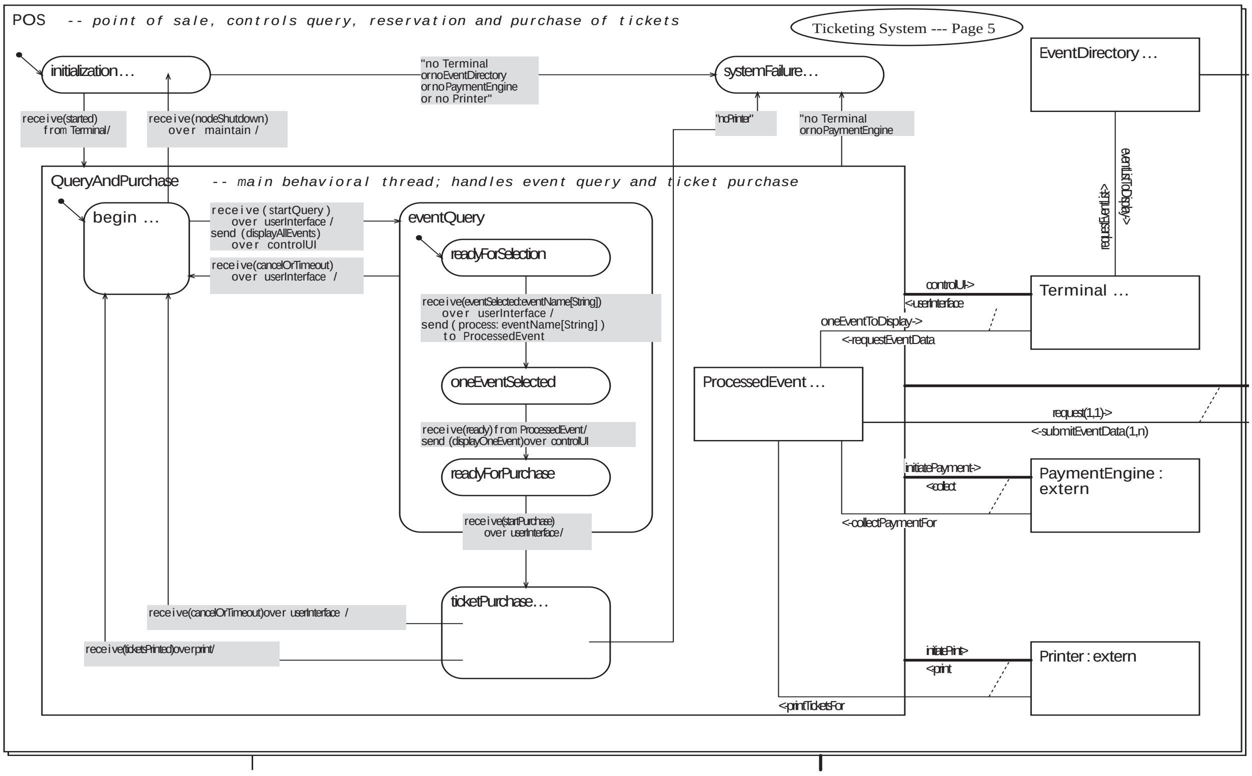
A2. ADORA specification of the ticketing system

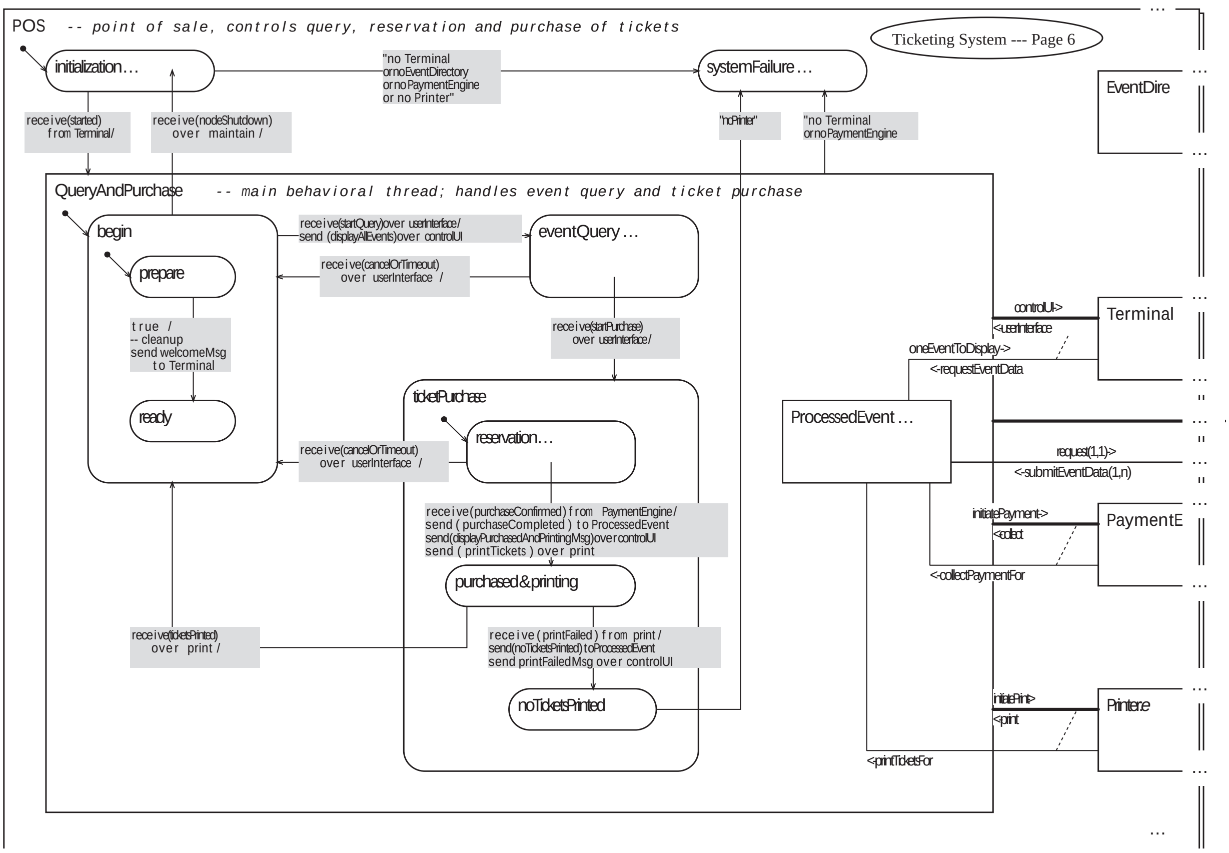


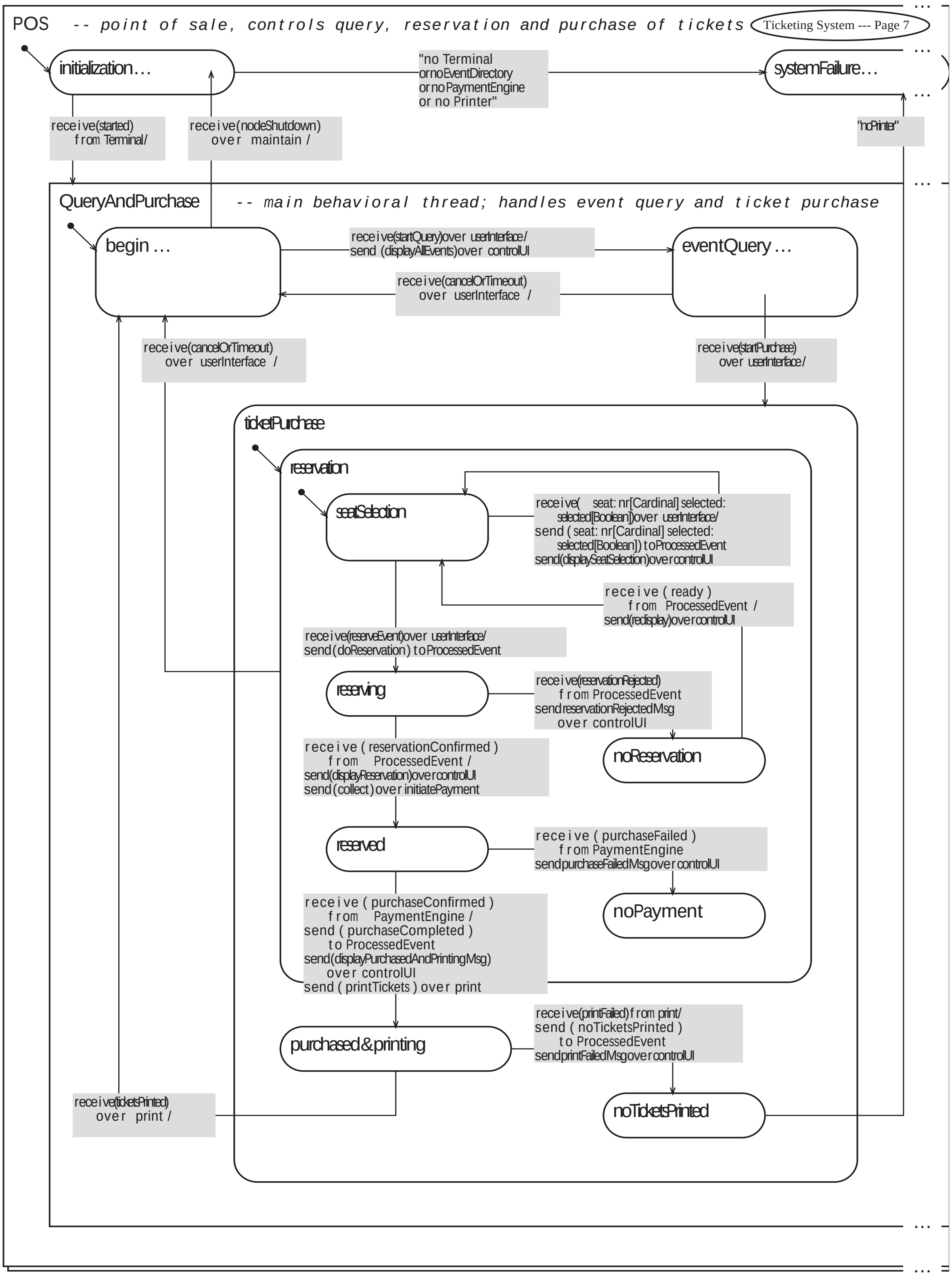

POS -- point of sale, controls query, reservation and purchase of tickets
initialization...
"no Terminal or no EventDirectory or no PaymentEngine or no Printer"
systemFailure...
EventDirectory...
qu ...
<- s...
receive(started) from Terminal/
receive(nodeShutdown) over maintain /
"noPrinter"
"no Terminal or no PaymentEngine
eventListToDisplay->
requestEventList<-
QueryAndPurchase -- main behavioral thread; handles event query and ticket purchase
begin ...
receive(startQuery)over userInterface/ send (displayAllEvents)over controlUI
eventQuery...
receive(cancelOrTimeout) over userInterface /
receive(startPurchase) over userInterface/
ticketPurchase...
receive(cancelOrTimeout)over userInteface /
receive(ticketsPrinted)over print/
controlUI>
<userInterface
oneEventToDisplay->
<-requestEventData
Terminal ...
ProcessedEvent ...
request(1,1)>
<-submitEventData(1,n)
initiatePayment>
<collect
<-collectPaymentFor
PaymentEngine : extern
initiatePrint>
<print
<printTicketsFor
Printer : extern
Ticketing System --- Page 2

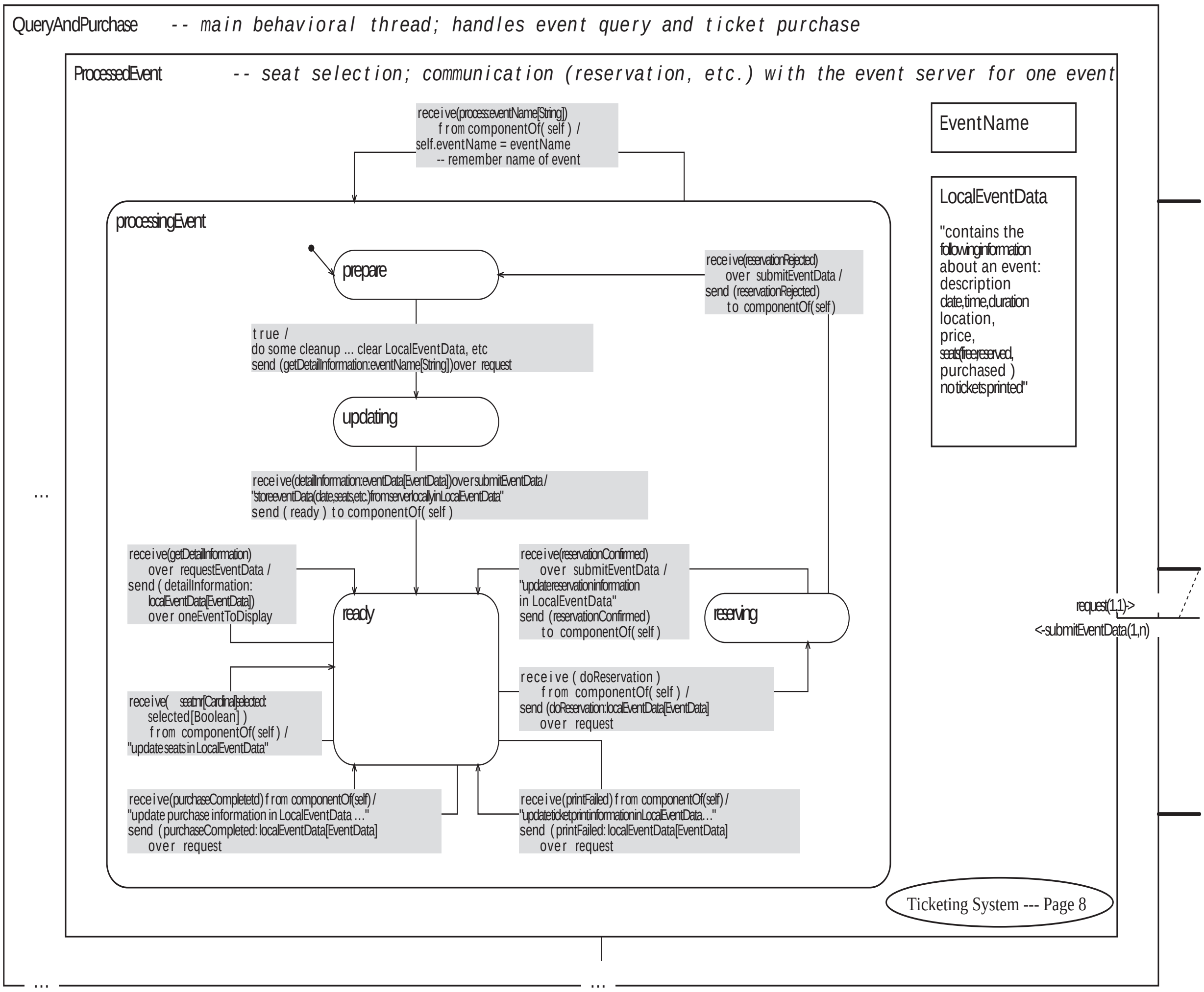

POS -- point of sale, controls query, reservation and purchase of tickets
Ticketing System --- Page 3
initialization
receive(nodeStartup)over maintain/
send (nodeConnect: posName ...)
over registerClient
stopped
startup
registeringPOS
receive(nodeConnected)over administerClient/
send (startup) to EventDirectory
registeredToNet
receive(started) from EventDirectory/
send (startup) to PaymentEngine
eventDirectoryUp
receive(started) from PaymentEngine/
send (initialize) to Printer
paymentEngineUp
receive(initialized) from Printer/
send(initialize) to Terminal
printerUp
true /
send(nodeDisconnect: ...)
over registerClient
shutdown ...
receive(nodeShutdown)
over maintain /
receive(started)
from Terminal/
QueryAndPurchase ...
systemFailure ...
"no Terminal
ornoEventDirectory
or no PaymentEngine
or no Printer"
"noPrinter"
"no Terminal
ornoPaymentEngine
EventDirectory ...
query(1,n...
<-submitE...
eventListToDisplay>
requestEventList>
Terminal ...
controlUI>
<userInterface
PaymentEngine :
extern
initiatePayment>
<collect
Printer : extern
initiatePrint>
<print


...
POS -- point of sale, controls query, reservation and purchase of tickets
Ticketing System --- Page 4
EventDirectory -- provision and (automatic) update of the (local) list of available events
initialization
stopped
receive(startUp) from compositionOf(self) /
"clearlistofavailableevents(->EventSummaries)"
send multicast (getService) over query
updating
receiveAll (eventSummary: name[String])
over submitEventNames /
"note: this means to receive all event names from all
servers thereafter, add event names to EventSummary"
send started to compositionOf(self)
EventSummaries ...
(0,n)
operational
receive(getEventSummaries) over requestEventList
"ask each event summary for the name of the event it
represents, and generate a list of event names"
send "this list of event names" over eventListToDisplay
idle
receive(serviceUnavailable:eventName[String])
over submitEventNames /
"remove the event summary (event name, etc.)
from EventSummaries"
receive(serviceAvailable:eventName[String])
over submitEventNames /
"add an event summary (event name, etc.)
to EventSummaries"
...
requestEventList>
eventListToDisplay
query(1,n)>
<-submitEventNames(n,m)
query(1,n)>
<-submitEventNames(n,m)
EventServer ...
request(1,1)>
<-submitEventData(1,n)
...
(1,n)
...
...
...
...

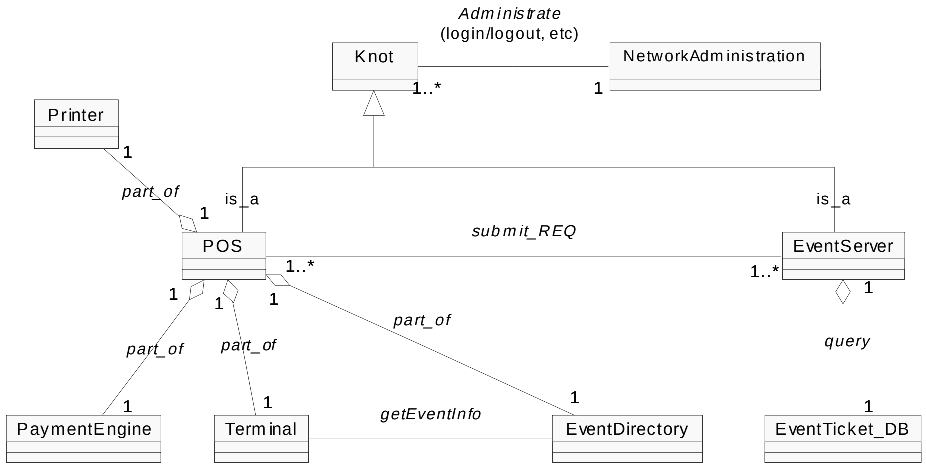

POS -- point of sale, controls query, reservation and purchase of tickets
Ticketing System --- Page 5
initialization…
"no Terminal ornoEventDirectory ornoPaymentEngine or no Printer"
systemFailure…
EventDirectory…
receive(started) from Terminal/
receive(nodeShutdown) over maintain /
"noPrinter"
"no Terminal ornoPaymentEngine
QueryAndPurchase -- main behavioral thread; handles event query and ticket purchase
begin …
receive (startQuery) over userInterface / send (displayAllEvents) over controlUI
receive(cancelOrTimeout) over userInterface /
eventQuery
readyForSelection
receive(eventSelected:eventName[String]) over userInterface / send (process: eventName[String]) to ProcessedEvent
oneEventSelected
receive(ready) from ProcessedEvent/ send (displayOneEvent)over controlUI
readyForPurchase
receive(startPurchase) over userInterface/
ticketPurchase…
receive(cancelOrTimeout)over userInterface /
receive(ticketsPrinted)overprint/
eventListToDisplay->
<-requestEventList
controlUI->
<-userInterface
Terminal …
oneEventToDisplay->
<-requestEventData
ProcessedEvent …
request(1,1)>
<-submitEventData(1,n)
initiatePayment->
<-collect
PaymentEngine : extern
<-collectPaymentFor
initiatePrint>
<print
Printer : extern
<printTicketsFor


…
POS -- point of sale, controls query, reservation and purchase of tickets
Ticketing System --- Page 6
initialization…
"no Terminal ornoEventDirectory ornoPaymentEngine or no Printer"
systemFailure…
EventDire
receive(started) from Terminal/
receive(nodeShutdown) over maintain /
"noPrinter"
"no Terminal ornoPaymentEngine
QueryAndPurchase -- main behavioral thread; handles event query and ticket purchase
begin
prepare
true / -- cleanup send welcomeMsg to Terminal
ready
receive(startQuery)over userInterface/ send (displayAllEvents)over controlUI
receive(cancelOrTimeout) over userInterface /
eventQuery …
receive(startPurchase) over userInterface/
ticketPurchase
reservation…
receive(cancelOrTimeout) over userInterface /
receive(purchaseConfirmed) from PaymentEngine/ send (purchaseCompleted) to ProcessedEvent send(displayPurchasedAndPrintingMsg)over controlUI send (printTickets) over print
purchased&printing
receive(ticketsPrinted) over print /
receive (printFailed) from print / send(noTicketsPrinted) toProcessedEvent send printFailedMsg over controlUI
noTicketsPrinted
controlUI->
<-userInterface
Terminal
oneEventToDisplay->
<-requestEventData
ProcessedEvent …
request(1,1)>
<-submitEventData(1,n)
initiatePayment->
<-collect
PaymentE
<-collectPaymentFor
initiatePrint>
<print
Printere
<printTicketsFor
…


POS -- point of sale, controls query, reservation and purchase of tickets
Ticketing System --- Page 7
...
initialization...
"no Terminal
or no EventDirectory
or no PaymentEngine
or no Printer"
systemFailure...
receive(started)
from Terminal/
receive(nodeShutdown)
over maintain /
"noPrinter"
QueryAndPurchase -- main behavioral thread; handles event query and ticket purchase
begin ...
receive(startQuery)over userInterface/
send (displayAllEvents)over controlUI
eventQuery ...
receive(cancelOrTimeout)
over userInterface /
receive(cancelOrTimeout)
over userInterface /
receive(startPurchase)
over userInterface/
ticketPurchase
reservation
seatSelection
receive(seat: nr[Cardinal] selected:
selected[Boolean])over userInterface/
send (seat: nr[Cardinal] selected:
selected[Boolean]) to ProcessedEvent
send(displaySeatSelection)over controlUI
receive (ready)
from ProcessedEvent /
send(redisplay)over controlUI
receive(reserveEvent)over userInterface/
send (doReservation) to ProcessedEvent
reserving
receive(reservationRejected)
from ProcessedEvent
send reservationRejectedMsg
over controlUI
noReservation
receive (reservationConfirmed)
from ProcessedEvent /
send(displayReservation)over controlUI
send (collect) over initiatePayment
reserved
receive (purchaseFailed)
from PaymentEngine
send purchaseFailedMsg over controlUI
noPayment
receive (purchaseConfirmed)
from PaymentEngine /
send (purchaseCompleted)
to ProcessedEvent
send(displayPurchasedAndPrintingMsg)
over controlUI
send (printTickets) over print
purchased&printing
receive(printFailed) from print/
send (noTicketsPrinted)
to ProcessedEvent
sendprintFailedMsg over controlUI
receive(ticketsPrinted)
over print /
noTicketsPrinted

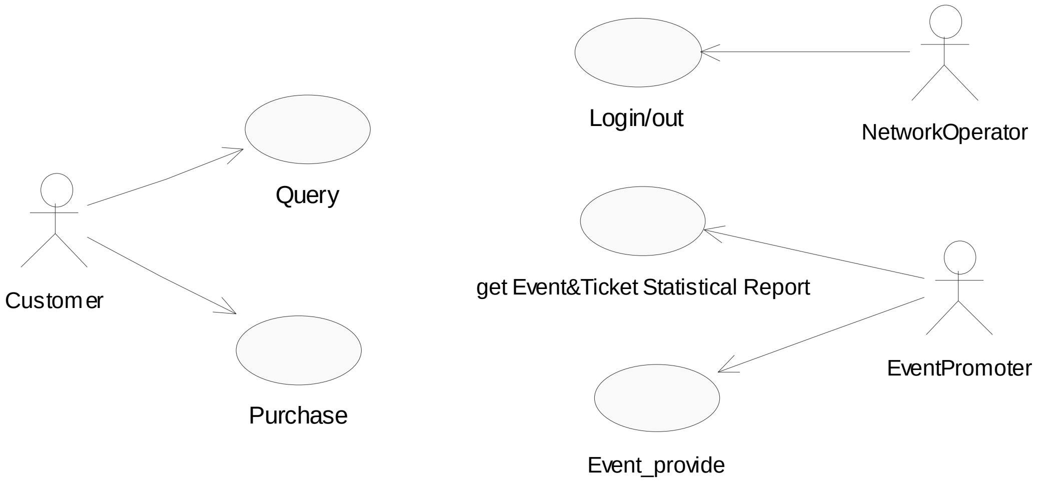

QueryAndPurchase -- main behavioral thread; handles event query and ticket purchase
ProcessedEvent -- seat selection; communication (reservation, etc.) with the event server for one event
receive(process:eventName[String])
from componentOf(self) /
self.eventName = eventName
-- remember name of event
EventName
LocalEventData
"contains the
followinginformation
about an event:
description
date,time,duration
location,
price,
seats(free,reserved,
purchased)
noticketsprinted"
processingEvent
prepare
receive(reservationRejected)
over submitEventData /
send (reservationRejected)
to componentOf(self)
true /
do some cleanup ... clear LocalEventData, etc
send (getDetailInformation:eventName[String])over request
updating
...
receive(detailInformation:eventData[EventData])oversubmitEventData /
"storeeventData(date,seats,etc.)fromserverlocallyinLocalEventData"
send (ready) to componentOf(self)
receive(getDetailInformation)
over requestEventData /
send (detailInformation:
localEventData[EventData])
over oneEventToDisplay
ready
receive(reservationConfirmed)
over submitEventData /
"updatereservationinformation
in LocalEventData"
send (reservationConfirmed)
to componentOf(self)
reserving
request(1,1)>
<-submitEventData(1,n)
receive(seat:nr[Cardinal]selected:
selected[Boolean])
from componentOf(self) /
"update seats in LocalEventData"
receive (doReservation)
from componentOf(self) /
send (doReservation:localEventData[EventData]
over request
receive(purchaseCompletetd) from componentOf(self) /
"update purchase information in LocalEventData ..."
send (purchaseCompleted: localEventData[EventData]
over request
receive(printFailed) from componentOf(self) /
"updateticketprintinformationinLocalEventData..."
send (printFailed: localEventData[EventData]
over request
Ticketing System --- Page 8
...
...

A3. UML specification of the ticketing system



Domain Classes in the Ticketing System

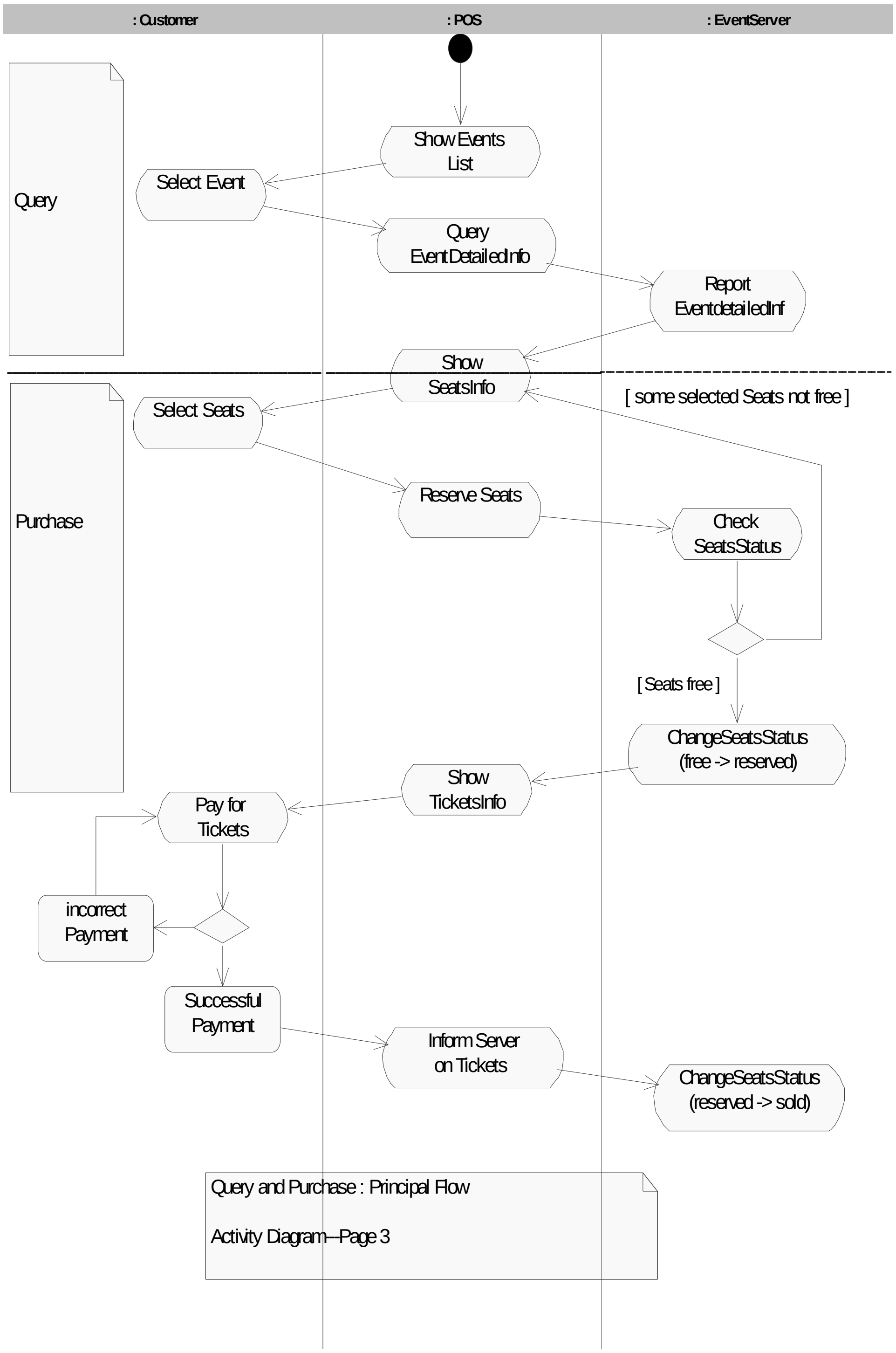
Class Diagram -- Page 1

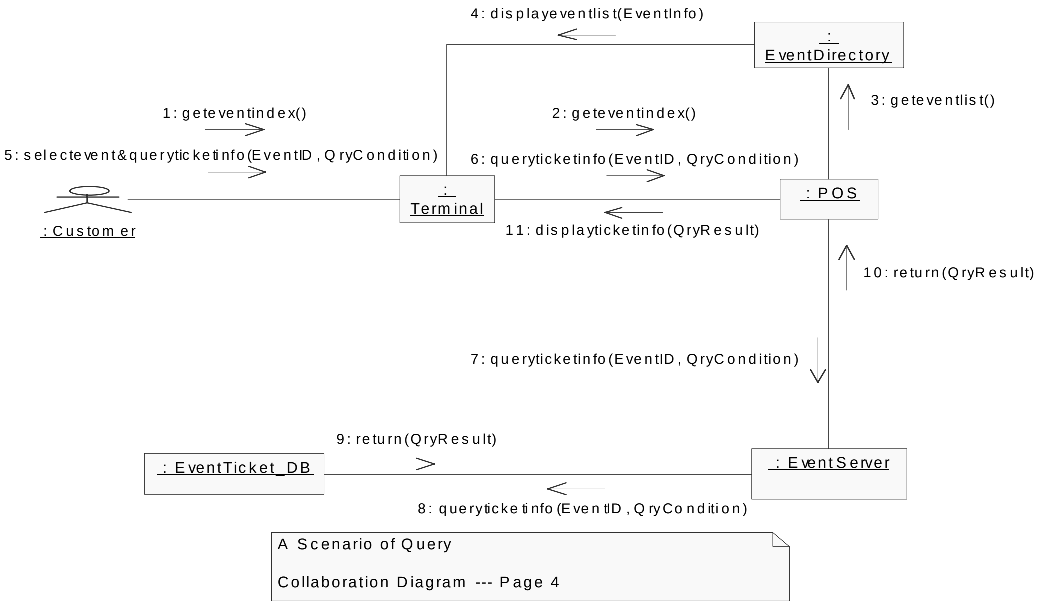


A User Case View of the whole System

With limited size of the model, the detailed scenarios of User Cases: "Login/out", "get Event&Ticket statistical Report", and "Event_porvide" will not be included in this model.

Use Case Diagram --- Page 2

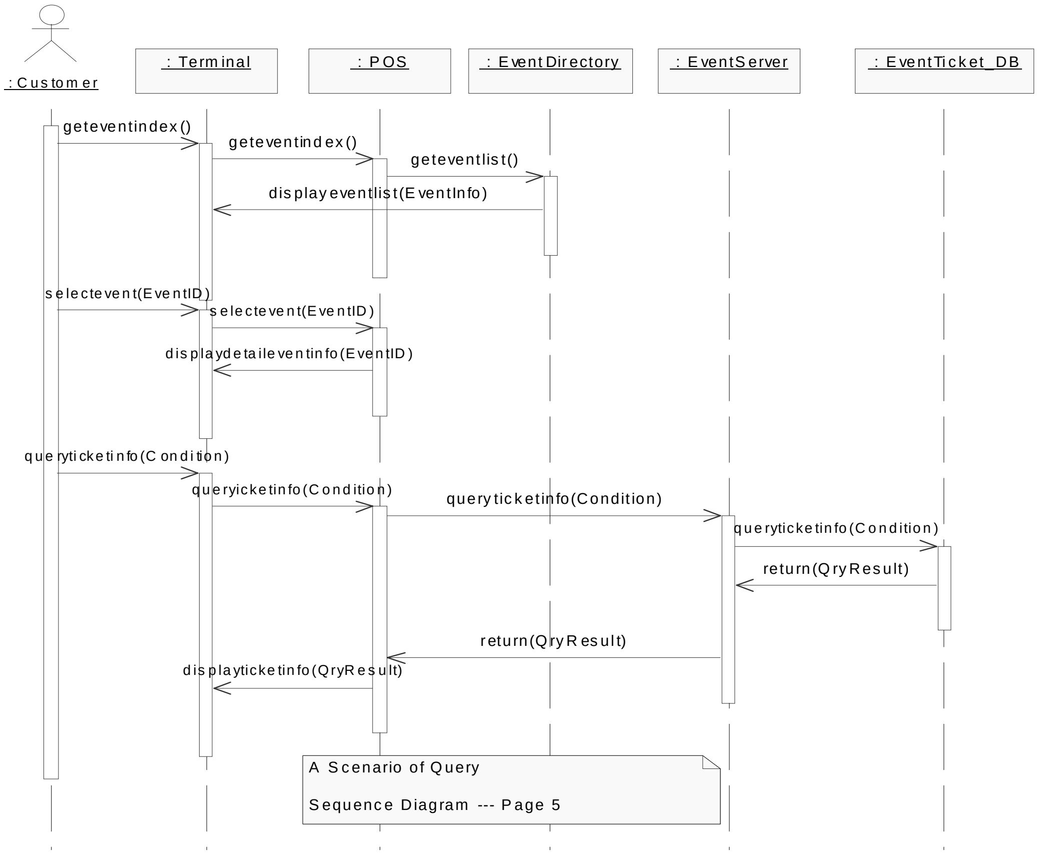

: Customer
: POS
: EventServer
Query
Show Events List
Select Event
Query EventDetailedInfo
Report EventdetailedInf
Show SeatsInfo
Purchase
Select Seats
[some selected Seats not free]
Reserve Seats
Check SeatsStatus
[Seats free]
ChangeSeatsStatus (free -> reserved)
Show TicketsInfo
Pay for Tickets
incorrect Payment
Successful Payment
Inform Server on Tickets
ChangeSeatsStatus (reserved -> sold)
Query and Purchase : Principal Flow
Activity Diagram--Page 3



4: displayeventlist(EventInfo)
: EventDirectory
3: geteventlist()
1: geteventindex()
2: geteventindex()
5: selectevent&queryticketinfo(EventID , QryCondition)
6: queryticketinfo(EventID, QryCondition)
: Terminal
: POS
: Customer
11: displayticketinfo(QryResult)
10: return(QryResult)
7: queryticketinfo(EventID, QryCondition)
9: return(QryResult)
: EventTicket_DB
: EventServer
8: queryticketinfo(EventID , QryCondition)

A Scenario of Query

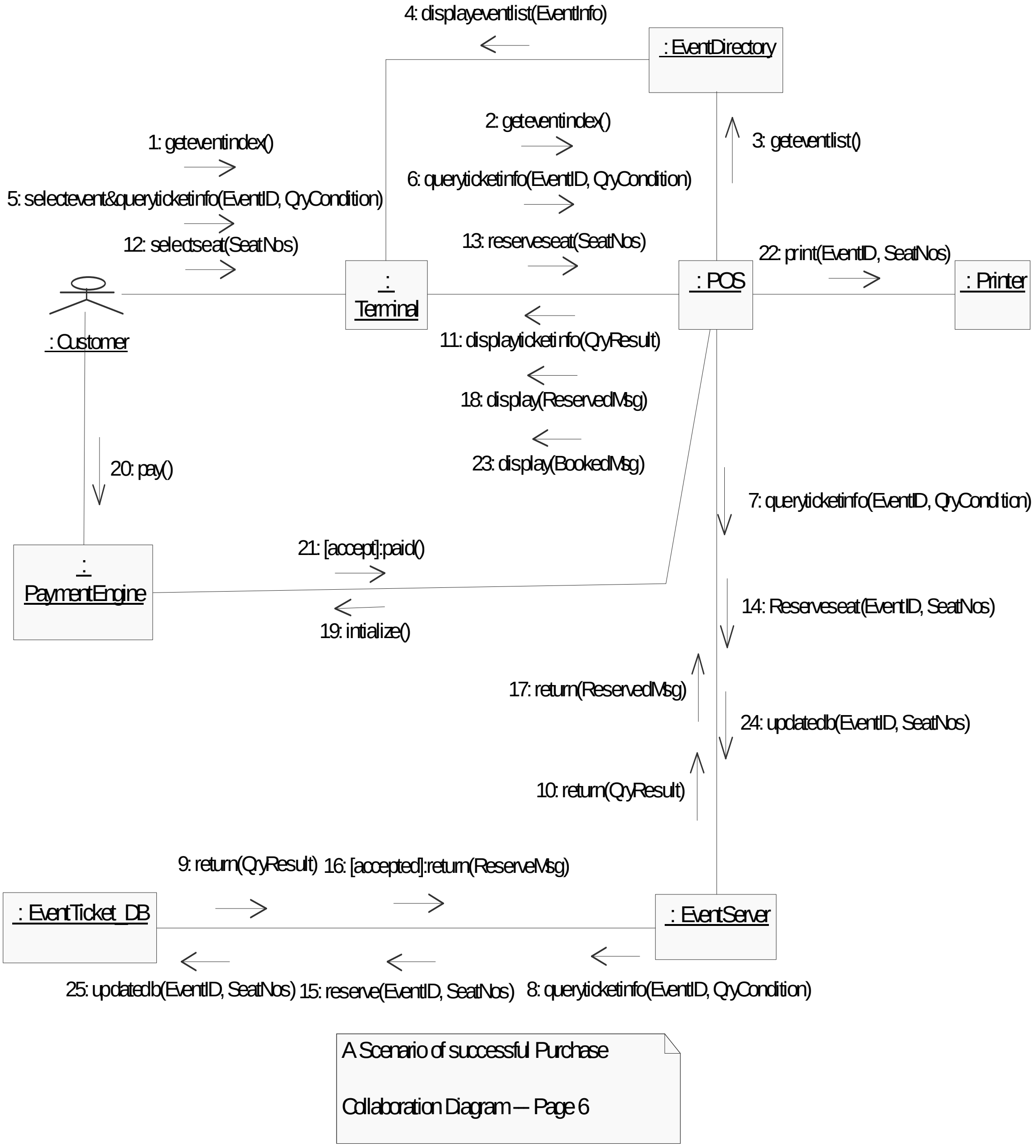
Collaboration Diagram --- Page 4

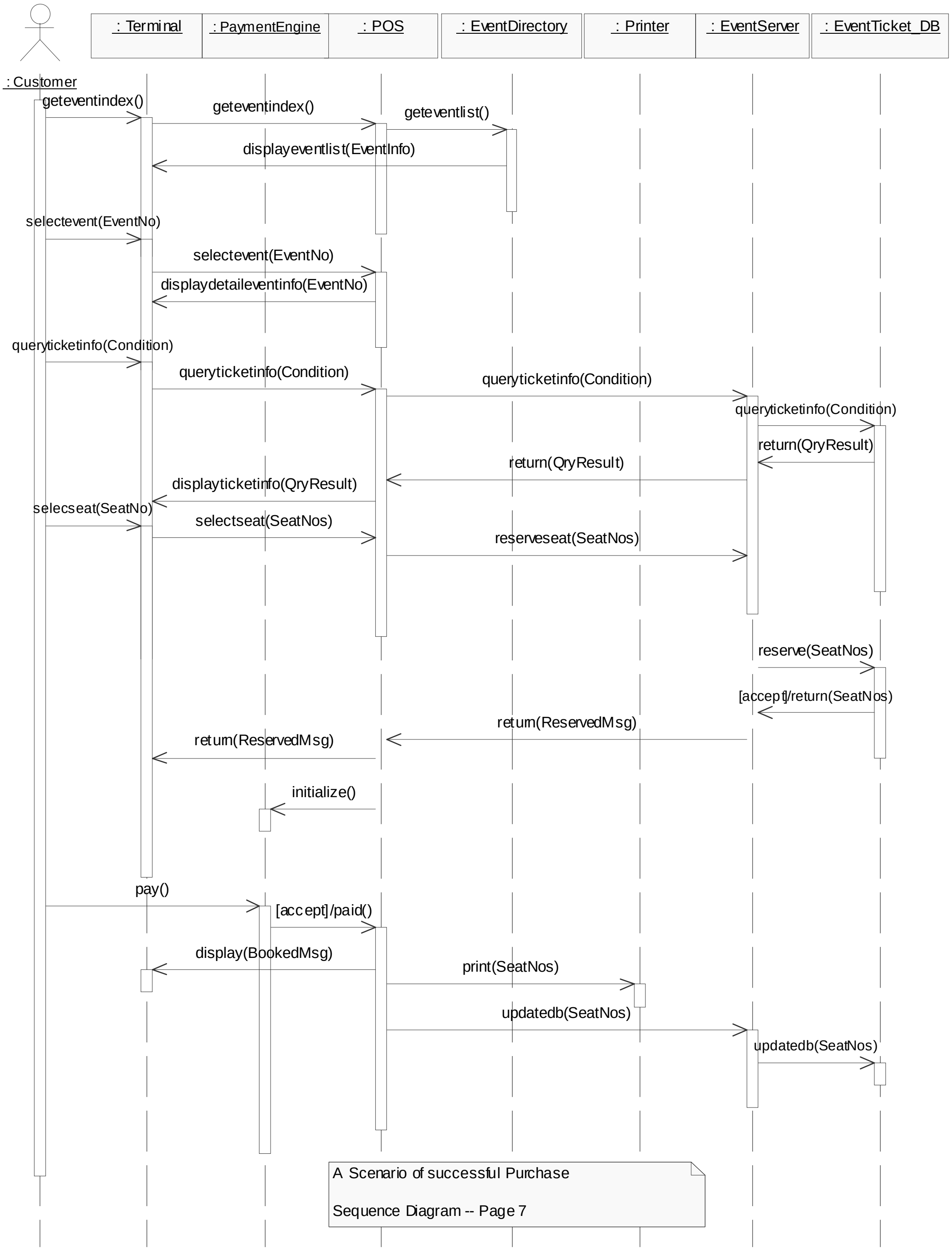


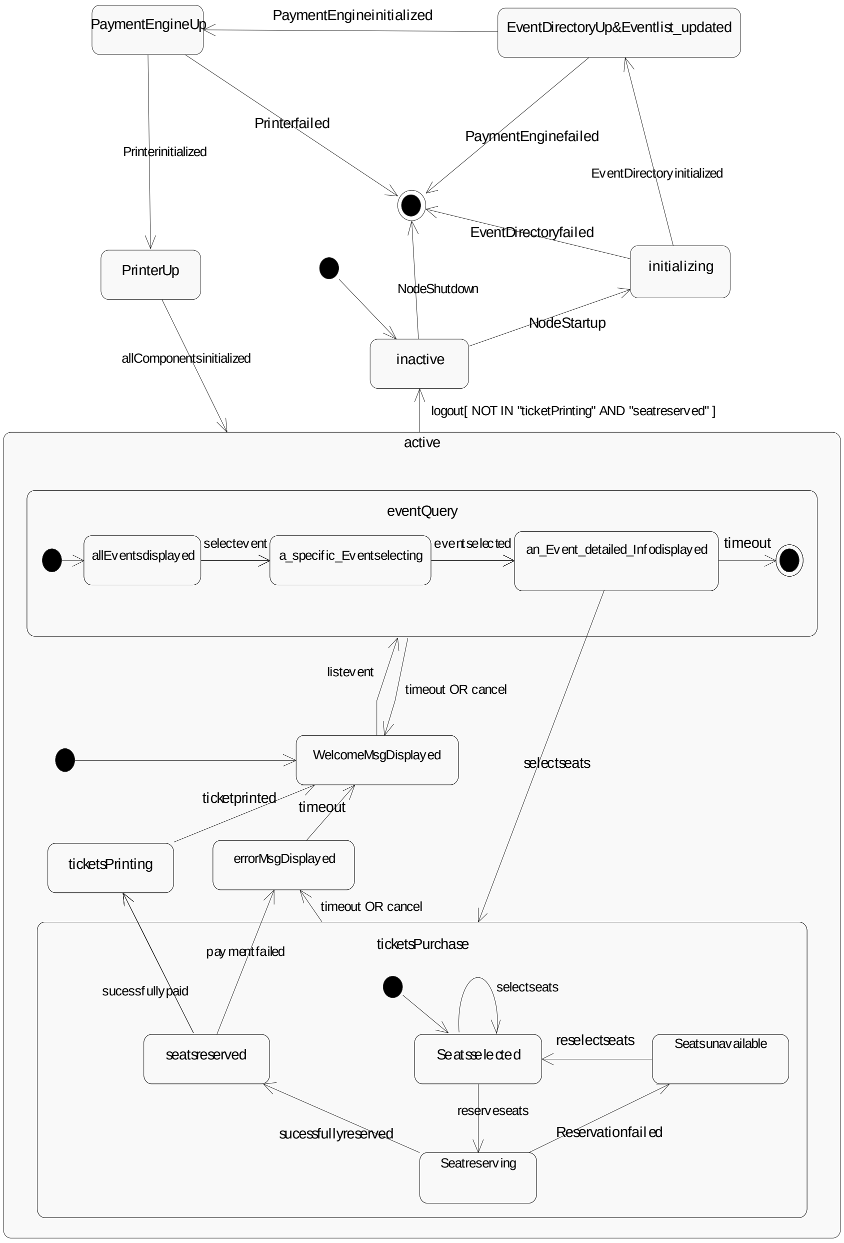
: Customer
: Terminal
: POS
: EventDirectory
: EventServer
: EventTicket_DB
geteventindex()
geteventindex()
geteventlist()
displayeventlist(EventInfo)
selectevent(EventID)
selectevent(EventID)
displaydetaileventinfo(EventID)
queryticketinfo(Condition)
queryicketinfo(Condition)
queryticketinfo(Condition)
queryticketinfo(Condition)
return(QryResult)
return(QryResult)
displayticketinfo(QryResult)

A Scenario of Query

Sequence Diagram --- Page 5

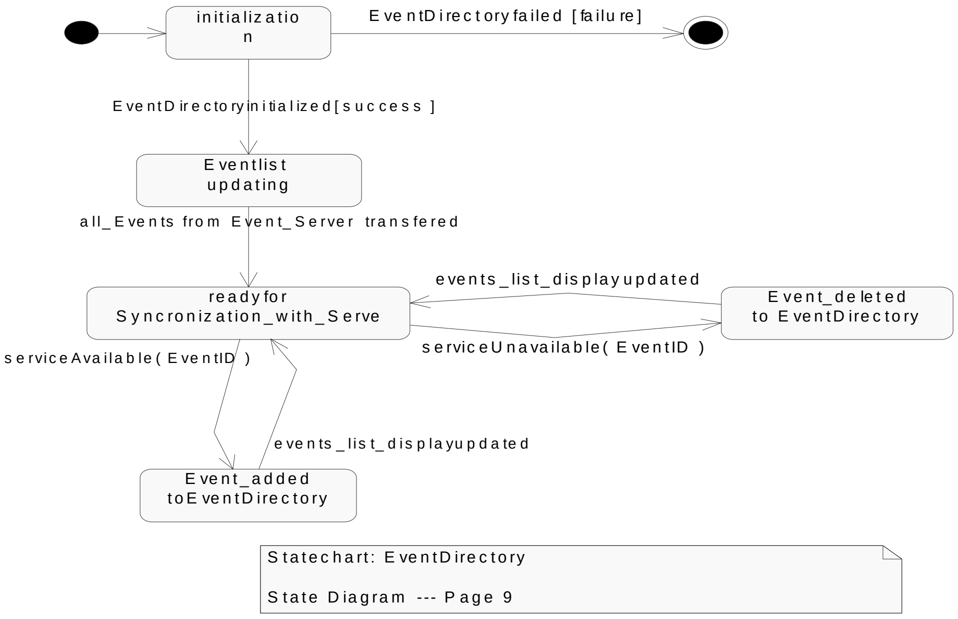

4: displayeventlist(EventInfo)
: EventDirectory
2: geteventindex()
3: geteventlist()
1: geteventindex()
6: queryticketinfo(EventID, QryCondition)
5: selectevent&queryticketinfo(EventID, QryCondition)
13: reserveseat(SeatNos)
12: selectseat(SeatNos)
22: print(EventID, SeatNos)
: Terminal
: POS
: Printer
: Customer
11: displayticketinfo(QryResult)
18: display(ReservedMsg)
23: display(BookedMsg)
20: pay()
7: queryticketinfo(EventID, QryCondition)
21: [accept]:paid()
: PaymentEngine
19: intialize()
14: Reserveseat(EventID, SeatNos)
17: return(ReservedMsg)
24: updatedb(EventID, SeatNos)
10: return(QryResult)
9: return(QryResult)
16: [accepted]:return(ReserveMsg)
: EventTicket_DB
: EventServer
25: updatedb(EventID, SeatNos)
15: reserve(EventID, SeatNos)
8: queryticketinfo(EventID, QryCondition)
A Scenario of successful Purchase
Collaboration Diagram — Page 6

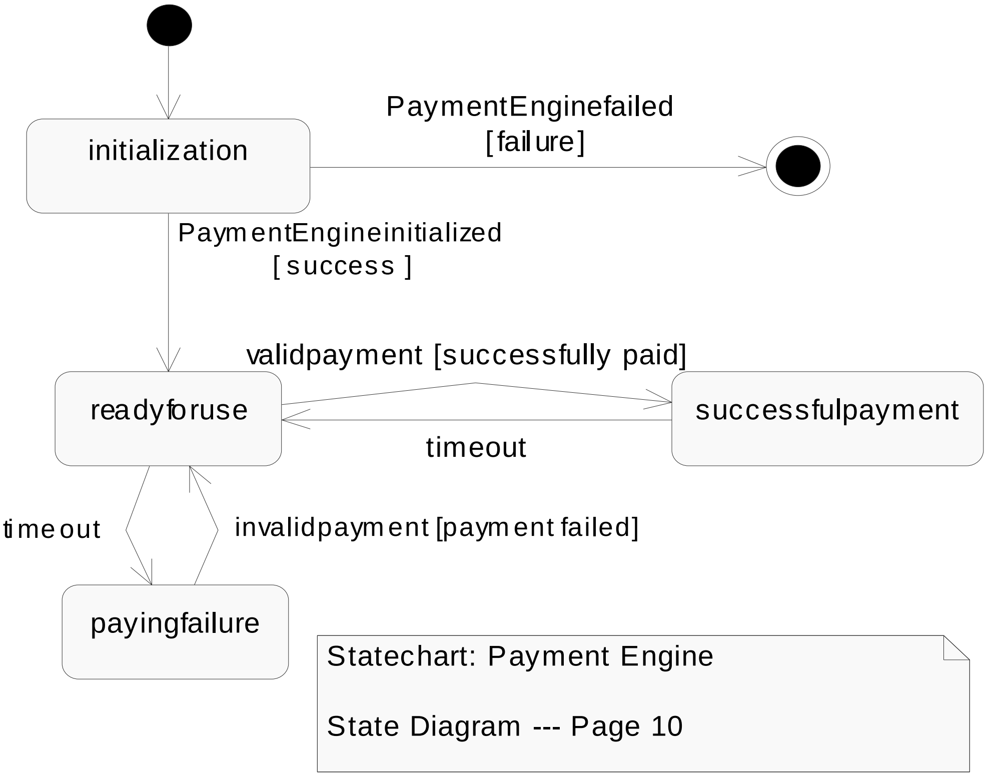

: Customer
: Terminal
: PaymentEngine
: POS
: EventDirectory
: Printer
: EventServer
: EventTicket_DB
geteventindex()
geteventindex()
geteventlist()
displayeventlist(EventInfo)
selectevent(EventNo)
selectevent(EventNo)
displaydetaileventinfo(EventNo)
queryticketinfo(Condition)
queryticketinfo(Condition)
queryticketinfo(Condition)
queryticketinfo(Condition)
return(QryResult)
return(QryResult)
displayticketinfo(QryResult)
selecseat(SeatNo)
selectseat(SeatNos)
reserveseat(SeatNos)
reserve(SeatNos)
[accept]/return(SeatNos)
return(ReservedMsg)
return(ReservedMsg)
initialize()
pay()
[accept]/paid()
display(BookedMsg)
print(SeatNos)
updatedb(SeatNos)
updatedb(SeatNos)
A Scenario of successful Purchase
Sequence Diagram -- Page 7



Statechart: POS

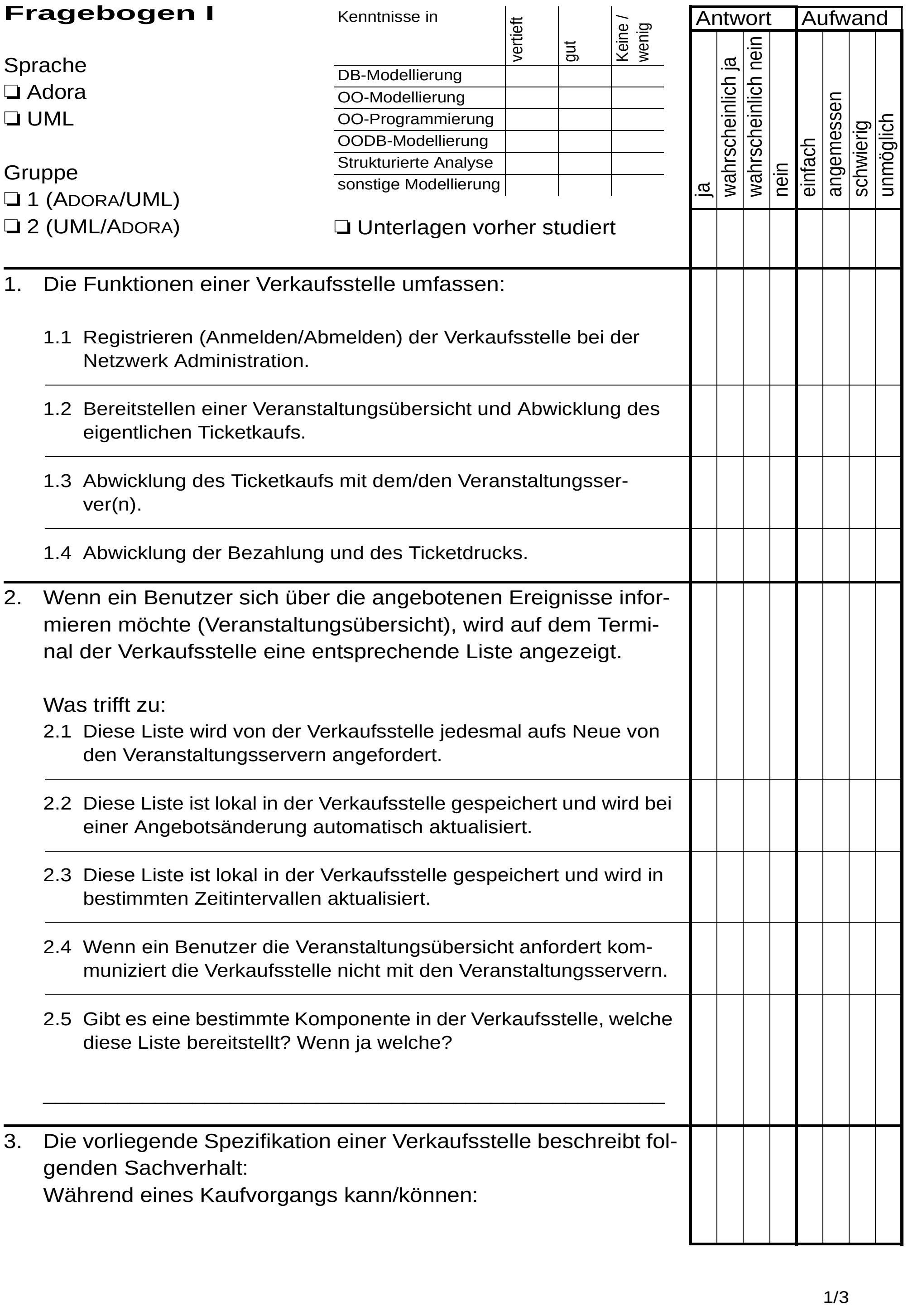
State Diagram --- Page 8


initialization
EventDirectoryfailed [failure]
EventDirectoryinitialized[success]
Eventlist updating
all_Events from Event_Server transfered
readyfor Syncronization_with_Serve
events_list_displayupdated
Event_deleted to EventDirectory
serviceUnavailable(EventID)
serviceAvailable(EventID)
events_list_displayupdated
Event_added toEventDirectory
Statechart: EventDirectory
State Diagram --- Page 9


initialization
PaymentEnginefailed [failure]
PaymentEngineinitialized [success]
validpayment [successfully paid]
readyforuse
successfulpayment
timeout
timeout
invalidpayment [payment failed]
payingfailure
Statechart: Payment Engine
State Diagram --- Page 10

A.4. Questionnaire

A4.1. Objective” questionnaire (in German)



Fragebogen I

Sprache
❑ Adora
❑ UML

Gruppe
❑ 1 (Adora/UML)
❑ 2 (UML/Adora)

Kenntnisse in	vertieft	gut	Keine / wenig
DB-Modellierung			
OO-Modellierung			
OO-Programmierung			
OODB-Modellierung			
Strukturierte Analyse			
sonstige Modellierung			

❑ Unterlagen vorher studiert

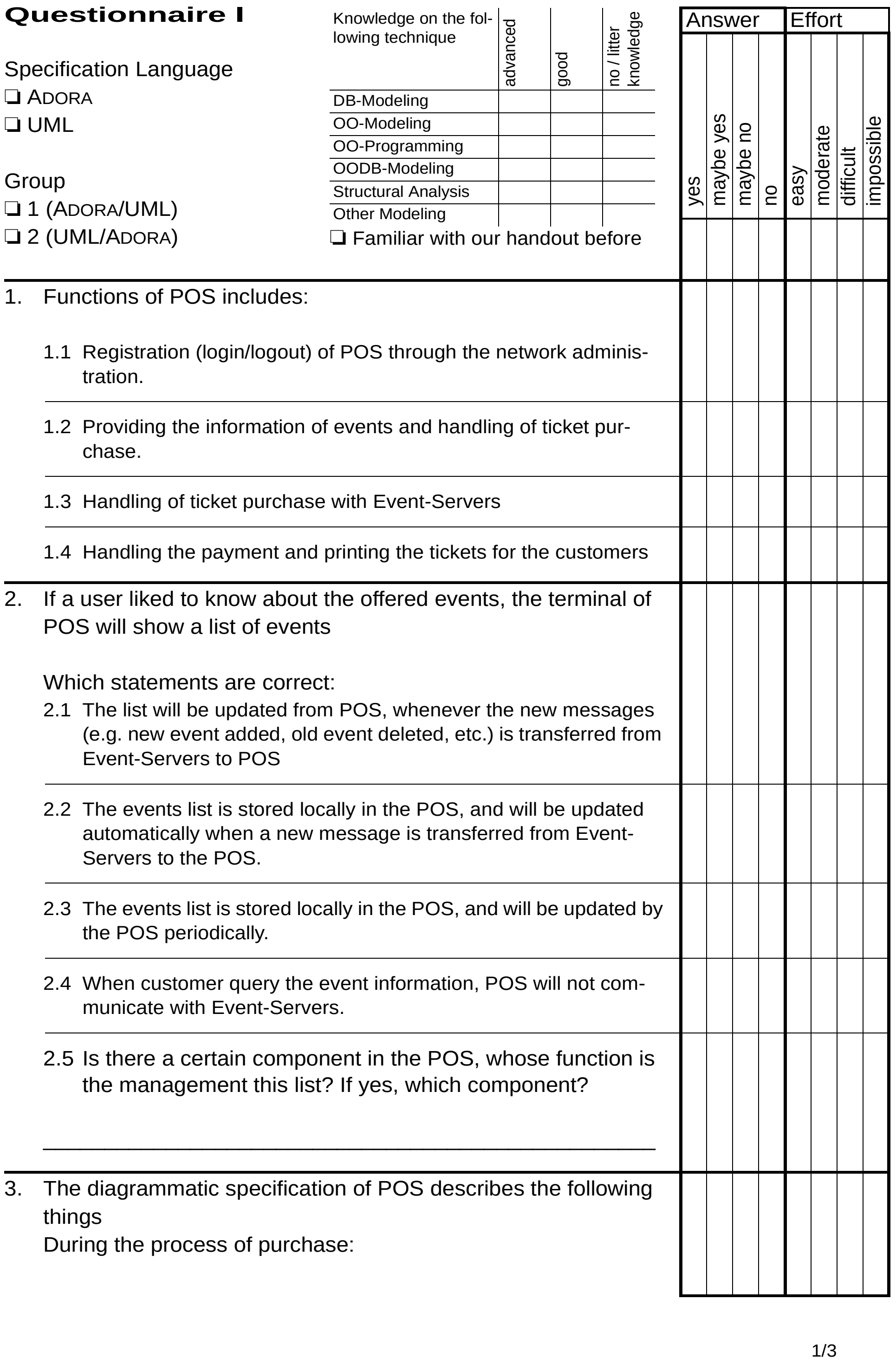
	Antwort				Aufwand			
	ja	wahrscheinlich ja	wahrscheinlich nein	nein	einfach	angemessen	schwierig	unmöglich
1. Die Funktionen einer Verkaufsstelle umfassen:								
1.1 Registrieren (Anmelden/Abmelden) der Verkaufsstelle bei der Netzwerk Administration.								
1.2 Bereitstellen einer Veranstaltungsübersicht und Abwicklung des eigentlichen Ticketkaufs.								
1.3 Abwicklung des Ticketkaufs mit dem/den Veranstaltungsserver(n).								
1.4 Abwicklung der Bezahlung und des Ticketdrucks.								
2. Wenn ein Benutzer sich über die angebotenen Ereignisse informieren möchte (Veranstaltungsübersicht), wird auf dem Terminal der Verkaufsstelle eine entsprechende Liste angezeigt. Was trifft zu:								
2.1 Diese Liste wird von der Verkaufsstelle jedesmal aufs Neue von den Veranstaltungsservern angefordert.								
2.2 Diese Liste ist lokal in der Verkaufsstelle gespeichert und wird bei einer Angebotsänderung automatisch aktualisiert.								
2.3 Diese Liste ist lokal in der Verkaufsstelle gespeichert und wird in bestimmten Zeitintervallen aktualisiert.								
2.4 Wenn ein Benutzer die Veranstaltungsübersicht anfordert kommuniziert die Verkaufsstelle nicht mit den Veranstaltungsservern.								
2.5 Gibt es eine bestimmte Komponente in der Verkaufsstelle, welche diese Liste bereitstellt? Wenn ja welche? ______________________________								
3. Die vorliegende Spezifikation einer Verkaufsstelle beschreibt folgenden Sachverhalt: Während eines Kaufvorgangs kann/können:								

1/3

	Antwort				Aufwand			
	ja	wahrscheinlich ja	wahrscheinlich nein	nein	einfach	angemessen	schwierig	unmöglich
3.1 beliebig viele Tickets (Platzkarten) für eine Veranstaltung erworben werden.								
3.2 nur ein einzelnes Ticket erworben werden; d.h. werden mehrere Tickets gewünscht, müssen mehrere Kaufvorgänge abgwickelt werden.								
4. Gibt es (eine) bestimmte Komponente(n) in einer Verkaufsstelle, die während des eigentlichen Ticketkaufs (nicht Angbotsabfrage) für die Kommunikation mit dem Veranstaltungsserver zuständig ist/sind. Wenn ja welche? ____								
5. Wann wird/werden beim (entsprechenden) Veranstaltungsserver der/die ausgewählte(n) Sitzplätze reserviert? 5.1 Nachdem der Benutzer alle gewünschten Sitzplätze ausgewählt hat, versucht die Verkaufsstelle all diese beim Veranstaltungsserver zu reservieren.								
5.2 Jeweils nachdem der Benutzer einen Sitzplatz ausgewählt hat, wird unmittelbar versucht, diesen beim Veranstaltungsserver zu reservieren.								
6. Ist es möglich, dass im Ticketing System: 6.1 keine Verkaufsstelle aktiv ist								
6.2 kein Veranstaltungsserver aktiv ist								
7. Eine Verkaufsstelle (ggf. die entsprechene Komponente) wendet sich zur Erstellung einer Veranstaltungsübersicht (Verzeichnis der angebotenen Veranstaltungen) an 7.1 alle Veranstaltungsserver								
7.2 genau einen Veranstaltungsserver								
7.3 die Netzwerkadministration								

	Antwort				Aufwand			
	ja	wahrscheinlich ja	wahrscheinlich nein	nein	einfach	angemessen	schwierig	unmöglich
8. Eine Verkaufsstelle wickelt den eigentlichen Ticketkauf (nicht Angbotsabfrage) mit								
8.1 genau einem Veranstaltungsserver ab								
8.2 mehreren Veranstaltungsserver ab								
9. Welche Aktivität gehört nicht zur Ticketabfrage und Ticketkauf:								
9.1 Ticketdruck (a)								
9.2 Anzeige der Veranstaltungsübersicht (b)								
9.3 Aktualisierung der Veranstaltungsübersicht (c)								
9.4 Sitzplatzauswahl/Reservierung beim Veranstaltungsserver (d)								
9.5 Veranstaltungsauswahl (e)								
9.6 Bezahlung (f)								
9.7 Wie ist die Abfolge der Aktivitäten während eines Ticketkaufs ______________________________								
10. Während des Ticketkaufs gibt es mindestens zwei (reguläre) Situationen (Zustände), in denen eine spezielle Fehlerbehandlung notwendig ist. Welche sind diese? ______________________________								
11. Wann genau kann der Benutzer (während eines Ticketkaufs) den Kaufvorgang nicht mehr abbrechen? ______________________________								

A4.2. Objective” questionnaire (Translation in English)



Questionnaire I

Specification Language
❑ ADORA
❑ UML

Group
❑ 1 (ADORA/UML)
❑ 2 (UML/ADORA)

Knowledge on the following technique	advanced	good	no / litter knowledge
DB-Modeling			
OO-Modeling			
OO-Programming			
OODB-Modeling			
Structural Analysis			
Other Modeling			

❑ Familiar with our handout before

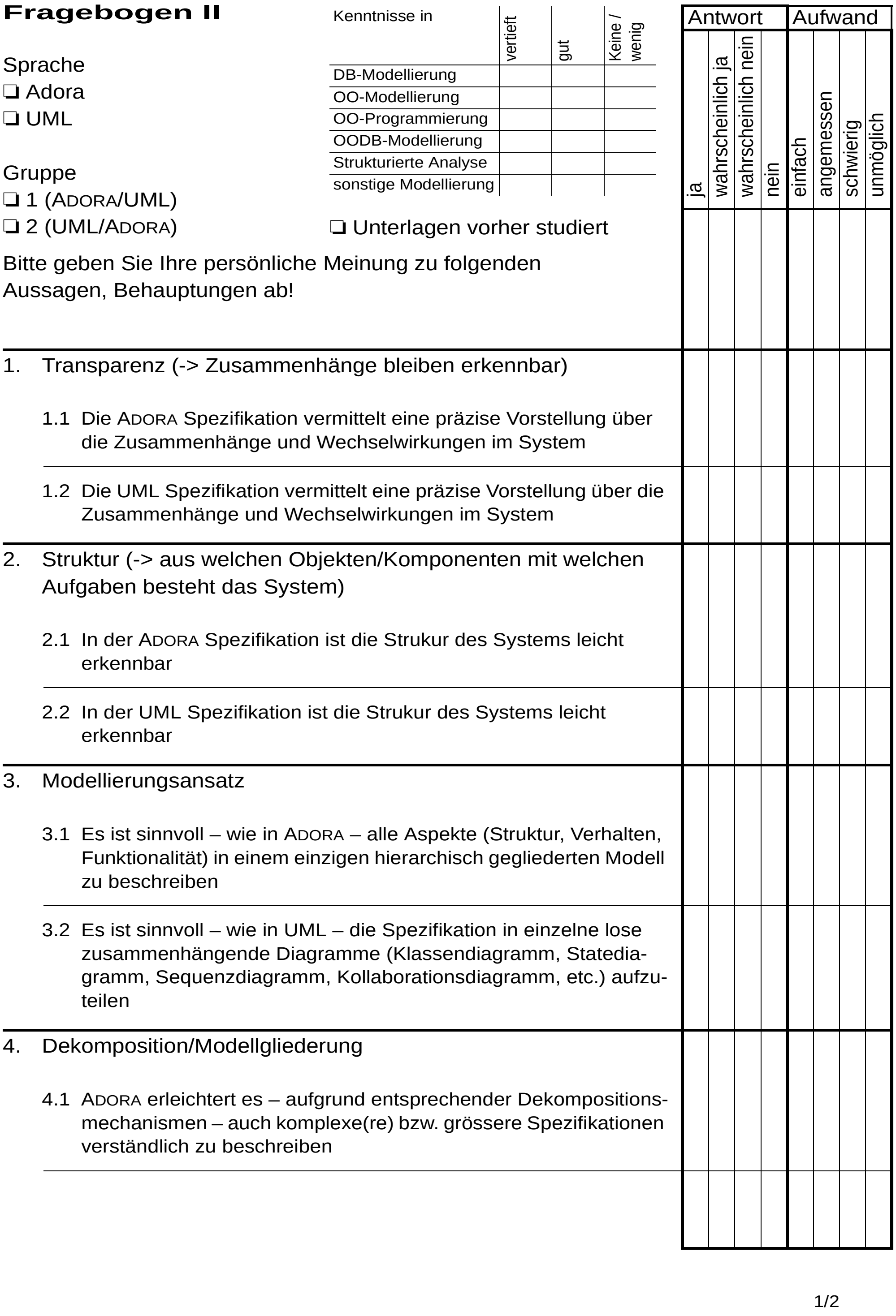
	Answer				Effort			
	yes	maybe yes	maybe no	no	easy	moderate	difficult	impossible
1. Functions of POS includes:								
1.1 Registration (login/logout) of POS through the network administration.								
1.2 Providing the information of events and handling of ticket purchase.								
1.3 Handling of ticket purchase with Event-Servers								
1.4 Handling the payment and printing the tickets for the customers								
2. If a user liked to know about the offered events, the terminal of POS will show a list of events Which statements are correct: 2.1 The list will be updated from POS, whenever the new messages (e.g. new event added, old event deleted, etc.) is transferred from Event-Servers to POS								
2.2 The events list is stored locally in the POS, and will be updated automatically when a new message is transferred from Event-Servers to the POS.								
2.3 The events list is stored locally in the POS, and will be updated by the POS periodically.								
2.4 When customer query the event information, POS will not communicate with Event-Servers.								
2.5 Is there a certain component in the POS, whose function is the management this list? If yes, which component? ______________________________								
3. The diagrammatic specification of POS describes the following things During the process of purchase:								

1/3

	Answer				Effort			
	yes	maybe yes	maybe no	no	easy	moderate	difficult	impossible
3.1 a customer can buy many tickets at one purchasing process								
3.2 a customer can only buy one ticket at one purchasing process. I.e. if she/he want to buy more tickets, she/he must start the purchasing process several times								
4. Is there a certain component in a POS, which is responsible for the communication with Event Servers, when customers buys a ticket (not querying events information)? If yes, which? ______________________________								
5. When will a seat/seats be reserved in the Event Servers? 5.1 After the customer selects all the seat, the POS will try to reserve those tickets in the corresponding Event Server								
5.2 Whenever the customer selects one single ticket, the POS will try to reserve that ticket in the corresponding Event Server								
6. Is it possible that in the ticketing system: 6.1 there is no active POS								
6.2 there is no active Event Server								
7. A POS (as the case maybe, an appropriate component) will generate a events list from 7.1 all the Event Server								
7.2 exactly one Event Server								
7.3 the NetworkAdministration								
8. One POS handles the ticket purchase (not events query) with 8.1 exactly one Event Server								
8.2 many Event Servers								

	Answer				Effort			
	yes	maybe yes	maybe no	no	easy	moderate	difficult	impossible
9. Which activities belong to the processes of events query and tickets purchase:								
9.1 printing tickets (a)								
9.2 showing the events list (b)								
9.3 updating the events list (c)								
9.4 selecting the seat(s) and reserving it (them) in the Event Server (d)								
9.5 selecting the favorite event (e)								
9.6 paying for the tickets (f)								
9.7 what is the sequence of the above activities, which belong to the processes of events query and tickets purchase. ____								
10. During the processes of tickets purchase, there are at least two situations in which we need to add some error- or exception-handling modules when we implement the system. Indicate them. ____								
11. At which step of the process of ticket purchase that the Customer can not abandon their operation? ____								

A4.3. Subjective” questionnaire (in German)



Fragebogen II

Sprache
❑ Adora
❑ UML

Gruppe
❑ 1 (Adora/UML)
❑ 2 (UML/Adora)

Kenntnisse in	vertieft	gut	Keine / wenig
DB-Modellierung			
OO-Modellierung			
OO-Programmierung			
OODB-Modellierung			
Strukturierte Analyse			
sonstige Modellierung			

❑ Unterlagen vorher studiert

Bitte geben Sie Ihre persönliche Meinung zu folgenden Aussagen, Behauptungen ab!

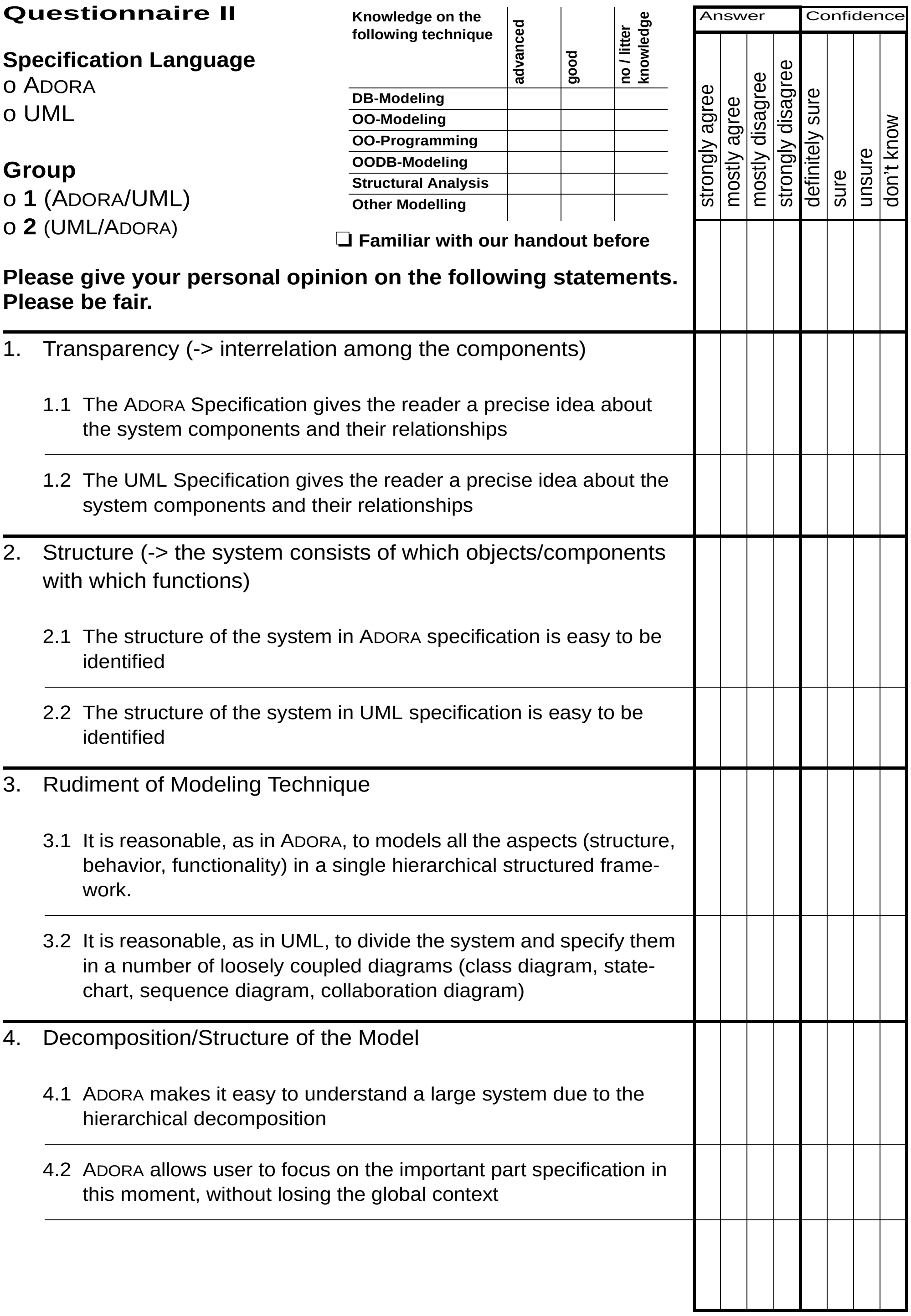
	Antwort				Aufwand			
	ja	wahrscheinlich ja	wahrscheinlich nein	nein	einfach	angemessen	schwierig	unmöglich
1. Transparenz (-> Zusammenhänge bleiben erkennbar)								
1.1 Die Adora Spezifikation vermittelt eine präzise Vorstellung über die Zusammenhänge und Wechselwirkungen im System								
1.2 Die UML Spezifikation vermittelt eine präzise Vorstellung über die Zusammenhänge und Wechselwirkungen im System								
2. Struktur (-> aus welchen Objekten/Komponenten mit welchen Aufgaben besteht das System)								
2.1 In der Adora Spezifikation ist die Strukur des Systems leicht erkennbar								
2.2 In der UML Spezifikation ist die Strukur des Systems leicht erkennbar								
3. Modellierungsansatz								
3.1 Es ist sinnvoll – wie in Adora – alle Aspekte (Struktur, Verhalten, Funktionalität) in einem einzigen hierarchisch gegliederten Modell zu beschreiben								
3.2 Es ist sinnvoll – wie in UML – die Spezifikation in einzelne lose zusammenhängende Diagramme (Klassendiagramm, Statediagramm, Sequenzdiagramm, Kollaborationsdiagramm, etc.) aufzuteilen								
4. Dekomposition/Modellgliederung								
4.1 Adora erleichtert es – aufgrund entsprechender Dekompositionsmechanismen – auch komplexe(re) bzw. grössere Spezifikationen verständlich zu beschreiben								

1/2

	Antwort				Aufwand			
	ja	wahrscheinlich ja	wahrscheinlich nein	nein	einfach	angemessen	schwierig	unmöglich
4.2 ADORA erlaubt es dem Benutzer, sich auf momentan wichtige Teile einer Spezifikation zu konzentrieren, ohne den globalen Zusammenhang zu verlieren								
4.3 Die hierarchische Gliederung eines ADORA-Modells erleichtert es, Informationen über bestimmte Systemteile zu finden								
4.4 In UML ist es einfach die gewünschten Informationen über bestimmte Systemteile aus den unterschiedlichen Diagrammen zusammenzutragen								
5. Kontext/Rollen								
5.1 Um ein System zu spezifizieren ist es angebracht, zu beschreiben welche *Rolle* Objekte einer Klasse haben bzw. in welchem Kontext diese Objekte benutzt werden								
5.2 Es reicht eigentlich aus, Klassen zu beschreiben								
6. Detaillierungsgrad								
6.1 Das ADORA-Modell des Beispielsystems ist eine geeignete Grundlage für nachfolgende Realisierungsschritte (Entwurf, Feinentwurf, Codierung, Test, etc.)								
6.2 Das UML-Modell des Beispielsystems ist eine geeignete Grundlage für nachfolgende Realisierungsschritte (Entwurf, Feinentwurf, Codierung, Test, etc.)								

2/2

A4.4. Subjective” questionnaire (Translation in English)



Questionnaire II

Specification Language
o ADORA
o UML

Group
o **1** (ADORA/UML)
o **2** (UML/ADORA)

Knowledge on the following technique	advanced	good	no / litter knowledge
DB-Modeling			
OO-Modeling			
OO-Programming			
OODB-Modeling			
Structural Analysis			
Other Modelling			

❑ **Familiar with our handout before**

Please give your personal opinion on the following statements. Please be fair.

	Answer				Confidence			
	strongly agree	mostly agree	mostly disagree	strongly disagree	definitely sure	sure	unsure	don't know
1. Transparency (-> interrelation among the components)								
1.1 The ADORA Specification gives the reader a precise idea about the system components and their relationships								
1.2 The UML Specification gives the reader a precise idea about the system components and their relationships								
2. Structure (-> the system consists of which objects/components with which functions)								
2.1 The structure of the system in ADORA specification is easy to be identified								
2.2 The structure of the system in UML specification is easy to be identified								
3. Rudiment of Modeling Technique								
3.1 It is reasonable, as in ADORA, to models all the aspects (structure, behavior, functionality) in a single hierarchical structured framework.								
3.2 It is reasonable, as in UML, to divide the system and specify them in a number of loosely coupled diagrams (class diagram, statechart, sequence diagram, collaboration diagram)								
4. Decomposition/Structure of the Model								
4.1 ADORA makes it easy to understand a large system due to the hierarchical decomposition								
4.2 ADORA allows user to focus on the important part specification in this moment, without losing the global context								

1/2

	Answer				Confi-			
	strongly agree	mostly agree	mostly disagree	strongly disagree	definitely sure	sure	unsure	don't know
4.3 The hierarchal structure of an ADORA-Model make it easy to find some specific information from the whole system								
4.4 It is easy to integrate information of some specific parts of the system from different diagrams.								
5. Context / Role								
5.1 It is advisable to specify objects with their roles and context								
5.2 Describing classes is sufficient.								
6. Degree of Detail								
6.1 The Adora model of the example system is a suitable basis for following implementation steps (Design, Detailed Design, Coding, Test, etc.)								
6.2 The UML model of the example system is a suitable basis for following implementation steps (Design, Detailed Design, Coding, Test, etc.)								

A5. Statistical Analysis of the Results of the Objective Questionnaire

In order to do the statistical analysis, we reorganize and simplify the Table 1 into the following tables:

Group 1 (ADORA/UML)							
ADORA				UML			
Right	sure	112 (77.2 %)	132 (91.0 %)	Right	sure	100 (65.8 %)	131 (86.2 %)
	unsure	20 (13.6 %)			unsure	31 (20.4 %)	
Wrong	sure	9 (6.2 %)	13 9.0 %	Wrong	sure	12 (7.9 %)	21 (13.8 %)
	unsure	4 (2.8 %)			unsure	9 (5.9 %)	

Table A.1

Group 2 (UML/ADORA)							
ADORA				UML			
Right	sure	103 (81.1 %)	113 (89.0 %)	Right	sure	75 (58.1 %)	99 (76.7 %)
	unsure	10 (7.9 %)			unsure	24 (18.6 %)	
Wrong	sure	9 (7.1 %)	14 (11.0 %)	Wrong	sure	21 (16.3 %)	30 (23.3 %)
	unsure	5 (3.9 %)			unsure	9 (7.0 %)	

Table A.2

Group 1 + Group 2							
ADORA				UML			
Right	sure	215 (79.0 %)	245 (90.1 %)	Right	sure	175 (62.3 %)	230 (81.9 %)
	unsure	30 (11.0 %)			unsure	55 (19.6. %)	
Wrong	sure	18 (6.6 %)	27 (9.9 %)	Wrong	sure	33 (11.7 %)	51 (18.1 %)
	unsure	9 (3.3 %)			unsure	18 (6.4 %)	

Table A.3

In this appendix, we give detailed mathematical proofs to the following statements:

- **Reading ADORA models is less prone to errors than reading UML models**.

  From the data in the above tables, it is obviously that the proportion of correctly reading ADORA model is higher than the proportion of correctly reading UML model. However, are the differences of the proportions significant? Or it just happens randomly?
  As we observe a sampling at size of 553 (15 participants totally gave 553 answers -- 272 of ADORA and 281 of UML), let us analyze the results using some statistical methods.
  Studying Table A.4, which is a simplified form of Table A.3, we use the method of Hypothesis Testing [6] to test whether the differences are significant.

Group 1 + Group 2			
ADORA		UML	
Right	245 ($\overline{p_1}$= 90.1 %)	Right	230 ($\overline{p_2}$ = 81.9 %)
Wrong	27	Wrong	51
Sum	n_1 = 245+27 = 272	Sum	n_2 = 230+51 = 281

Table A.4

We test the difference between two proportions, p_1 (the percentage of correctly answers of ADORA part) and p_2(the percentage of correctly answers of UML part). In Table 5, $\overline{p_1}$ is the percentage of correctly answers of ADORA part in our sampling data, and $\overline{p_2}$ is the percentage of correctly answers of UML part in our sampling data, where $\overline{p_1} = \frac{245}{245+27} = 90.1\%$ and $\overline{p_2} = \frac{230}{230+51} = 81.9\%$.

Our hypotheses are

$$H_0 : p_1 - p_2 = 0 \; ; \qquad H_1 : p_1 - p_2 > 0$$

The statistic is approximately normally distributed with a mean of 0 and a standard deviation of 1. Suppose we choose α = 0.5%.

Then for a one-tailed test our decision rule is: reject H_0 if Z > 2.58; accept H_0 if $Z \le 2.58$.

We calculate

$$Z = \frac{(\overline{p_1} - \overline{p_2}) - (p_1 - p_2)}{S_{\overline{p_1} - \overline{p_2}}} \,,$$

where $S_{\overline{p_1} - \overline{p_2}} = \sqrt{\frac{\overline{p_1} \cdot \overline{(1-p_1)}}{n_1} + \frac{\overline{p_2} \cdot \overline{(1-p_2)}}{n_2}}$ is the estimate of $\sigma_{\overline{p_1} - \overline{p_2}}$.

For the date of this problem,

$$Z = \frac{(0.901 - 0.819)}{\sqrt{\frac{0.901 \cdot (1 - 0.901)}{272} + \frac{0.819 \cdot (1 - 0.819)}{281}}} = 2.81$$

Since 2.81 > 2.58, we would reject H_0, and conclude that, from the consolidated results of Group 1 and Group 2, the proportion of correctly reading ADORA model is higher than the proportion of correctly reading UML model at a 0.5% level of significance[1].

[1] 0.5% level of significance is a very stringent requirement to refuse the null hypothesis. I.e. according to the statistical theory, we could already be confident that our judgement will be correct (very little chance to make Type I or Type II error) at a 1% level of significance or an even less stringent 5% level of significance.

We did a computer program implementing the above algorithm and calculated the statistic Z for the data of Group 1 and Group 2 separately too. The results are also fairly supportive to our judgment.

- **when participants correctly answer a question, they are more confident of themselves after reading the ADORA model**

  From the consolidated results of Group 1 and Group 2, using exactly the same way as above, we calculate the statistic Z = 3.32 (> 2.58). Therefore, we conclude that the proportion of answering questions correctly with confidence after reading ADORA model is higher than the proportion of answering questions correctly with confidence after reading UML model at a 0.5% level of significance[1].

- ***Though participants make a mistake, they are more confident that they are "correct", after reading the UML model.***

  From the consolidated results of Group 1 and Group 2, using the method of Hypothesis Testing, we calculate the statistic Z = 0.174 (< 2.58, < 2.33, < 1.65, <0.25[2]), we conclude that there is not a significant difference between the proportions of answering questions wrong with random guess reading two models at a 40% level of significance.
  Therefore, we think this statement can **not** be proved from our experiment.

[1] Similar conclusion can be drawn by calculating the statistic Z using data of Group 1 and Group 2 separately.

[2] When α = 0.005, the critical value is 2.58; when α = 0.01, that value is 2.33;
when α = 0.05, that value is 1.65; when α = 0.40, that value is 0.25.
Note: 40% level of significance is a very stringent requirement to accept the null hypothesis.