# A Problem-Oriented Approach to Common Criteria Certification

Thomas Rottke[1], Denis Hatebur[1], Maritta Heisel[2], and Monika Heiner[3]

[1] TÜViT GmbH, System- und Softwarequalität
Am Technologiepark 1, 45032 Essen, Germany
{t.rottke,d.hatebur}@tuvit.de

[2] Institut für Praktische Informatik und Medieninformatik
Technische Universität Ilmenau
98693 Ilmenau, Germany
maritta.heisel@tu-ilmenau.de

[3] Brandenburgische Technische Universität Cottbus, Institut für Informatik
03013 Cottbus, Germany
mh@informatik.tu-cottbus.de

**Abstract.** There is an increasing demand to certify the security of systems according to the Common Criteria (CC). The CC distinguish several evaluation assurance levels (EALs), level EAL7 being the highest and requiring the application of formal techniques. We present a method for requirements engineering and (semi-formal and formal) modeling of systems to be certified according to the higher evaluation assurance levels of the CC. The method is problem oriented, i.e. it is driven by the environment in which the system will operate and by a mission statement. We illustrate our approach by an industrial case study, namely an electronic purse card (EPC) to be implemented on a Java Smart Card. As a novelty, we treat the mutual asymmetric authentication of the card and the terminal into which the card is inserted.

## 1 Introduction

In daily life, security-critical systems play a more and more important role. For example, smart cards are used for an increasing number of purposes, and e-commerce and other security-critical internet activities become increasingly common. As a consequence, there is a growing demand to certify the security of systems.

The common criteria (CC) [1] are an international standard that is used to assess the security of IT products and systems. The CC distinguish several evaluation assurance levels (EALs), level EAL7 being the highest and requiring the application of formal techniques even in the high-level design.

Whereas the CC state conditions to be met by secure systems, they do not assist in constructing the systems in such a way that the criteria are met. In this paper, we present a method for requirements engineering and (semi-formal or formal) modeling of systems to be certified according to the higher evaluation

© Springer-Verlag Berlin Heidelberg 2002

assurance levels of the CC. This method is used by TÜViT Essen[1] in supporting product evaluations.

The distinguishing feature of our method is its *problem orientation*. First, problem orientation means that the starting point of the system modeling is an explicit *mission statement*, which is expressed in terms of the application domain. This approach is well established in systems engineering [3], but in contrast to other requirements engineering approaches, especially in software engineering [9]. Such a mission statement consists of the following parts:

1. external actors and processes
2. objective/mission of the system
3. system services
4. quality of services

From our experience, the mission statement provides the main criteria for assessing, prioritizing and interpreting requirements.

Second, problem orientation means that we do not only model the system to be constructed, but also its environment, as proposed by Jackson [6]. This approach has several advantages:

– Without modeling the environment, only trivial security properties can be assessed.
  For example, an intruder who belongs to the environment must be taken into account to demonstrate that a system is secure against certain attacks.
– With the model of the environment, a test system can be constructed at the same time as the system itself.
– The problem oriented approach results in a strong correspondence between the reality and the model, which greatly enhances validation and verification activities as they are required for certification.

In the CC, the environment of the system is contained only indirectly via subjects, which must be part of the security policy model. Another difference to the CC is that our method not only takes into account the new system to be constructed, but also performs an analysis of the current system in its environment.

Figure 1 shows the most important documents that have to be constructed and evaluated for certification:

– The objective of the security target (ST) is to specify the desired security properties of the system in question by means of security requirements and assurance measures.
– The security policy model (SPM) shows the interaction between the system and its environment. This model provides a correspondence between the functional requirements and the functional specification enforced by the security policy.

[1] TÜViT is an independent organization that performs IT safety and security evaluation.

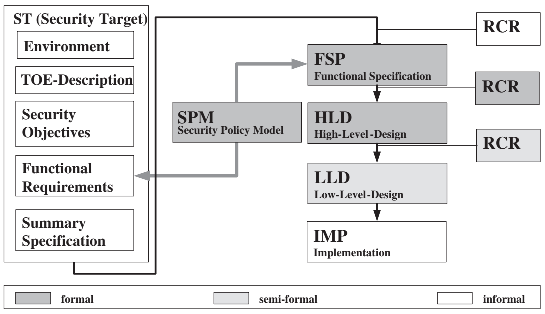

**Fig. 1.** CC documents for development

– The functional specification (FSP), high-level-design (HLD), low-level-design (LLD) and implementation (IMP) are development documents that are subject to evaluation.
– In addition to the development documents, representation correspondences (RCR) documentation is required to ensure that appropriate refinement steps have been performed in the development documents.

In the following, we describe our problem oriented approach in Section 2, and then illustrate it in Section 3 by an industrial case study, namely an EPC to be implemented on a Java Smart Card. As a novelty, we treat the mutual asymmetric authentication of the card and the terminal into which the card is inserted. In this case study, we use the notations SDL [2], message sequence charts (MSCs) [5], and colored Petri nets [7]. Finally, we sum up the merits of our method and point out directions for future work.

## 2 The Method

Our method gives guidance how to develop the documents required for a CC certification in a systematic way. Because of the systematic development and the use of semi-formal and formal notations, the developed documents can readily be evaluated for conformance with the CC.

To express our method, we use the *agenda* concept [4]. An agenda is a list of steps or phases to be performed when carrying out some task in the context of software engineering. The result of the task will be a document expressed in some language. Agendas contain informal descriptions of the steps, which may depend on each other. Agendas are not only a means to guide software development activities. They also support quality assurance because the steps may have validation conditions associated with them. These validation conditions state necessary semantic conditions that the developed artifact must fulfill in order to serve its purpose properly.

Table 1 gives an overview of the method. Note that the method does not terminate with Phase 4. There are two more phases that are beyond the scope

**Table 1.** Agenda for problem oriented requirements engineering and system modeling

| Phase | Content | Format | CC Documents | Validation |
|---|---|---|---|---|
| **1.** Problem oriented requirements capture | list of requirements | informal | — | reviews |
| **2.** Analysis of current system | description of current system status | informal or semi-formal | — | reviews |
| **3.** Problem oriented requirements analysis | description of desired system status, mission statement | informal or semi-formal | ST: environment, TOE description,security objectives | each statement of phase 1 must be incorporated; internal consistency must be guaranteed. |
| **4.** Problem oriented modeling | context diagram, system interface descriptions, system environment description | possibly formal | ST: functional requirements, summary specification; FSP, SPM | see sub-agenda, Table 2. |

of this paper. In Phase 5, the model constructed in Phase 4 is validated. Finally, the model is refined by constructing a high-level design, a low-level design and an implementation (Phase 6). In this paper, however, we concentrate on the systematic development of the requirements and specification documents. Validation and refinement issues will be treated in separate papers.

Setting up the documents required by the CC need not necessarily proceed in the order prescribed by the CC outline. Our process proceeds in a slightly different order. The “CC Documents” column in Table 1 shows in which phases which CC documents are developed.

The purpose of Phase 1 is to collect the requirements for the system. These requirements are expressed in the terminology of the system environment or the application domain, respectively. Requirements capture is performed by conducting interviews and studying documents. The results of Phase 1 are validated by reviewing the minutes of the interviews together with the interview partner and by reviewing the used documents with their authors.

In Phase 2, the current state of affairs must be described, analyzed and assessed. External actors and entities must be identified; functionalities must be described and decomposed. The result of this phase are domain-specific rules, as well as descriptions of the strengths and weaknesses of the current system. As in Phase 1, the validation of the produced results is done by reviews.

The results of Phases 1 and 2 are not covered by the CC. However, they are needed to provide a firm basis for the preparation of the CC documents.

**Table 2.** Agenda for problem oriented system modeling

| Phase | Content | Format | CC Documents | Validation |
|---|---|---|---|---|
| **4.1.** Context modeling | structure of system embedded in its environment | context diagram | SPM part 1 | must be compatible with Phase 3 |
| **4.2.** Define constraints and system properties | TOE security requirements, security requirements of environment | instantiated text from CC part 2 catalogue | ST: functional requirements, summary specification | see Phase 3 |
| **4.3.** Interface definition | data formats, system behavior at external interfaces | data dictionary, MSCs | FSP, part 1 | each service contained in the mission statement must be modeled |
| **4.4.** Modeling of system environment | external components and their behavior, environmental constrains and assumptions | CEFSM | SPM, part 2 | must be compatible with Phases 3 and 4.1 |
| **4.5** Modeling of system services | service specifications | informal text and CEFSM | FSP, part 2 | see Phase 4.3 |

The goal of the requirements analysis, i.e. Phase 3, is to qualitatively describe which purpose the new system serves in its environment, and which services it must provide. Strict requirements and constraints for the new system are set up. As in Phase 2 for the existing system, external actors and entities are identified for the desired system. The requirements captured in Phase 1 can now be made more concrete. Thus, the mission statement is set up. The validation condition associated with Phase 3 requires that all requirements captured in Phase 1 be taken into account in the mission statement. In contrast to Phase 2, which makes *descriptive* statements, Phase 3 makes *prescriptive* statements.

The purpose of Phase 4 is to define the system and its environment. The system entities and their attributes are defined, as well as the processes and procedures they are involved in. Case distinctions imposed by domain rules are identified with respect to the entities and processes.

This phase consists of five sub-phases, which are also represented as an agenda, see Table 2.

In Phase 4.1, the boundary between the system and its environment is defined.

In Phase 4.2, the security requirements for the target of evaluation (TOE) and for the system environment are instantiated from the CC by defining constraints or properties.

In Phase 4.3, the interface of the system is specified in detail. MSCs are used to represent traces of system services.

In Phases 4.4 and 4.5, communicating extended finite state machines (CEFSMs) are set up for the environment as well as for each system service identified in the mission statement. For each service, functional as well as non-functional properties must be defined.

## 3 Case Study: Electronic Purse Card (EPC)

We now illustrate the method presented in Section 2 by an industrial case study[2]. EPCs are introduced to replace Eurocheque (EC) cards for the payment of goods purchased in retail stores. For reasons of space, we can present only parts of the documents produced when performing our method.

As a novelty, we define an EPC system that uses mutual asymmetric authentication. This security protocol guarantees mandatory authenticity of two communication partners. In contrast to the symmetric authentication, the asymmetric authentication procedures have the advantage that they do not need a common secret between the partners.

In case of asymmetric authentication, each communication partner has its own key pair which is generated independently from other key pairs within the terminals and the cards, respectively. A *personalization authority* initiates the key generation within the components and the signing of the public key by a *certification authority* to ensure the correctness of the key generation procedure. By using asymmetric authentication, the e-cash procedure becomes open to other terminal manufacturers and card emitters, as long as their components are personalized by the personalization authority. In the following, we sketch each of the development phases introduced in Section 2.

*Phase 1: problem oriented requirements capture.* Requirements for the EPC system include:

– Payment must be simple for all participants.
– Payment is anonymous, but at the same time authentic; non-repudiation is guaranteed.
– Stolen EPCs cannot be used.
– EPCs and terminals can be neither intercepted nor forged.

*Phase 2: analysis of current system.* In this phase, it is described how payment with EC cards proceeds. Figure 2 shows the different stages.

Examples of domain-specific rules are that a personal identification number (PIN) has four digits, and that it must be counted how often a wrong PIN has been entered.

Some weaknesses of the EC card system are that payments are not anonymous, that the access to the customer's account is protected only by the PIN,

[2] Similar systems have been evaluated by TÜViT, Essen.

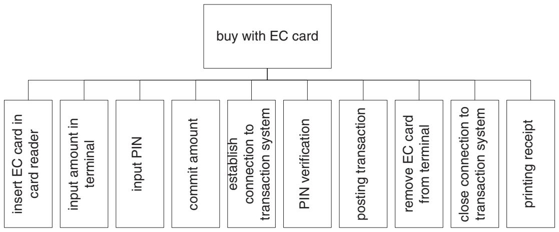
buy with EC card
insert EC card in card reader
input amount in terminal
input PIN
commit amount
establish connection to transaction system
PIN verification
posting transaction
remove EC card from terminal
close connection to transaction system
printing receipt

**Fig. 2.** Payment with Eurocheque card

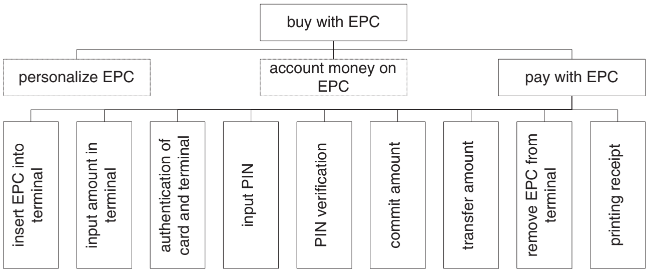

**Fig. 3.** EPC system

and that the connection between the terminal and EC card is insecure. Hence, customer profiles can be constructed, the customer can be damaged by revelation of the PIN, and the system is not protected against man-in-the-middle-attacks.

*Phase 3: problem oriented requirements analysis.* EPCs function differently from EC cards. Before the customer can pay with the EPC, the card must be loaded. For this purpose, it must be inserted into a bank terminal, and the PIN and the desired amount must be entered. Purchasing goods with the EPC proceeds similarly as with the EC card, but the amount is debited from the card instead of from the customer's account. Moreover, the bank and cash terminals and the EPC must be personalized by a personalization authority. This means that a pair of keys (a public and a private one) is generated for each component, where the public key is certified by a certification authority.

Figure 3 shows the desired services of the EPC system.

The EPC system, too, has potential weaknesses. For example, a man-in-the-middle attack or spying out the PIN may be possible. An analysis of such weaknesses and the corresponding attack scenarios lead to the following security goals:

– Debiting the customer account is done in a secure environment (bank terminal).
– An EPC is useless if its secrets are unknown.
– Neither the cards nor the terminals can be copied or forged.

- The connections between the terminals and the EPC are encrypted, so that intercepting those connections is useless.
- Transactions take place only between authenticated components.

The following assumptions concerning the environment must be made:

- The personalization of the card is secure.
- The bank terminals are installed in a protected area and cannot be intercepted.

Now, the requirements set up in Phase 1 can be made more concrete. Simplicity of payment means that a payment is performed just by debiting the EPC. The customers only need to type their PIN and to confirm the transaction. The store personnel only needs to specify the amount and to hand the printed receipt to the customer. Anonymity is guaranteed, because the only documentation of the payment is the customer receipt. Because of the authentication mechanism, authenticity and non-repudiation are guaranteed. Stolen cards cannot be used, because the PIN is known only to the card holder, and the card secret will not be revealed by an authenticated terminal. Interception is made useless by encryption, and copying cards is prevented by preventing direct access to the physical card storage. Inh this paper, we have only given an informal sketch of Phase 3. In real-life projects, this phase is performed much more thoroughly.

*Phase 4.1: context modeling.* We present two different documents that show the system and its embedding in its environment. Figure 4 shows the security policy model for the EPC system in SDL notation. It shows the EPC in its environment consisting of an intruder, a terminal, a personalization and a certification authority (CA).

The personalization and CA components are not the main concern of our discourse and are therefore drawn with dotted lines. The terminal is used for e-cash transactions. It is personalized, which means that it has a key pair, and its public key is signed by certification authority. The intruder models a man-in-the-middle attack, i.e. the intruder intercepts the communication between card and terminal and can therefore attack both the card and the terminal. The EPC is the target of evaluation (TOE). The card application includes functionality for mutual asymmetric authentication, PIN check, credit and debit transactions. It is assumed that the card is personalized. The components in the SPM are interconnected by channels, shown in the diagram by inscripted arcs. The external channels $chUser$ and $chCard$ represent the interactions between $Terminal$ and $User$ and between $Terminal$ and $CardReader$. The internal channels connect $Terminal$ and $Intruder$ or $Intruder$ and $EPC$, respectively.

If a system is to be certified according to EAL7, we need a completely formal model, which we express as a colored Petri net (CPN).

*Phase 4.2: define constraints and system properties.* As an example for the CC part 2 catalogue we take the component FCS_COP.1.1 (Cryptographic operation from class cryptographic support):

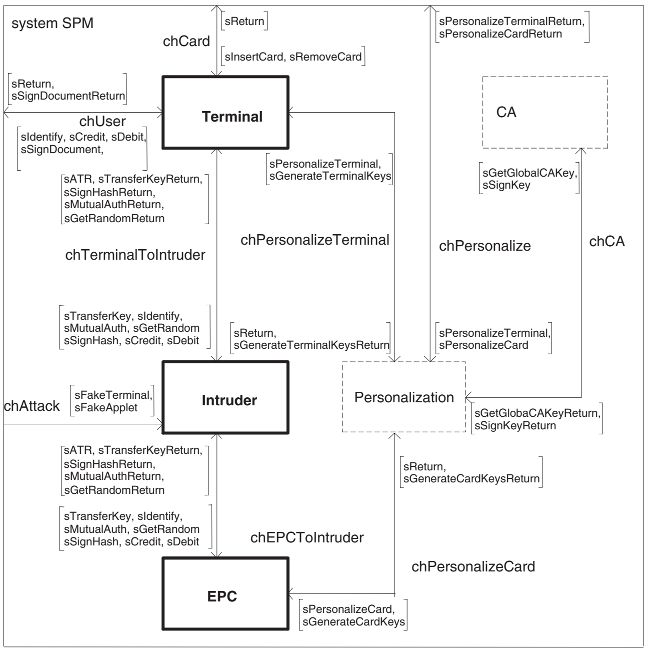
system SPM
chCard
sReturn
sInsertCard, sRemoveCard
sReturn, sSignDocumentReturn
chUser
sIdentify, sCredit, sDebit, sSignDocument,
Terminal
sPersonalizeTerminal, sGenerateTerminalKeys
sPersonalizeTerminalReturn, sPersonalizeCardReturn
CA
sGetGlobalCAKey, sSignKey
sATR, sTransferKeyReturn, sSignHashReturn, sMutualAuthReturn, sGetRandomReturn
chTerminalToIntruder
chPersonalizeTerminal
chPersonalize
chCA
sTransferKey, sIdentify, sMutualAuth, sGetRandom sSignHash, sCredit, sDebit
sReturn, sGenerateTerminalKeysReturn
sPersonalizeTerminal, sPersonalizeCard
chAttack
sFakeTerminal, sFakeApplet
Intruder
Personalization
sGetGlobaCAKeyReturn, sSignKeyReturn
sATR, sTransferKeyReturn, sSignHashReturn, sMutualAuthReturn, sGetRandomReturn
sReturn, sGenerateCardKeysReturn
sTransferKey, sIdentify, sMutualAuth, sGetRandom sSignHash, sCredit, sDebit
chEPCToIntruder
chPersonalizeCard
EPC
sPersonalizeCard, sGenerateCardKeys

**Fig. 4.** SDL security policy model

> FCS_COP.1.1 The TSF[3] shall perform [assignment: list of cryptographic operations] in accordance with a specified cryptographic algorithm [assignment: cryptographic algorithm] and cryptographic key sizes [assignment: cryptographic key sizes] that meet the following: [assignment: list of standards].

For our EPC system, this component is instantiated as follows:

> FCS_COP.1.1 The TSF shall perform the mutual authentication procedure in accordance with a specified cryptographic algorithm RSA and cryptographic key sizes of 1024 bit that meet the following: IEEE 1363.

In addition, it is necessary to follow all dependencies between the components. In this case, the FCS_COP.1.1 component requires to include the cryptographic key generation component.

*Phase 4.3: interface definition.* As an example, we consider the asymmetric authentication protocol. It is specified by means of a message sequence chart, which in fact is the common specification technique for technical protocols.

[3] TOE security function

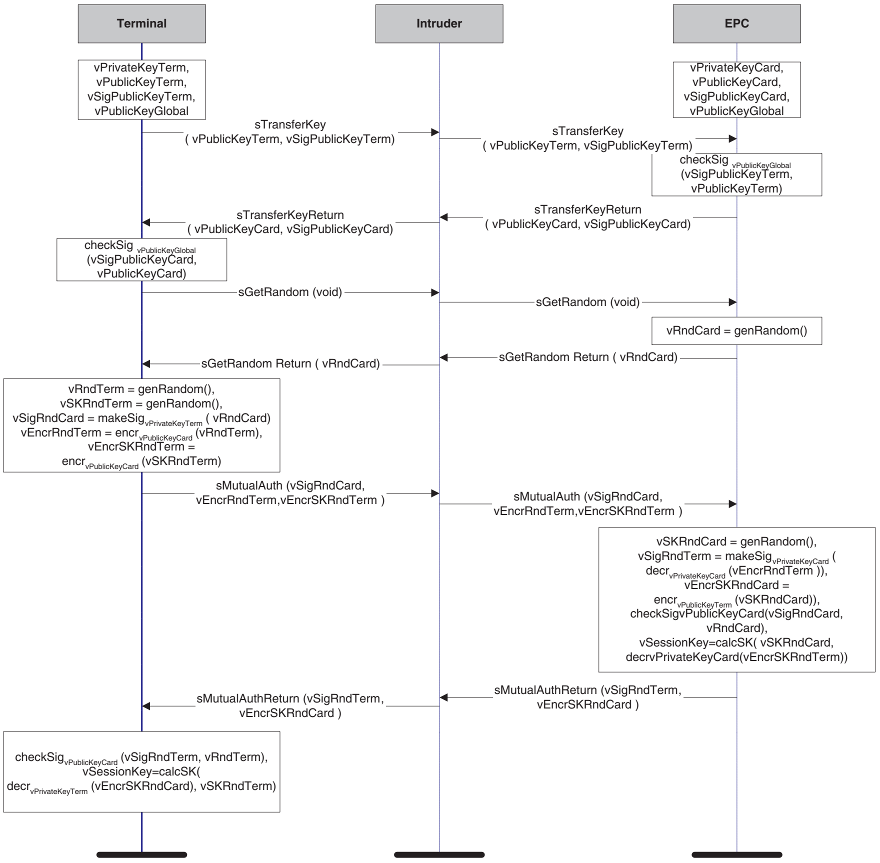
Terminal
Intruder
EPC
vPrivateKeyTerm, vPublicKeyTerm, vSigPublicKeyTerm, vPublicKeyGlobal
vPrivateKeyCard, vPublicKeyCard, vSigPublicKeyCard, vPublicKeyGlobal
sTransferKey ( vPublicKeyTerm, vSigPublicKeyTerm)
sTransferKey ( vPublicKeyTerm, vSigPublicKeyTerm)
checkSig vPublicKeyGlobal (vSigPublicKeyTerm, vPublicKeyTerm)
sTransferKeyReturn ( vPublicKeyCard, vSigPublicKeyCard)
sTransferKeyReturn ( vPublicKeyCard, vSigPublicKeyCard)
checkSig vPublicKeyGlobal (vSigPublicKeyCard, vPublicKeyCard)
sGetRandom (void)
sGetRandom (void)
vRndCard = genRandom()
sGetRandom Return ( vRndCard)
sGetRandom Return ( vRndCard)
vRndTerm = genRandom(), vSKRndTerm = genRandom(), vSigRndCard = makeSigvPrivateKeyTerm ( vRndCard) vEncrRndTerm = encrvPublicKeyCard (vRndTerm), vEncrSKRndTerm = encrvPublicKeyCard (vSKRndTerm)
sMutualAuth (vSigRndCard, vEncrRndTerm,vEncrSKRndTerm )
sMutualAuth (vSigRndCard, vEncrRndTerm,vEncrSKRndTerm )
vSKRndCard = genRandom(), vSigRndTerm = makeSigvPrivateKeyCard ( decrvPrivateKeyCard (vEncrRndTerm )), vEncrSKRndCard = encrvPublicKeyTerm (vSKRndCard)), checkSigvPublicKeyCard(vSigRndCard, vRndCard), vSessionKey=calcSK( vSKRndCard, decrvPrivateKeyCard(vEncrSKRndTerm))
sMutualAuthReturn (vSigRndTerm, vEncrSKRndCard )
sMutualAuthReturn (vSigRndTerm, vEncrSKRndCard )
checkSigvPublicKeyCard (vSigRndTerm, vRndTerm), vSessionKey=calcSK( decrvPrivateKeyTerm (vEncrSKRndCard), vSKRndTerm)

**Fig. 5.** MSC of authentication protocol

Figure 5 shows a successful asymmetric mutual authentication between a terminal and an EPC, intercepted by an intruder. The sequence of action can be divided into three phases:

– public key exchange and check of public key signature using the pulic key of the CA
– random number request by terminal
– authentication and session key generation

Each phase starts with a command called by the terminal followed by an EPC response.

*Phase 4.4: modeling of system environment.* For reasons of space, we cannot present the CEFSMs representing the environment.

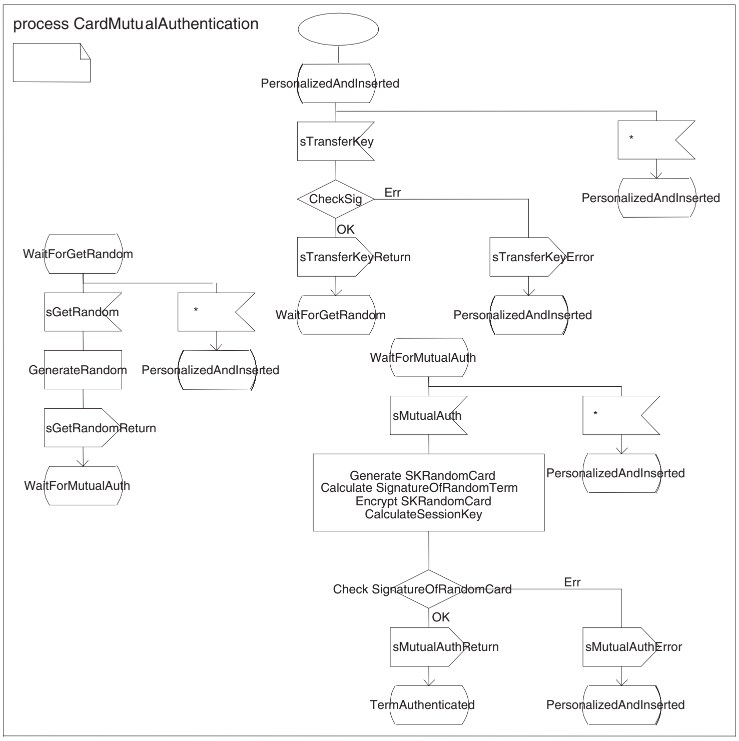

**Fig. 6.** SDL definition of authentication

*Phase 4.5: modeling of system services.* We consider the authentication service, where we present an SDL and a CPN version.

The *EPC* part of the asymmetric authentication protocol is modeled as an SDL state machine, see Figure 6. The three phases of the MSC are reflected by the three parts of the state machine. The state machine starts in the state *PersonalizedAndInserted*.

1. The phase "Public Key Exchange and Check of Public Key Signature" leads to the state *WaitForGetRandom* or remains in the state *Personalized–AndInserted*.
2. The phase "Random Number Request by Terminal" leads to the state *Wait–ForMutualAuth* or back to the state *PersonalizedAndInserted*.
3. The phase "Authentication and Session Key generation" leads to the state *Term– Authenicated* or back to the state *PersonalizedAndInserted*.

The state machine is now modeled formally in CPN, see Figure 7. It is just a translation of the SDL protocol machine into CPN. Therefore the CPN model also contains the same three phases, states and transitions. In CPN, the states of the state machine are modeled by places for simple tokens. The channels are

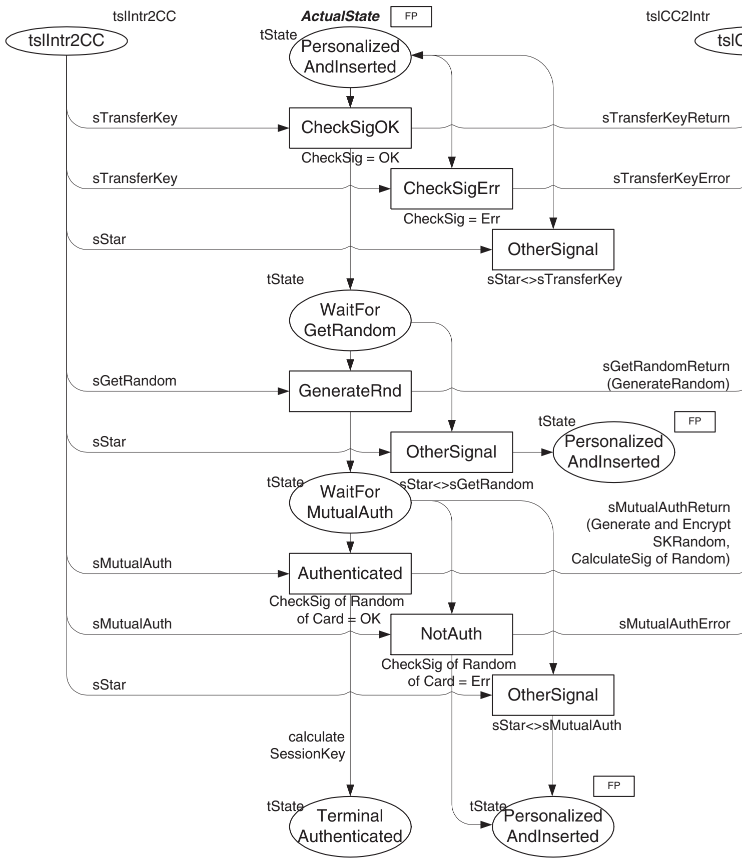

**Fig. 7.** CPN model of authentication

also modeled by places, but using more complex tokens (colors). Arc inscriptions are used to model the functionality of the transitions. Conditions are modeled with the CPN guard mechanism.

In the subsequent phases of the method, these documents will be further validated and refined. Because of the formal nature of the documents, the required security properties can be demonstrated in a routine way [8].

## 4 Conclusions

We have presented a method to model systems in such way that their certification according to the higher levels of the CC is well prepared. This method shows that problem analysis can be performed in an analytic and systematic way, even though problem analysis is often regarded as an unstructured task that needs – above all – creative techniques. In contrast, we are convinced that problem analysis requires sound engineering techniques to achieve non-trivial and high-quality results. Using our method, formal documents as they are required by the CC, can be developed in an appropriate way.

The problem orientation of the method ensures a high degree of correspondence between the system model and the reality. This is due to the facts that the modeling process is oriented on the system mission and that the requirements are analyzed using terms of the application domain.

Such a correspondence is crucial. If it is not given, inadequate models may be set up. Such inadequate models may have serious consequences: relevant properties may be impossible to prove. Instead, irrelevant properties may be proven, which would lead to an unjustified trust in the system.

Our method is systematic and thus repeatable, and gives guidance how to model security properties. The risk of omissions is reduced, because the agenda leads the attention of the system engineers to the relevant points.

Because of our method, we are now able to suggest some improvements to the CC. Until now, the CC required security models only for access control policies and information flow policies, because only these belonged to the state of the art. By modeling the system environment, we have succeeded in setting up a formal model also for authentication.

To the best of our knowledge, we are the first to propose such a systematic, problem oriented approach to CC certification.

In the future, we will work on validation and refinement in general, and on a complete validation of authentication SPM in particular.

## References

[1] Common criteria. See http://www.commoncriteria.org/. 334

[2] F. Belina and D. Hogrefe. The CCITT Specification and Description Language SDL. *Computer Networks and ISDN Systems*, 16(4):311–341, March 1989. 336

[3] B. Blanchard and W. Fabrycky. *Systems Engeneering and Analysis.* Prentice Hall, 1980. 335

[4] M. Heisel. Agendas – a concept to guide software development activites. In R. N. Horspool, editor, *Proc. Systems Implementation 2000*, pages 19–32. Chapman & Hall London, 1998. 336

[5] ITU-TS. ITU-TS Recommendation Z.120anb: Formal Semantics of Message Sequence Charts. Technical report, ITU-TS, Geneva, 1998. 336

[6] M. Jackson. *Problem Frames. Analyzing and structuring software development problems.* Addison-Wesley, 2001. 335

[7] K. Jensen. Colored Petri nets. *Lecture Notes Comp. Sci.: Advances in petri nets*, 254:248–299, 1986. 336

[8] K. Jensen. *Colored Petri nets, Vol. II.* Springer, 1995. 345

[9] G. Kolonya and I. Sommerville. *Requirements Engineering.* Wiley, 1997. 335