Available online at www.sciencedirect.com
ScienceDirect
ELSEVIER
Electronic Notes in Theoretical Computer Science 197 (2008) 15–29
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

Extending UML Sequence Diagrams to Model Trust-dependent Behavior With the Aim to Support Risk Analysis

Atle Refsdal[1,2]

Department of Informatics
University of Oslo, and
SINTEF ICT
Oslo, Norway

Ketil Stølen[3]

Department of Informatics
University of Oslo, and
SINTEF ICT
Oslo, Norway

Abstract

UML sequence diagrams are intuitively simple and can be understood by most stakeholders, including end-users, decision makers, engineers and other parties involved in a risk analysis. Building on UML sequence diagrams and trying to maintain their intuitive simplicity we propose a language for modeling systems where the trust considerations of actors play a major role. Trust considerations are integrated with behavioral descriptions in order to facilitate analysis of the trust considerations of the actors as well as their resulting behavior. We claim that our language allows trust dependent behavior to be described at a level of abstraction suitable for communication between different groups of stakeholders in a risk analysis situation. Furthermore, we argue that the increased expressiveness is required to facilitate the kind of analysis necessary to properly weigh and treat trust dependent risk behavior.

Keywords: Trust modeling, risk analysis, sequence diagrams.

1 Introduction

In a potentially hostile environment such as the internet, an actor needs to decide whether an entity can be trusted before engaging in any potentially harmful trans-

[1] The research on which this paper reports has been carried out within the context of the IKT-2010 project SARDAS (15295/431) and the IKT SOS project ENFORCE (164382/V30), both funded by the Research Council of Norway. Thanks to all the members of the ENFORCE project and the SARDAS project for comments on this work.

[2] Email: Atle.Refsdal@sintef.no

[3] Email: Ketil.Stolen@sintef.no

doi:10.1016/j.entcs.2007.12.014

action with the entity. When performing a risk analysis of a system where trust considerations play a major role, the system model must include information on how actors actually make decisions based on trust. This is illustrated by the example below.

OldGoods is a company selling antiques. Their business is based on buying old items such as furniture and watches on the internet and selling them from a fashionable shop at a much higher price. To this end they have hired a purchaser agent called Billy, whose job it is to search the internet for suitable items that can be bought and then sold from the shop. Billy finds a lot of different offers from various sites on the internet, and it is frequently required that the item is paid for before it is received. In such cases he needs to decide whether to send the payment. The decision, of course, depends on whether he trusts that the item will be shipped.

Business has not been good for OldGoods lately. A lot of money has been lost paying for items that never arrived. The management therefore decides that something has to be done. Someone suggests introducing one of the following two policy rules for the purchaser agent:

(i) "Do not pay for an item until it is received." This rule does not allow the purchaser agents' trust in the seller to play any role at all. It ensures that money will not be lost paying for items that are not received. But it also means that business opportunities are lost, since sellers (including the honest ones) may not accept this condition. Hence, many items that might give a good profit can not be acquired.

(ii) "Do not pay for an item in advance unless you have talked to the seller on the phone and feel confident that the item will be shipped as promised." This rule allows the purchaser agents' trust in the seller to play a major role in the decision. In order to make a decision the purchaser agent needs to estimate the probability that the seller will actually ship the item and decide whether this estimate amounts to feeling confident.

Should one of these rules be adopted? Or perhaps different rules or other measures would be better? These questions cannot be answered without a thorough understanding of the system. The first rule could be a good choice if most honest sellers were willing to ship items before they receive payment. The second rule may be a good choice if the purchaser agent is able to give reasonably correct probability estimates for the behavior of the sellers based on a phone conversation.

In this kind of risk analysis situation it may be necessary to interact with various stakeholders including end-users, decision makers as well as engineers. We have positive experience with the use of UML sequence diagrams [12] for this purpose [3],[16]. A UML sequence diagram is a specification of how messages are sent between entities to perform a task. Sequence diagrams seem to have the ability to be understood by professionals of computer systems design as well as potential end-users and stakeholders of the system in question, and are used in a number of different situations. They are used to get a better grip of an interaction scenario for an individual designer or for a group that needs to achieve a common under-

standing of the situation. Sequence diagrams are also used during more detailed design considerations where the precise inter-process communication must be set up according to formal protocols. Unlike for example state machines, sequence diagrams will typically not tell the complete story. There are normally other legal and possible behaviors that are not contained within the diagrams.

The contribution of this paper is the extension of the UML sequence diagram notation to allow trust dependent behavior to be described at the level of abstraction suitable for communication between different groups of stakeholders in a risk analysis situation. By trust dependent behavior we mean scenarios where an actor makes a decision about how to behave depending on the degree of trust the actor has in another entity. We claim that our language allows trust dependent behavior to be described at a level of abstraction suitable for communication between different groups of stakeholders in a risk analysis situation. Furthermore, we argue that the increased expressiveness is required to facilitate the kind of analysis necessary to properly weigh and treat trust dependent risk behavior.

The rest of the paper is organized as follows: In Section 2 we state the requirements to the modeling language. A brief introduction to a small subset of UML sequence diagrams is given in Section 3. In Section 4 we extend the UML sequence diagram notation with a construct for probabilistic choice. The definition of trust on which we base our work is presented in Section 5. In Section 6 we extend the UML sequence diagram notation further to capture trust-dependent behavior. In Section 7 we illustrate the new possibilities for analysis facilitated through the increased expressiveness. Some related work is presented in Section 8, before we conclude in Section 9.

2 Requirements to the trust modeling language

The overall goal is to develop a language facilitating in-depth analysis of systems whose critical behavior depends on trust, with the purpose of identifying vulnerabilities and treatments. A vulnerability in a trust-dependent system could typically be a decision to engage in a potentially harmful transaction made by an actor based on misplaced trust in another entity. During the analysis it should be possible to identify such a decision as well as the trust consideration behind the decision. Furthermore, it should be possible to quantify the likelihood of a harmful outcome. This is necessary to decide whether a treatment must be found or not, and the kind of treatment required.

A treatment in this context could typically be some kind of mechanism designed to control, restrict and support trust dependent behavior, for example a trust policy. Before deciding whether to implement a treatment it is necessary to assess its effect as well as cost.

Analysis requirements The language should facilitate analysis of

- systems whose critical behavior depends on trust;
- mechanisms designed to control, restrict and support trust dependent behavior.

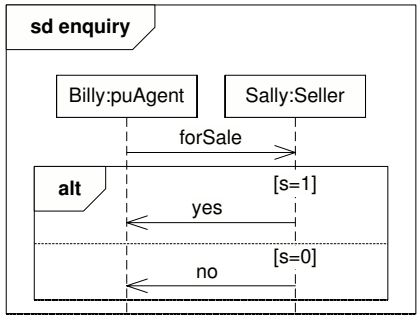

Fig. 1. UML sequence diagram

In order to facilitate in-depth analysis, it is necessary that all relevant aspects of the system can be expressed in the model. This includes the trust considerations that influence the agents' behavior as well as the behavior itself.

Expressiveness requirements It should be possible to

- express to what degree an agent trusts another agent with respect to a certain transaction;
- express how trust considerations influence a choice made by an agent between different courses of action;
- describe the behavior of the actors and the interaction between the actors.

In order to perform a successful analysis of the system it is important to involve various stakeholders such as end-users, decision makers and engineers. These groups will typically have very different backgrounds and levels of training, and the language should be a tool for arriving at a joint and correct understanding of the system (under the guidance of an analyst). Therefore the models expressed in the language must be intuitive and easily understandable for the involved parties.

Comprehensiveness requirement

- The language should facilitate communication between end-users, decision makers and engineers by being intuitively understandable by all these groups.

3 UML sequence diagrams

The UML [12] is widely used in the computer and software industry, and is seen as the de facto industry standard for system modeling. As explained above, UML sequence diagrams are used to show how entities in a system interact. The entities in question can be for example subsystems, components, pieces of software, or users. Figure 1 shows a UML sequence diagram representing a simple interaction between Billy and a seller called Sally. Each of these two entities is represented by a dashed vertical line called a lifeline. The box at the top of the lifeline may contain the name of the entity (before the colon) and its type (after the colon). Communication between the lifelines is shown by messages. These are represented by arrows pointing from the transmitter lifeline to the receiver lifeline, where the message content is given by the name above the arrow. Each message defines two events: a transmit

event occurring on the transmitter of the message and a receive event occurring on the receiver of the message (at the arrow head). For each lifeline the events are ordered in time from top to bottom. In addition, every message must be transmitted before it is received.

The diagram **enquiry** in Figure 1 shows a scenario where Billy asks Sally whether a certain item is for sale, as represented by the transmission of the "for-Sale" message. This message is received by Sally. The alt operator shows that there are then two possible alternatives (separated by the horizontal dashed line) that may occur [4]: Either Sally transmits the "yes" message, which is then received by Billy, or she transmits the "no" message, which is then received by Billy. The Boolean expressions in square brackets at the beginning of the two operands of the alt are called guards, and state conditions for the alternative to be chosen; Sally will respond with the "yes" message if $s = 1$ and with the "no" message if $s = 0$.

4 Extending UML sequence diagrams with probabilistic choice

STAIRS [6], [15], [14] gives a formal semantics for all the major operators of UML sequence diagrams, as well as a refinement calculus. Probabilistic STAIRS [13], [14] extends STAIRS with an operator palt for probabilistic choice, as well as extending the semantic model of STAIRS to include probabilities. For the purpose of this paper, however, it is sufficient to keep the discussion at the syntactic level. Definitions of formal semantics and refinement relations can be found in the papers referenced above.

We now introduce the probabilistic sequence diagram notation based on the OldGoods example. To assess the current situation the management has obtained a specification that describes what happens in the current purchasing system after a suitable item has been found by the purchaser agent. We may assume this specification is based on historical data, so that the probabilities in the specification reflect percentages of the observed behavior. The established practice is that the purchaser agent asks the seller a test question on the phone before deciding whether to send the payment. This is done in order to assess whether the seller will actually send the item if she receives advance payment. If the purchaser agent is not happy with the answer then advance payment will not be sent. The specification **purchase** in Figure 2 shows what takes place [5]. The purchaser agent starts by asking the seller a question on the phone. After receiving the reply, he decides whether to send advance payment or cancel the deal. The choice between these alternatives is expressed by the palt operator (for probabilistic alternative), which has two or more operands separated by a dotted line. Each operand expresses one possible alternative. The lower operand of the outermost palt shows the alternative where

[4] If using STAIRS we would have used the xalt operator instead of the alt. Since xalt is not part of the UML we have chosen to use alt in the example.

[5] Sequence diagrams allow specification of both negative and positive behavior. Negative behavior is behavior that the system is not allowed to produce. In order to get keep the diagrams simple we only specify positive behavior in this paper. This is sufficient for our explanations at the intuitive level.

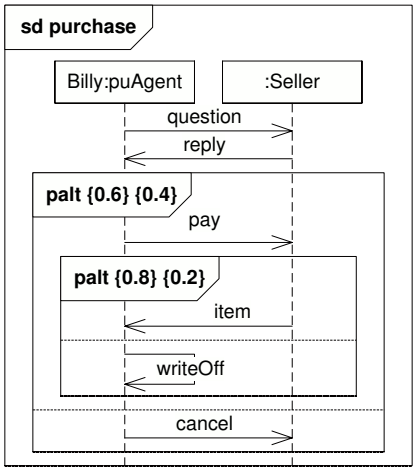

Fig. 2. The interaction between the purchaser agent and a seller expressed in probabilistic STAIRS

Billy cancels the deal after receiving the reply to his question. This alternative has probability 0.4, which can be seen in the upper left part of the palt operand. Probabilities are shown from left to right in the order of the operands, so the first (uppermost) operand of the outermost palt has probability 0.6, while the second operand has probability 0.4.

If Billy chooses to send the payment then again one of two things may happen: either the seller sends the item (with probability 0.8), or she does not (with probability 0.2). If she does not then Billy must write off the money, represented by the message from Billy to himself.

Clearly, trust affects the behavior of Billy. However, there is no explicit representation of the way he makes his choice of behavior based on trust. Our objective is to represent this explicitly in the specification. But first we need to define what we mean by trust.

5 Trust

We use the definition of trust given in [11] [6]. This definition is an adaption from [10], which is based on [4].

Definition 5.1 Trust is the *subjective probability* by which an actor, the trustor, expects that another entity, the trustee, performs a given transaction on which its welfare depends.

Thus defined, trust is a belief of the trustor regarding the behavior of the trustee. Since trust is a belief it is a subjective notion. An actor is an active entity which has goals, intentions and capabilities. An actor may be an organization, a human or an automated artefact such as hardware and software. Often a trustor will only expect the trustee to perform a transaction if a certain scenario takes place. We call this the *antecedent scenario* of the trust with respect to the transaction.

[6] We have used the word 'transaction' instead of 'transition'.

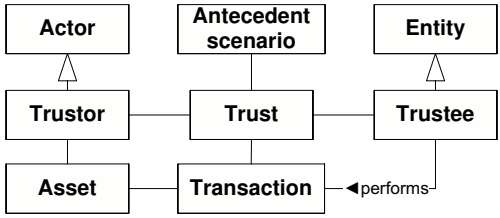

Fig. 3. Trust and related concepts

Figure 3 is a slightly simplified version of Figure 37 from [11] and shows the relation between concepts of relevance to trust in the UML class diagram notation. If the antecedent scenario takes place then the trustor trusts to some degree that the trustee performs a certain transaction, i.e. the trustor believes that there is a certain probability that the trustee will perform the transaction. An asset is something of value to the owner (the trustor). That the welfare of the trustor depends on the transaction is shown by the relationship between the transaction and the trustor's asset; the value of the asset depends on whether the transaction is performed. The trustor is an actor, which is shown by the generalization relationship (open arrowhead). The trustee may be any kind of entity, including an actor.

In our example Billy would be the trustor, while the seller would be the trustee. The antecedent scenario would be that Billy sends advance payment after the phone conversation, and the transaction would be that the seller sends the item after receiving payment. If Billy believes that the probability that the seller will send the item is 0.9, then this means that Billys trust in the seller with respect to this transaction is 0.9. The asset in question would be the combination of the items received by the purchaser agent and the money he has available for purchasing new items. If an item is paid for, but not received, then this asset will decrease in value.

6 Extending UML sequence diagrams with a notion of trust

In our example we need to know to what degree the purchaser agent trusts a seller to send the item after receiving advance payment, i.e. the subjective probability assigned to this outcome by the purchaser agent. We also need to know how this trust influences his decision whether to send advance payment or not.

6.1 Subjective sequence diagrams

Since trust is defined as a subjective probability for a certain alternative to occur, we may express trust by a probabilistic sequence diagram simply by letting the sequence diagram represent an agents' subjective belief or estimate rather than an objective description of the system. To show that a diagram is subjective we write **ssd** (for subjective sequence diagram) in front of the diagram name instead of **sd**. In addition, we write **subj** in the lifeline head of exactly one lifeline to show that this is the subject, i.e. the lifeline whose subjective belief is captured by the diagram. Subjective diagrams can only be composed if their subjects are identical.

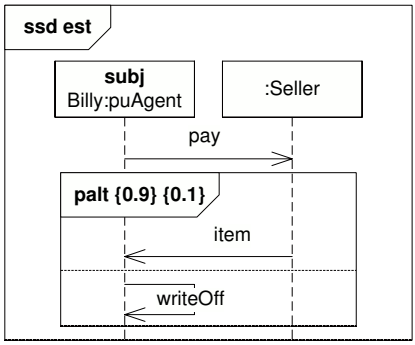

Fig. 4. Billys trust in the seller with respect to sending the item after receiving advance payment

For example, assume that Billy after the phone conversation with the seller believes the probability is 0.9 that she will send the item if he pays in advance. This belief can be expressed by the specification **est** (for estimate) in Figure 4. With respect to the scenario described by **est** Billy believes that the probability is 0.9 that the seller sends the item after receiving payment. He knows that if the item is not received then he will write off the money; therefore he believes that the probability is 0.1 that the money will be written off. The specification does not say anything about Billy's belief about scenarios not described, such as payment not being sent.

In order to express how Billy's trust relates to the overall system behavior, we need to show how the subjective diagram **est** representing Billy's trust relates to the objective diagram.

6.2 Including trust considerations in the system specification

We are now in position to give a more detailed description of the system described in Figure 2, where also relevant details concerning the purchaser agent's trust are expressed explicitly. Firstly, by using a subjective sequence diagram we may express what probability estimate is made by the purchaser agent before he decides whether to send advance payment or cancel the deal. Secondly, by the use of guards we may express how this probability estimate determines his choice.

Figure 5 shows the system with explicit information about the trust considerations made by Billy after a phone conversation with a seller. This specification does not represent one particular interaction occurrence with one particular seller; instead it represents a general interaction where the objective probabilities would typically be based on historical data. Since the subjective probability estimates given by Billy varies from seller to seller, we use the variable x in the subjective diagram instead of a constant. This variable can be used in the objective diagram to show how the estimate determines Billy's choice of whether to send advance payment. The notation **(out x)** after the diagram name in the subjective diagram is used to declare that this variable can be referred to from an objective diagram. To refer to the final value of a variable v in a subjective diagram **d** from an objective diagram we use the notation $\mathbf{d}.v$, which means that $\mathbf{est}.x$ refers to the variable x in

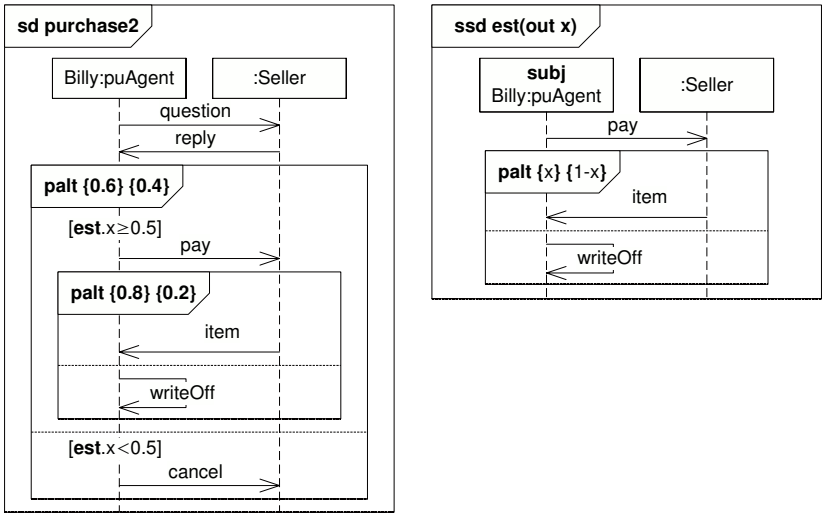

Fig. 5. A specification showing how Billy makes his choice

the subjective diagram **est** at its point of termination. Hence, in cases where the value of a variable v changes in the diagram $\mathbf{d}$, the expression $\mathbf{d}.v$ will evaluate to the final value of v.

The variable x in the diagram **est** represents the estimated probability of receiving the item if sending advance payment according to Billy. The guards in the operands of the outermost palt show that Billy will send advance payment only if he believes that the probability of receiving the item is at least 0.5. The probability of the operand where this holds (i.e. $\mathbf{est}.x \geq 0.5$) is 0.6. This means that in 60% of the cases Billy believes that the probability of receiving the item is at least 0.5.

7 Analyzing systems with trust-dependent behavior

Figure 5 gives already a rough analysis of the current system. From this specification we see that Billy sends advance payment in 60% of the cases. Of these, the item will be received in 80% of the cases. It follows that out of all the items considered, 48% will be paid for and received, while 12% will be paid for but not received. We now demonstrate how models can be used in a more detailed analysis of a trust-dependent system. We focus only on the issues specific for trust dependent systems: subjective probability estimates and decisions based on such estimates.

From the specification **purchase2** in Figure 5 we see that the following two components determines whether Billy will send advance payment to the seller:

- Billys probability estimate, and
- the threshold value of 0.5 that the estimated probability of receiving the item has to meet for Billy to be willing to send the advance payment.

To evaluate the impact of subjective probability estimates, two questions need to be answered. The first is: How accurate are the estimates? We need to know this in order to decide if it is acceptable to base decisions on the existing method of making

estimates. If the subjective probability estimates are not sufficiently close to the objective probabilities, then either decisions should not be based on the subjective estimates or some way of improving the accuracy of the estimates must be found.

The second question we need to answer is how the actor acts based on a probability estimate, or more specifically: Is the threshold right? A good probability estimate is of little use if the actor engages in a potentially harmful transaction despite his belief that the probability of being harmed is very high. On the other hand, if the actor is not willing to engage in the transaction unless he believes that the probability of being harmed is extremely low, then business opportunities may be lost.

7.1 Assessing subjective probability estimates

In order to assess the accuracy of subjective probability estimates we need to know the estimated probabilities as well as the objective probabilities for all cases. This requires a description of the system that shows what will happen if the trustor engages in the potentially harmful transaction no matter what his probability estimate is; otherwise the objective probabilities for the case where the trustor decides not to engage in the transaction could not be shown. Consider the specification in Figure 5. From this specification we cannot tell what would be the probability of receiving an item from those sellers that the purchaser agent believes are least likely to send the item. Of course it may be that the probability of receiving items from these sellers would actually be very high; they just have not been given the chance to prove it.

Figure 6 shows a specification from which we can evaluate Billy's probability estimates [7]. In Figure 6 it is assumed that Billy sends advance payment in *all* cases. The specification could for example be based on an experiment where Billy actually accepts all offers for a certain period of time, or possibly on some expert's judgment.

In addition to assuming that Billy always sends advance payment, we also distinguish between three different intervals of estimated probability in Figure 6, instead of just two as in Figure 5. For each interval we have a separate palt operand for the case where Billy's estimate lies within the interval. Thus the specification shows that the probability of receiving the item from a seller that Billy has estimated will send the item with a probability in the interval $[0.8, 1]$ is 0.9 (from **pay1**), while the probability of receiving the item from a seller that Billy has estimated will send the item with a probability in the interval $[0.5, 0.8\rangle$ is 0.75 (from **pay2**). From a seller that Billy has estimated will send the item with a probability in the interval $[0, 0.5\rangle$ the probability is actually 0.6 (from **pay3**).

Splitting up the cases as in Figure 6 gives a better picture of how good Billy's estimates really are, and is useful as a means to identify the optimal threshold for sending advance payment, as will be demonstrated below. We have chosen to use

[7] The ref construct used in Figure 6 is a reference to the diagram whose name occurs inside the frame. Its meaning is the same as if the contents of the referenced diagram was inserted in place of the ref construct. The ref construct allows a modular presentation of diagrams, as well as reuse of diagrams.

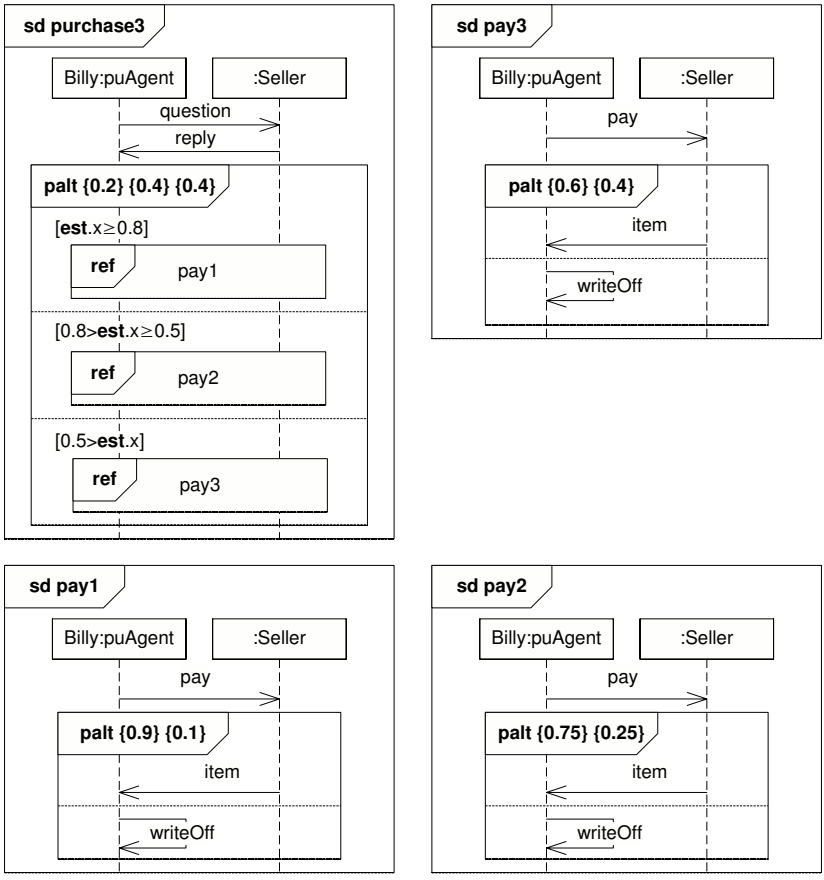

Fig. 6. A specification from which we can evaluate the accuracy of Billy's probability estimates

three different cases in order to keep the specification simple, but any finite number of cases could of course be used, depending on the desired granularity of the analysis.

7.2 Finding the right threshold

We now explain how to identify the optimal threshold, against which Billy should compare his estimated probability when deciding whether to send advance payment, based on what we know about his subjective probability estimates as illustrated in Figure 6. The desire to buy as many items as possible (since these will generate income) must be balanced against the desire to minimize loss from items that are paid for but not received. We therefore want to know how many items out of the total number considered will be paid for and received, and how many will be paid for but not received, depending on the chosen threshold. This information is easily obtained from Figure 6. The table in Figure 7 shows the results. There is one row for (the lower bound of) each of the three estimate intervals that has been considered in Figure 6.

The Paid column shows the number of items that will be paid for if the corresponding guard (in the leftmost column) is used. Consider for example the row

Guard	Paid	Received	Lost
est.$x \geq 0.8$	20%	18%	2%
est.$x \geq 0.5$	60%	48%	12%
est.$x \geq 0$	100%	72%	28%

Fig. 7. Results of using different thresholds. All numbers are given as percentages of the total number of items considered.

where the guard is **est**.$x \geq 0.5$. From Figure 6 it is clear that the estimated probability is 0.8 or higher in 20% of the cases and between 0.5 and 0.8 in 40% of the cases. Hence the estimated probability is 0.5 or higher in 60% of the cases.

The Received column shows the number of items that will be received if the corresponding guard is used. This number is found from the probabilities of the palt operands where the guard is fulfilled, together with the probabilities of receiving the item in these cases. Again consider the row where the guard is **est**.$x \geq 0.5$, which corresponds to the first two operands of the palt operator in the purchase specification in Figure 6. The probability of the first operand is 0.2, and the probability of receiving the item in this case (as shown by **pay1**) is 0.9. The probability of the second operand is 0.4, and the probability of receiving the item in this case (as shown by **pay2**) is 0.75. Hence, the number in the Received column is $0.2 * 0.9 + 0.4 * 0.75 = 0.48 = 48\%$.

The Lost column shows the number of items that are paid for but not received; it is the difference between the paid items and the received items.

By combining the information from Figure 7 with information about how much money will be lost or gained in the different scenarios, the analysts have a good basis from which to decide what is the best threshold to use [8].

8 Related work

We are not aware of other languages where subjective probability estimates are integrated in specifications of behavior along with objective probabilities. In the literature there is, however, much work on uncertainty, belief and trust.

Subjective logic [7],[8] is a probabilistic logic that explicitly takes uncertainty about probability values into account. The logic operates on subjective belief about the world. Different actors have different subjective beliefs, and these beliefs are associated with uncertainty. In subjective logic it is for example possible to calculate to what degree an actor believes that a system will work based on the actor's beliefs about the subsystems, or to calculate the consensus opinion of a group of actors. Subjective logic deals strictly with the actors' beliefs and reasoning, and there is no representation of how this reasoning influences their behavior.

In [9] it is shown how to use the belief calculus of subjective logic in a risk

[8] As noted in Section 8, information about asset values could be integrated in our model by following the approach of [1].

analysis. Subjective beliefs about threats and vulnerabilities are used as input parameters to the analysis. Hence, the computed risk assessments will also contain information about the uncertainty associated with the result of the analysis.

Epistemic logics are modal logics concerned with reasoning about belief. A modal belief-operator is used to express statements like "actor A believes P". BAN logic [2] is an epistemic logic for analyzing communication protocols and authentication. The belief operator can be used for example to express that two actors believes that they are indeed communicating with each other (and not with an intruder).

A formal framework for modeling and analyzing security and trust requirements is presented in [5]. Focus is on modeling organizations (which may include computer systems). The approach is based on a separation of functional dependencies, trust and delegation relationships, which allows security and trust requirements to be captured without going into details about how these will be realized. The formal framework supports automatic verification of security and trust requirements.

In [1] a semantic paradigm for component-based specification supporting the documentation of security risk behavior is proposed. Probabilistic sequence diagrams are used to express the probability of unwanted scenarios. Assets and their values are modeled explicitly as lifelines that receives messages when their value changes. Alternatively, assets could have been represented by variables that are assigned new values as the asset value changes. Explicit representation of assets (either as lifelines or variables) can also be included in our models in the same way; this will be highly useful for example when evaluating the cost and benefit of a treatment.

9 Conclusion

We have presented a language designed to support risk analysis of trust-dependent systems. In Section 2 we stated the requirements that such a language should fulfill. We now argue that these requirements have been fulfilled

Analysis requirements

- The language facilitates analysis of systems whose critical behavior depends on trust by offering models where trust considerations (subjective probability estimates) and decisions based on trust considerations are modeled explicitly along with system behavior. Trust considerations are represented by subjective sequence diagrams, while decisions based on trust are represented by guards referring to subjective sequence diagrams. This makes it easy to recognize trust considerations and decisions based on trust considerations in a model. Hence it is easier to identify vulnerabilities and treatments related to such considerations and decisions, and to find treatments.
- Analysis of mechanisms designed to control, restrict and support trust dependent behavior is facilitated since models can be built of systems where such mechanisms are (assumed to be) implemented. We may then obtain two models of the same system: one where the mechanism is implemented and one where it is not implemented. The effect of the mechanism can be evaluated by

comparing probabilities for desired and undesired outcomes in the two models.

Expressiveness requirements

- To what degree an agent trusts another agent with respect to a certain transaction can be expressed by probabilities in a subjective sequence diagram.
- How trust considerations influence a choice made by an agent between different courses of action can be expressed by a guard referring to a subjective sequence diagram.
- The behavior of the actors and the interaction between them can be expressed by an objective sequence diagram.

Comprehensiveness requirement

- We have positive experience from using sequence diagrams to facilitate communication between end-users, decision makers and engineers during risk analysis [3],[16]. The language presented in this paper is a conservative extension of UML sequence diagrams where only a few new constructs (probability on alternatives and subjective diagrams) have been added. We therefore have strong reason to believe that also our language will facilitate communication and be intuitively understandable by the persons taking part in the analysis.

As future work we intend to add the possibility to express an actor's uncertainty about his subjective probability estimates. When giving an estimate, an actor may be more or less certain that the estimate is correct. This could be modeled as a second-order subjective probability, where the value 1 means that the actor is certain that the estimate is correct. Such second-order probabilities could be assigned either for each subjective diagram or for each palt operator in a subjective diagram, and these probabilities could be referred to in guards along with the first-order subjective probabilities. Hence we may express statements such as "the purchaser agent will not send advance payment unless he believes that the probability of receiving the item is at least 0.5 *and* he believes that his estimate is correct with a probability of at least 0.9".

References

[1] Brændeland, G. and K. Stølen, *Using model-based security analysis in component-oriented system development*, in: *Proceedings of Quality of Protection (QoP)* (2006), pp. 11–18.

[2] Burrows, M., M. Abadi and R. Needham, *A logic of authentication*, ACM Transactions on Computer Systems **8** (1990), pp. 18–36.

[3] den Braber, F., A. B. Mildal, J. Nes, K. Stølen and F. Vraalsen, *Experiences from using the CORAS methodology to analyze a web application*, Journal of Cases in Information Technology **7** (2005), pp. 110–130.

[4] Gambetta, D., *Can we trust trust?*, in: D. Gambetta, editor, *Trust: Making and Breaking Cooperative Relations*, Basil Blackwell, Oxford, 1988 pp. 213–237.

[5] Giorgini, P., F. Massacci, J. Mylopoulos and N. Zannone, *Requirements engineering meets trust management: Model, methodology, and reasoning*, in: *Trust Management*, LNCS **2995** (2004), pp. 176–190.

[6] Haugen, Ø., K. E. Husa, R. K. Runde and K. Stølen, *STAIRS towards formal design with sequence diagrams*, Journal of Software and Systems Modeling **22** (2005), pp. 349–458.

[7] Jøsang, A., *A logic for uncertain probabilities*, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems **9** (2001), pp. 279–311.

[8] Jøsang, A., *Probabilistic logic under uncertainty*, in: *Thirteenth Computing: The Australasian Theory Symposium (CATS2007)*, Conferences in Research and Practice in Information Technology (CRPIT) **65** (2007), pp. 101–110.

[9] Jøsang, A., D. Bradley and S. J. Knapskog, *Belief-based risk analysis*, in: *Proceedings of the Australasian Information Security Workshop (AISW)*, Conferences in Research and Practice in Information Technology (CRPIT) **32** (2004), pp. 63–68.

[10] Jøsang, A., C. Keser and T. Dimitrakos, *Can we manage trust?*, in: *Trust Management*, LNCS **3477** (2005), pp. 93–107.

[11] Lysemose, T., T. Mahler, B. Solhaug, J. Bing, D. Elgesem and K. Stølen, *ENFORCE conceptual framework*, Technical Report A1209, SINTEF ICT (2007).

[12] Object Management Group, "UML 2.1 Superstructure Specification," Document: ptc/06-04-02 edition (2006).

[13] Refsdal, A., K. E. Husa and K. Stølen, *Specification and refinement of soft real-time requirements using sequence diagrams*, in: *Proceedings of Formal Modeling and Analysis of Timed Systems (FORMATS)*, LNCS **3829** (2005), pp. 32–48.

[14] Refsdal, A., R. K. Runde and K. Stølen, *Underspecification, inherent nondeterminism and probability in sequence diagrams*, in: *Proceedings of Formal Methods for Open Object-Based Distributed Systems (FMOODS)*, LNCS **4037** (2006), pp. 138–155.

[15] Runde, R., Ø. Haugen and K. Stølen, *Refining UML interactions with underspecification and nondeterminism*, Nordic Journal of Computing **12(2)** (2005), pp. 157–188.

[16] Stamatiou, Y. C., E. Henriksen, M. S. Lund, E. Mantzouranis, M. Psarros, E. Skipenes, N. Stathiakis and K. Stølen, *Experiences from using model-based risk assessment to evaluate the security of a telemedicine application*, in: *Proceedings of Telemedicine in Care Delivery (TICD)*, 2002, pp. 115–119.