Modelling and Analysing the Finance Case Study in UMC*

Maurice H. ter Beek and Franco Mazzanti

ISTI–CNR, Via G. Moruzzi 1, 56124 Pisa, Italy
{terbeek,mazzanti}@isti.cnr.it

Abstract. We describe a UMC model of the credit request scenario from Sensoria's Finance case study and verify a number of properties expressed in the service-oriented logic SocL. The UMC model is a faithful translation of the scenario's specification in UML4SOA, which is a profile for UML that defines a high-level domain-specific modelling language for behavioural service specifications.

1 Introduction

Service-Oriented Computing (SOC) has emerged as a new paradigm for distributed computing, evolving from object-oriented computing by allowing autonomous, platform-independent computational entities (called services) to be built (described, discovered, composed) using object-oriented techniques. Service-Oriented Architectures (SOA) are then used to implement the SOC paradigm in areas like e-commerce and e-government. The work described in this report was performed in the context of the EU project Sensoria [6], whose aim was to develop a comprehensive and pragmatic—but theoretically well-founded—approach to software engineering for service-oriented systems. We refer to [10] and the references therein for details on Sensoria.

2 Credit Request Scenario

Our starting point is the credit request scenario from the Finance case study. It has been provided by S&N [5], which is one of Sensoria's industrial partners and a leading IT company of the financial services industry. The scenario has been specified in UML by using the profiles SoaML [7], which defines a metamodel for designing the structural aspects of SOA, and UML4SOA [2], which defines a high-level domain-specific modelling language for behavioural service specifications.

Figure 1 shows the scenario's overall configuration. Its main services are the CreditRequest and—to a lesser degree—Rating. The remaining components are web services that serve to interact with clients, save the information they provide, calculate ratings, and the like. Portal, Credit Management, Customer Management, Security Analysis, Balance Analysis, and Rating Calculator are self-contained services used by the Credit Request and Rating services.

* This work was funded by the EU project Sensoria (IST-2005-016004).

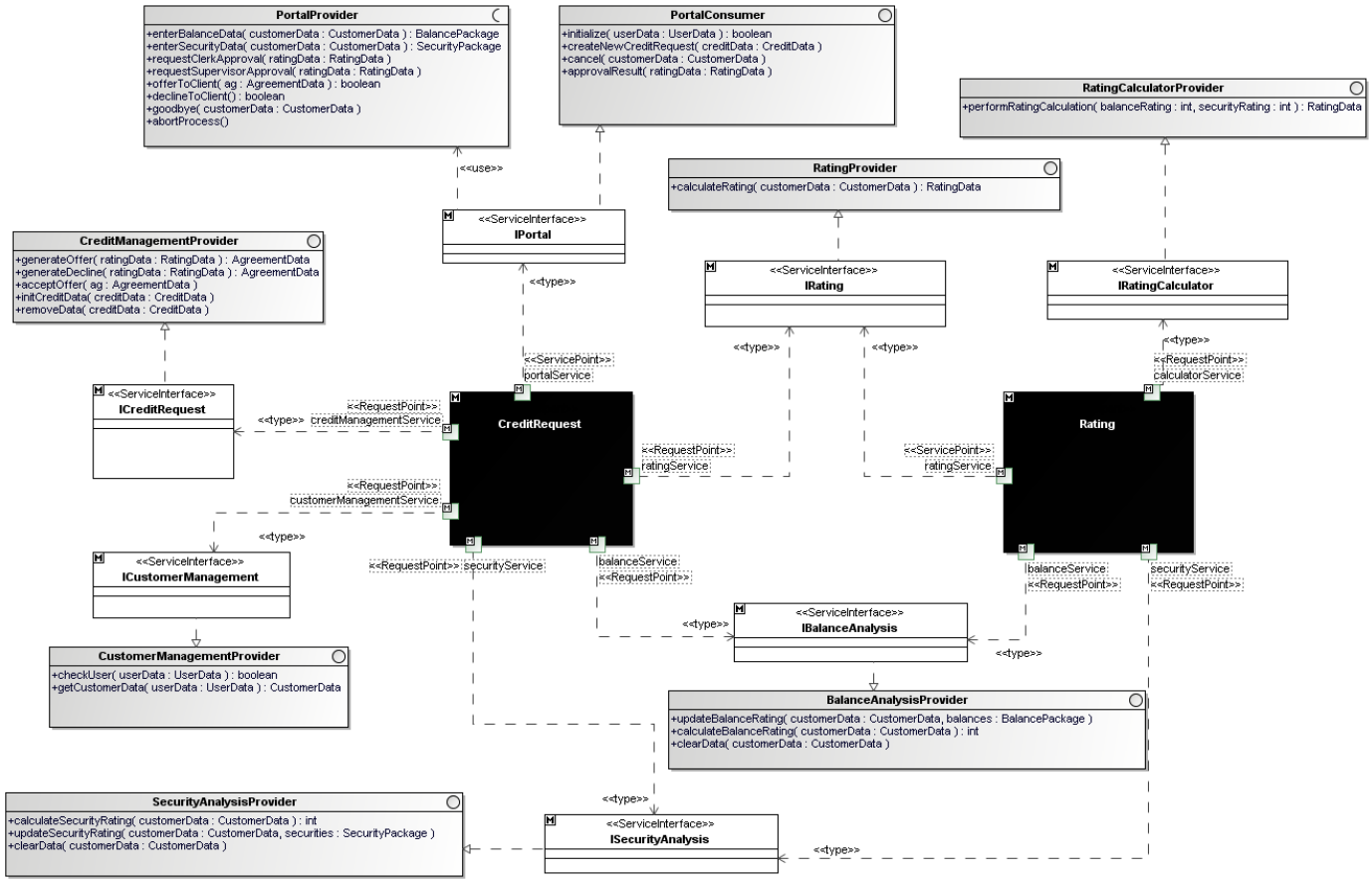


Fig. 1. Overall configuration of the Credit Request scenario.

The Credit Request service is depicted in Fig. 2. It describes a bank service that offers clients the possibility to ask for a loan and subsequently orchestrates the steps needed to process this request, which may involve an evaluation by a clerk or a supervisor before a contract proposal is sent to the client. The scopes **Initialize** and **Finalize** handle a client's login and log off. The loan workflow is represented in the scope **Main**, whose initial activity is to service the request.

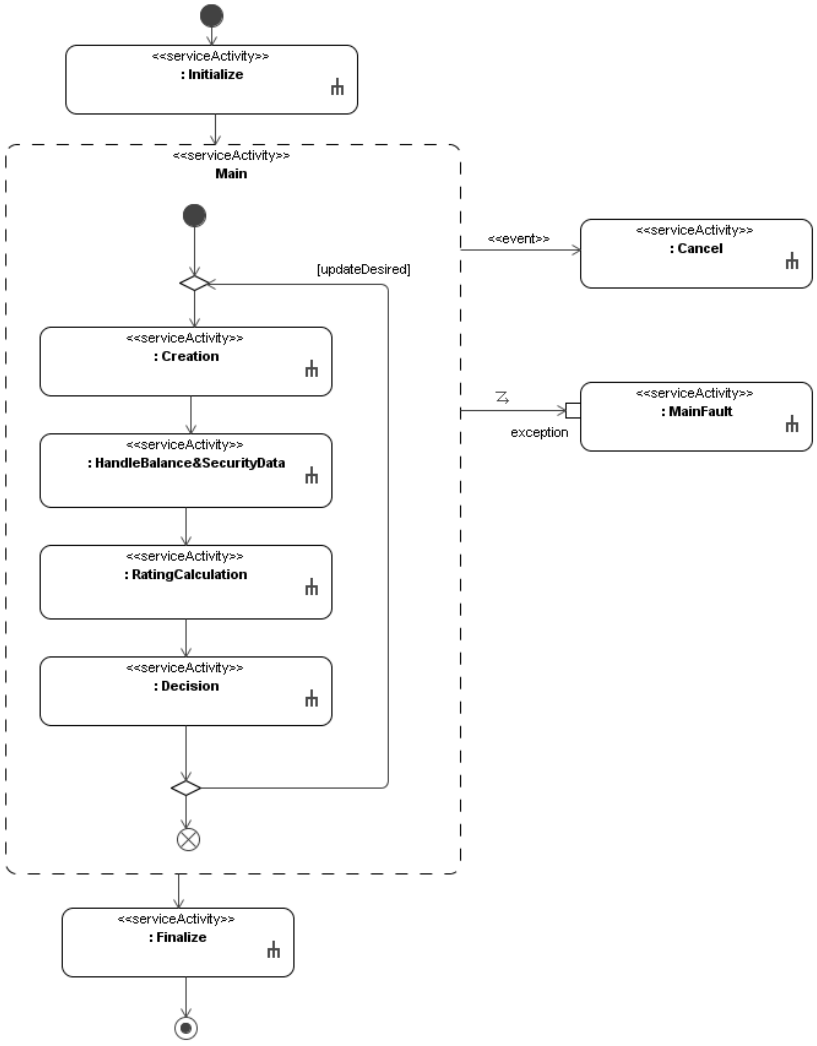


Fig. 2. Service Credit Request.

Figure 3 shows the **Creation** scope, which starts with a call from the Portal. The data involved is stored in the Credit Management service and a confirmation is sent to the Portal. In case of a fault, the compensation handler removes the request from the Credit Management service.

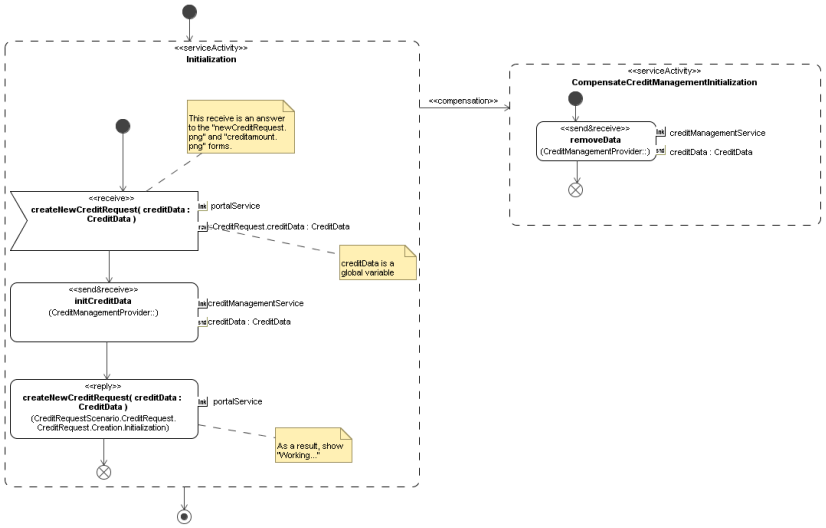


Fig. 3. Scope **Creation**.

The retrieval of the client's balances and securities is handled by the **HandleBalance&SecurityData** scope, depicted in Fig. 4, and these are stored in the Balance and Security services, respectively.

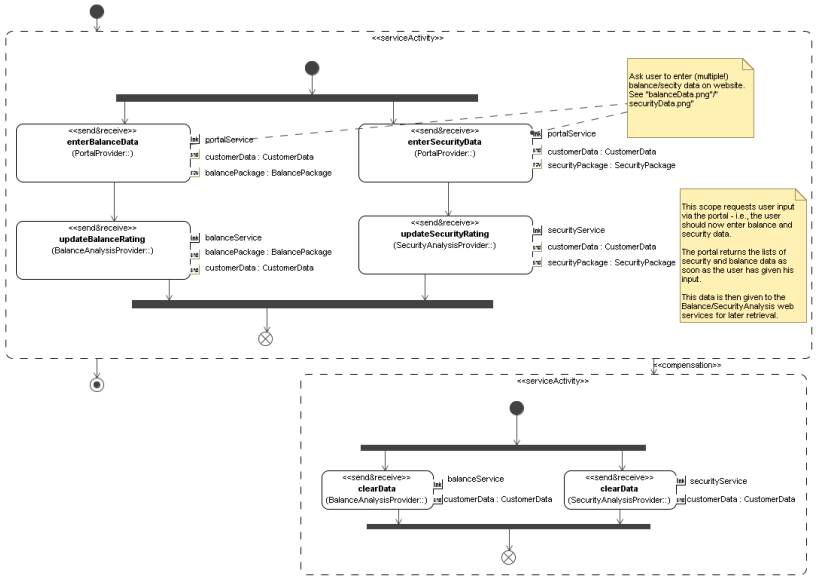


Fig. 4. Scope **HandleBalance&SecurityData**.

Upon completion of the request, the Rating service calculates the rating of the client's request after being asked to do so through the **RatingCalculation** scope depicted in Fig. 5.

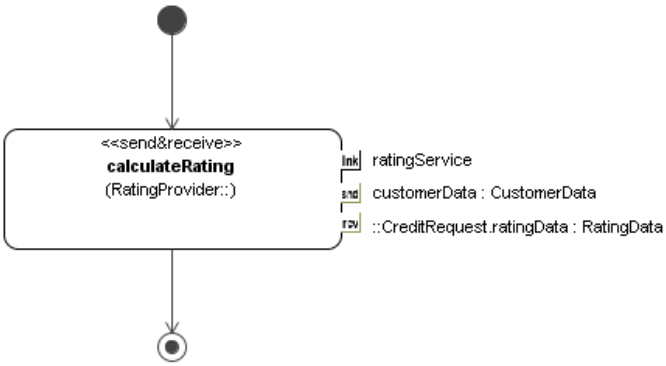


Fig. 5. Scope **RatingCalculation**.

The Rating service calculates a rating, which implies whether the request can be accepted automatically or a clerk or a supervisor needs to decide this.

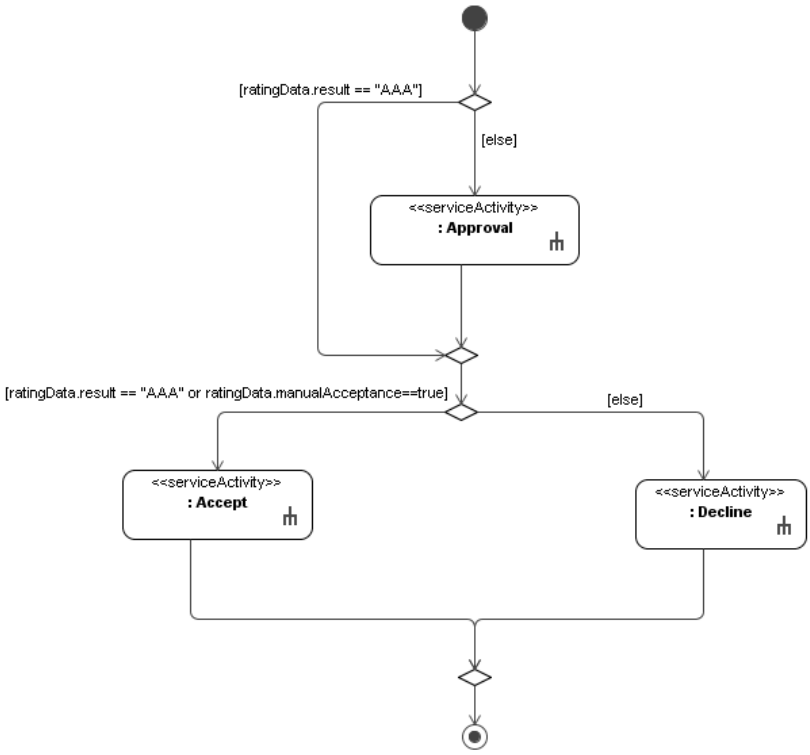


Fig. 6. Scope **Decision**.

Figure 6 depicts the **Decision** scope. Rating AAA automatically leads to an offer to the client. In any other case, the **Approval** scope, depicted in Fig. 7, is used for a decision by an employee, based on which an offer or a decline is generated, according to the **Accept** and **Decline** scopes depicted in Fig. 8.

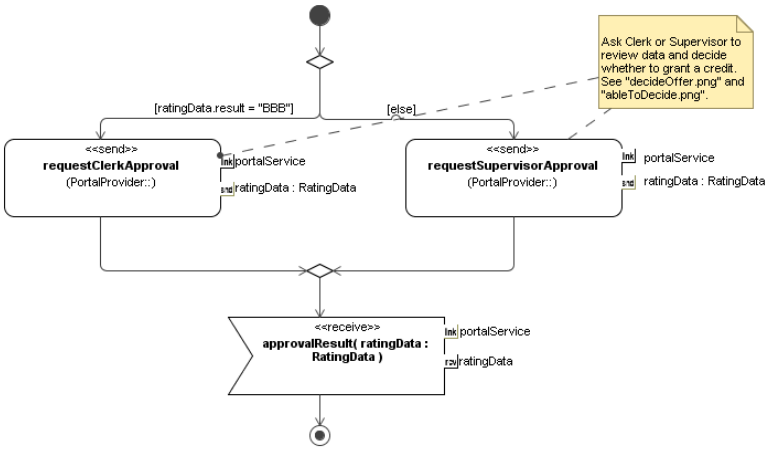
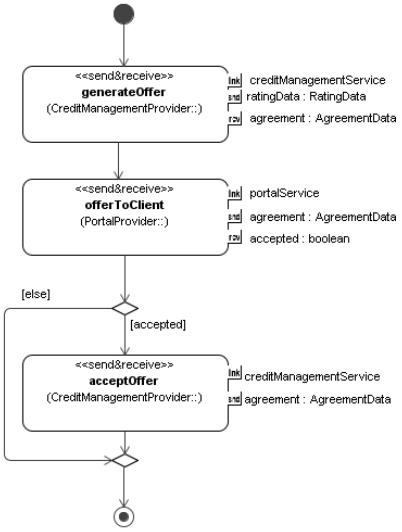
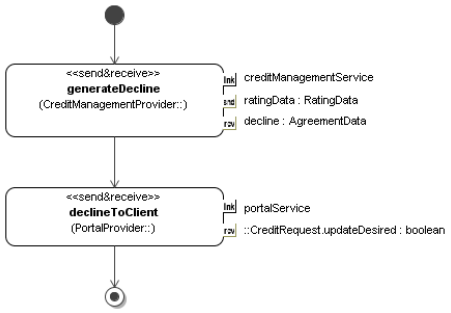


Fig. 7. Scope **Approval**.

Rating BBB means that the request has some risk, but can be decided by a clerk, while CCC indicates a much more risky request that needs a supervisor to decide. The decision is sent to the Portal. An offer is saved in the Credit Management service and sent to the Portal, allowing the client to see it and decide to accept or decline it through interaction with the Portal. In case of a decline, the client is allowed to update the data and restart the request. At any moment, the client may want to abort the request, in which case the request data needs to be deleted. This requires the execution of compensation activities to semantically rollback the action of storing the request data performed by the involved services, preventing services to keep information of aborted requests.



(a) Scope **Accept**.



(b) Scope **Decline**.

Fig. 8. (a)-(b) Scopes **Accept** and **Decline**.

3 A UMC Model

Appended to this report, we provide a UMC model of the scenario described in the previous section. This model has been obtained by translating the specification provided by S&N. In the specification discussed in the previous section only the CreditRequest and Rating services are described as UML4SOA activity diagrams, while the other components used by these services (the Portal, Credit Management, Customer Management, Security Analysis, Balance Analysis, and Rating Calculator services) are described by UML4SOA protocol state machines. Since UMC needs a closed model, we have moreover modelled a single Client (interacting with a single instance of the Portal) that can make up to two credit requests. The translation of the main CreditRequest and Rating services has been performed in a rather general way, in order to be able to serve as a blueprint for an automatic translation tool from UML4SOA specifications into UMC code. This explains the at times long-winded organizational structure. The UMC enconding of the other components, originally specified by protocol state machines, has been done by hand.

It is outside the purpose of this paper to try to present the rationale and the rules according to which the UML4SOA activity diagrams have been translated into UML statecharts. We just hint here that in general a UML4SOA activity diagram is mapped into a corresponding UML statechart with a single state. Each progression step during the execution of a UML4SOA activity diagram is modelled by one (or sometimes more) transitions in the statechart model, which implement the corresponding semantics. In general, the UML statechart transitions modelling the execution of an activity node have the following form, where tau represents an internal signal of the statechart:

```
s1 -> s1 -- actually no state change
  { tau [enabling conditions for incoming edges] /
        resetting of enabling conditions;
        execution of specific node activity;
        setting of enabling conditions for outgoing edges;
        self.tau;
  }
```

As a specific example, we consider the ≪send≫ node shown in Fig. 9. The corresponding specific UMC transition rule modelling the execution of this node is as follows:

```
s1 -> s1
  { tau [N1_Finalize_DefaultInitialOUT = true] /
        N1_Finalize_DefaultInitialOUT := false;
        LNK_portalService.goodbye([self],
          VAR_CreditRequest_customerData);
        N2_Finalize_Send_goodbyeOUT := true;
        self.tau;
  }
```

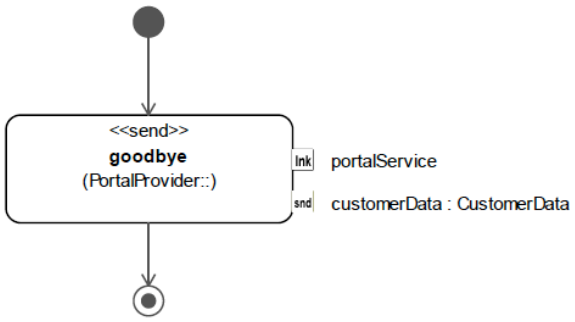


Fig. 9. A ≪send≫ node from scope **Finalize**.

Since UML4SOA is a high-level modelling language, some details of the specification are voluntarily underspecified. The UMC model is therefore sometimes forced to make some specific assumption on some details of the underlying executable model, and such assumptions can often be explicitly specified and customized by the user.

For instance, the default UMC choice adopted in our translation is that messages which arrive at a time at which they cannot be accepted are never discarded. This specific choice can be reversed by simple avoiding to declare as 'Deferred' the corresponding events.

Moreover, the order in which events are removed from an object queue is by default supposed to be random in our translation. This specific choice can be reversed (even on a *per-object* basis) by changing the object queue policy from 'RANDOM' to 'FIFO'.

Finally, in our translation all objects are by default assigned the same priority (thus modelling the maximal degree of nondeterminism in their scheduling). This choice can be reversed by assigning to each entity its own priority, thereby modelling more deterministic schemas.

4 Verification with UMC

We have analyzed the UMC model of the previous section (appended to this report) with the on-the-fly model checker UMC [1, 3] (its current prototype can be experimented via a web interface [9], which also includes a user guide [4]). UMC allows the efficient verification of SocL formulae over a set of communicating UML state machines. SocL [3] is an event- and state-based, branching-time, efficiently verifiable, parametric temporal logic that was specifically designed to capture peculiar aspects of services. Below are some examples of the properties that we have verified, inspired by the Patterns of service properties listed in [3].

– "The CreditRequest service is *available*"
The formula:

$$AF\ (accepting\_requests(initialize))$$

is TRUE (11 states generated and 22 computation fragments generated). This formula means that in every state the CreditRequest service may eventually accept (the initialization of) a credit request.

– "The CreditRequest service is *responsive*"
The formula:

$$AG\ [accept(createNewCreditRequest)]$$
$$AF\ \{response(declineToClient)\ OR\ response(offerToClient)\ OR$$
$$receive(cancel)\}\ true$$

is TRUE (39530 states generated, 79060 computation fragments generated). This formula means that whenever the CreditRequest service accepts a credit request, it always eventually provides a (successful or unsuccessful) response to the credit request or (unless) the credit request is cancelled by the client.

– "The CreditRequest service is *cancelable*"
The formula:

$$AG\ [accept(createNewCreditRequest)]$$
$$A\ [accepting\_requests(cancel)\ \{true\}\ U$$
$$\{response(declineToClient)\ OR\ response(offerToClient)\ OR$$
$$receive(cancel)\}\ true]$$

is TRUE (39530 states generated, 79060 computation fragments generated). This formula means that whenever the CreditRequest service accepts a credit request, it always is ready to accept a cancellation of the credit request by the client until it provides a (successful or unsuccessful) response to the credit request or the credit request is indeed cancelled.

– "The CreditRequest service is *compensable*"
The formula:

$$AG\ [request(calculateRating)]\ AG\ [receive(cancel)]$$
$$((AF\ \{send(clearData)\}\ AF\ \{send(clearData)\}\ true)\ AND$$
$$AF\ \{send(removeData)\}\ true)$$

is TRUE (39530 states generated, 190970 computation fragments generated). This formula means that whenever the CreditRequest service asks the Rating service to calculate the client's rating and whenever subsequently the client asks to cancel the credit request, the CreditRequest service always eventually clears both the client's balance and security data and removes the credit data (amount, type, monthly installment, etc.) of the client's credit request.

– "The CreditRequest and Rating services are *well-orchestrated*"
The formula:

$$AG\ (([response(declineToClient)]\ true)\ IMPLIES \\ NOT\ rating\_evalutation(AAA))$$

is TRUE (39530 states generated, 197650 computation fragments generated). This formula means that whenever the CreditRequest service declines the client's request for a credit, this implies that the Rating service did not evaluate the client's rating to be AAA.

5 Conclusions and Future Work

We have translated a UML4SOA design of the Credit Request scenario from SENSORIA's Finance case study into a formal operational UMC model. This has provided us with many useful insights into a possible automatization of the translation from UML4SOA designs into UML statecharts. A feasibility study has been initiated with the purpose of achieving such a mechanical translation of a UML4SOA design into an operational UMC model. This has resulted in an early prototype of a translator (based on the standard Eclipse EMF format and its transformation capabilities) to validate the approach. This work is still in progress. Furthermore, a new graphical interface for UMC, which allows one to directly draw a UMC model in terms of graphical nodes and edges, has been experimented in the context of the Eclipse environment, and a preliminary presentation of the approach has been presented in [8].

Acknowledgements

We thank our partners in SENSORIA for detailed discussions on the Finance case study and in particular Jannis Elgner (S&N), Philip Mayer (LMU) and Francesco Tiezzi (DSIUF).

References

1. M.H. ter Beek, A. Fantechi, S. Gnesi and F. Mazzanti, A state/event-based model-checking approach for the analysis of abstract system properties. In *Science of Computer Programming*, 2010.
2. H. Foster, L. Gönczy, N. Koch, P. Mayer, C. Montangero and D. Varró, UML Extensions for Service-Oriented Systems. In [10], 2010.
3. S. Gnesi and F. Mazzanti, An Abstract, on the Fly Framework for the Verification of Service Oriented Systems. In [10], 2010.
4. F. Mazzanti, Designing UML models with UMC. Technical Report 2009-TR-43, ISTI–CNR, 2009.
5. S&N: `http://www.s-und-n.de/`.
6. SENSORIA: `http://www.sensoria-ist.eu/`.

7. OMG, Service oriented architecture Modeling Language (SoaML)—Specification for the UML Profile and Metamodel for Services (UPMS), April 2009. http://www.omg.org/cgi-bin/doc?ptc/09-04-01.
8. A. Sulova, Model Driven Software Development con Eclipse, StatechartUMC. In *Proceedings of the 4th Italian workshop on Eclipse technologies (Eclipse-IT 2009), Bergamo, Italy* (A. Gargantini, Ed.), Eclipse Italian Community, 2009, 113–114. An extended version appeared as Technical Report 2009-TR-050, ISTI–CNR, 2009.
9. UMC: http://fmt.isti.cnr.it/umc/.
10. M. Wirsing and M. Hölzl (Eds.), *Rigorous Software Engineering for Service-Oriented Systems—Results of the* Sensoria *project on Software Engineering for Service-Oriented Computing*, Springer, 2010.

A Complete UMC Model of the Credit Request Scenario

We append the complete UMC model discussed in Sect. 3. It is also listed among the example models on the UMC web interface [9] as `0-0-CreditPortal.umc`.

Once loaded, this model consists of 39, 530 states. Moreover, its system classes are **CreditRequest**, **CreditManagementService**, **CustomerManagementService**, **Client**, **PortalService**, **BalanceService**, **SecurityService**, **Rating** and **CalculatorService**, while its active system objects are **TheCredit:CreditRequest**, **TheCreditManagement:CreditManagementService**, **TheCustomerManagement: CustomerManagementService**, **TheClient: Client**, **ThePortal:PortalService**, **TheBalanceService:BalanceService**, **TheSecurityService:SecurityService**, **TheRating:Rating** and **TheCalculator: CalculatorService**.

UMC allows its users to specify an *observation mode* of the system under analysis, in which one explicitly specifies a set of *hiding* and *renaming* rules that precisely define the structural information or events one is interested to observe, possibly reshaping them to fit a standard format, hiding the rest (cf. [1] for more details). This is done by adding to the model's UMC encoding an **Abstractions** section containing this list of abstraction rules. Example abstractions that are relevant for the fomulae verified in Sect. 4 are as follows:

```
Abstractions {
  State  TheCredit.N1_Initialize_DefaultInitialOUT = true ->
           accepting_requests(initialize)
  State  TheCredit.MainTOK > 0 -> accepting_requests(cancel)
  State  TheRating.VAR_Rating_Evaluation = $1 -> rating_evaluation($1)
  --
  Action TheCredit:ThePortal.createNewCreditRequest($*) ->
         accept(createNewCreditRequest)
  Action TheCredit:accept(declineToClient) -> response(declineToClient)
  Action TheCredit:accept(offerToClient) -> response(offerToClient)
  Action TheCredit:ThePortal.cancel($*) -> receive(cancel)
  Action TheCredit:accept(calculateRating) -> request(calculateRating)
  Action TheCredit:accept(clearData) -> send(clearData)
  Action TheCredit:accept(removeData) -> send(removeData)
  ...
  }
```

```
------------------------------------------------------------------------
---------------------- CREDITREQUEST SERVICE ---------------------------
------------------------------------------------------------------------
Class CreditRequest is

Signals:
  abortProcess_return(callid); -- reply from PortalService
  acceptOffer_return(callid); -- reply from CreditManagementService
  approvalResult(callid,ratingData); -- receive from PortalService
  calculateRating_return(callid,result:Token); -- reply from RatingService
  cancel(callid,customerData); -- send&receive from PortalService
  checkUser_return(callid,result:bool); -- reply from CustomerManagementS
  clearData_return(callid); -- reply from Balance/SecurityService
  createNewCreditRequest(callid,creditData); -- send&receive from Portal
  enterBalanceData_return(callid,result); -- reply from PortalService
  enterSecurityData_return(callid,result); -- reply from PortalService
  generateDecline_return(callid,decline); -- reply from CreditManagementS
  generateOffer_return(callid,agreement); -- reply from CreditManagementS
  getCustomerdata_return(callid,result); -- reply from CustomerManagementS
  initCreditData_return(callid); -- reply from CreditManagementService
  initialize(callid,userData); -- send&receivce from PortalService
  offerToClient_return(callid,accepted:bool); -- reply from PortalService
  declineToClient_return(callid,updateDesired:bool); -- reply from PortalS
  updateBalanceRating_return(callid); -- reply from BalanceService
  updateSecurityRating_return(callid); -- reply from SecurityService
  removeData_return(callid); -- reply from CreditManagementService
  --
  tau;

Vars:
  Priority: int := 10;
  RANDOMQUEUE;
  NormalPriority: int := 2;
  Startup: bool := true;
  -- STATIC LNK of Activity TOP LEVEL:
  LNK_portalService: obj;
  LNK_customerManagementService: obj;
  LNK_creditManagementService: obj;
  LNK_balanceService: obj;
  LNK_securityService: obj;
  LNK_ratingService: obj;
  ------------------------------------------------------------------------
  ---------------------- Activity CreditRequest ------------------------
  ------------------------------------------------------------------------
  -- INITIALIZATION:
  CreditRequestTOK: int := 0;
  CreditRequestIN: bool := false;
  CreditRequestOUT: bool := false;
  CreditRequestERR: bool := false;
  -- ENABLING NODES:
  N1_CreditRequest_DefaultInitialOUT: bool := false;
  -- LOCAL variables:
  VAR_CreditRequest_customerData: int[];
  VAR_CreditRequest_creditData: int[];
  VAR_CreditRequest_ratingData: int[];
  VAR_CreditRequest_updateDesired: bool := false;
  ------------------------------------------------------------------------
  ----------------------- Activity Initialize ------------------------
  ------------------------------------------------------------------------
  -- INITIALIZATION:
  InitializeTOK: int := 0;
  InitializeIN: bool := false;
  InitializeOUT: bool := false;
  InitializeERR: bool := false;
  -- ENABLING NODES:
  N1_Initialize_DefaultInitialOUT: bool := false;
  N2_Initialize_Receive_initializeOUT: bool:= false;
  N3_Initialize_SendReceive_checkUserOUT: bool:= false;
  N3_Initialize_SendReceive_checkUserSTEP: bool:= false;
  N4_Initialize_Reply_initializeOUT: bool:= false;
  N5_Initialize_SendReceive_getCustomerDataSTEP: bool:= false;
  N5_Initialize_SendReceive_getCustomerDataOUT: bool:= false;
  N6_Initialize_raise_abortOUT: bool:= false;
  -- LOCAL variables:
  VAR_Initialize_userData;
  VAR_Initialize_userOK : bool;
  -- Implicit CALLID:
  N2_Initialize_callid;
  N3_Initialize_callid;
  N5_Initialize_callid;
  ------------------------------------------------------------------------
  -------------------------- Activity Main ---------------------------
  ------------------------------------------------------------------------
  -- INITIALIZATION:
  MainTOK: int := 0;
  MainIN: bool := false;
  MainOUT: bool := false;
  MainERR: bool := false;
  -- ENABLING NODES:
  N1_Main_DefaultInitialOUT: bool := false;
  N10_Main_DefaultInitialOUT: bool := false;
  ------------------------------------------------------------------------
  ------------------------ Activity Finalize -------------------------
  ------------------------------------------------------------------------
  -- INITIALIZATION:
  FinalizeTOK: int := 0;
  FinalizeIN: bool := false;
  FinalizeOUT: bool := false;
  FinalizeERR: bool := false;
```

```
--  ENABLING NODES:
N1_Finalize_DefaultInitialOUT: bool := false;
N2_Finalize_Send_goodbyeOUT: bool:= false;
-------------------------------------------------------------------------
------------------------- Activity Cancel ------------------------------
-------------------------------------------------------------------------
-- INITIALIZATION:
CancelTOK: int := 0;
CancelIN: bool := false;
CancelOUT: bool := false;
CancelERR: bool := false;
-- ENABLING NODES:
N2_Cancel_Receive_cancelOUT: bool := false;
N3_Cancel_Raise_processCanceledOUT: bool := false;
-- LOCAL variables:
VAR_Cancel_customerData;
-- Implicit CALLID:
N2_Cancel_callid;
-------------------------------------------------------------------------
------------------------ Activity MainFault -----------------------------
-------------------------------------------------------------------------
-- INITIALIZATION:
MainFaultTOK: int := 0;
MainFaultIN: bool := false;
MainFaultOUT: bool := false;
MainFaultERR: bool := false;
-- ENABLING NODES:
N1_MainFault_DefaultInitialOUT: bool := false;
N2_MainFault_CompensateAll_cleanUpDataOUT : bool := false;
N2_MainFault_CompensateAll_InProgress: bool := false;
N2_MainFault_CompensateAll_cleanUpDataERR : bool := false;
N3_MainFault_Send_abortProcessOUT: bool := false;
-- LOCAL variables:
VAR_MainFault_COMPENSATEALL_DONE: bool := false;
VAR_MainFault_COMPENSATEALL_FAIL: bool := false;
-- Implicit CALLID:
N2_MainFault_callid;
-------------------------------------------------------------------------
----------------------- Activity Creationerr ----------------------------
-------------------------------------------------------------------------
-- INITIALIZATION:
CreationTOK: int := 0;
CreationIN: bool:= false;
CreationOUT: bool:= false;
CreationERR: bool := false;
-- ENABLING NODES:
N1_Creation_DefaultInitialOUT: bool := false;
-- LOCAL variables:
-- return point of compensation:
VAR_CompensateCreditManagementInitialization_RETURNTO;
```

```
-------------------------------------------------------------------------
---------------------- Activity Initialization --------------------------
-------------------------------------------------------------------------
-- INITIALIZATION:
InitializationIN: bool := false;
InitializationOUT: bool := false;
InitializationERR: bool := false;
InitializationTOK: int := 0;
-- ENABLING NODES:
N1_Initialization_DefaultInitialOUT: bool := false;
N2_Initialization_Receive_createNewCreditRequestOUT : bool := false;
N3_Initialization_SendReceive_initCreditDataOUT: bool := false;
N3_Initialization_SendReceive_initCreditDataSTEP: bool := false;
N4_Initialization_Reply_createNewCreditRequestOUT: bool := false;
-- Implicit CALLID:
N2_Initialization_callid;
N3_Initialization_callid;
-------------------------------------------------------------------------
--------- Activity CompensateCreditManagementInitialization ----------
-------------------------------------------------------------------------
-- INITIALIZATION:
CompensateCreditManagementInitializationIN: bool := false;
CompensateCreditManagementInitializationOUT: bool := false;
CompensateCreditManagementInitializationERR: bool := false;
CompensateCreditManagementInitializationTOK: int := 0;
-- ENABLING NODES:
N1_CompensateCreditManagementInitialization_DefaultInitialOUT:
  bool := false;
N2_CompensateCreditManagementInitialization_SendReceive_removeDataOUT:
  bool := false;
N2_CompensateCreditManagementInitialization_SendReceive_removeDataSTEP:
  bool := false;
-- Implicit CALLID:
N2_CompensateCreditManagementInitialization_callid;
-------------------------------------------------------------------------
----------------- Activity HandleBalanceSecurityData -----------------
-------------------------------------------------------------------------
-- INITIALIZATION:
HandleBalanceSecurityDataTOK: int := 0;
HandleBalanceSecurityDataIN: bool := false;
HandleBalanceSecurityDataOUT: bool := false;
HandleBalanceSecurityDataERR: bool := false;
-- ENABLING NODES:
N1_HandleBalanceSecurityData_DefaultInitialOUT: bool := false;
-- LOCAL variables:
-- return point of compensation:
VAR_withoutname2_RETURNTO;
```

```
------------------------------------------------------------------------
----------------------- Activity withoutname1 ----------------------
------------------------------------------------------------------------
-- INITIALIZATION:
withoutname1IN: bool := false;
withoutname1OUT: bool := false;
withoutname1ERR: bool := false;
withoutname1TOK: int := 0;
-- ENABLING NODES:
N1_withoutname1_DefaultInitialOUT: bool := false;
N2_withoutname1_fork1OUT: bool := false;
N2_withoutname1_fork2OUT: bool := false;
N3_withoutname1_SendReceive_enterBalanceDataOUT: bool := false;
N3_withoutname1_SendReceive_enterBalanceDataSTEP: bool := false;
N4_withoutname1_SendReceive_enterSecurityDataOUT: bool := false;
N4_withoutname1_SendReceive_enterSecurityDataSTEP: bool := false;
N5_withoutname1_SendReceive_updateBalanceRatingOUT: bool := false;
N5_withoutname1_SendReceive_updateBalanceRatingSTEP: bool := false;
N6_withoutname1_SendReceive_updateSecurityRatingOUT: bool := false;
N6_withoutname1_SendReceive_updateSecurityRatingSTEP: bool := false;
-- LOCAL variables:
VAR_withoutname1_balancePackage;
VAR_withoutname1_securityPackage;
-- Implicit CALLID:
N3_withoutname1_callid;
N4_withoutname1_callid;
N5_withoutname1_callid;
N6_withoutname1_callid;
------------------------------------------------------------------------
----------------------- Activity withoutname2 ----------------------
------------------------------------------------------------------------
-- INITIALIZATION:
withoutname2IN: bool := false;
withoutname2OUT: bool := false;
withoutname2ERR: bool := false;
withoutname2TOK: int := 0;
-- ENABLING NODES:
N1_withoutname2_DefaultInitialOUT : bool := false;
N2_withoutname2_fork1OUT : bool := false;
N2_withoutname2_fork2OUT : bool := false;
N3_withoutname2_SendReceive_clearDataOUT: bool := false;
N3_withoutname2_SendReceive_clearDataSTEP: bool := false;
N4_withoutname2_SendReceive_clearDataOUT: bool := false;
N4_withoutname2_SendReceive_clearDataSTEP: bool := false;
-- Implicit CALLID:
N3_withoutname2_callid;
N4_withoutname2_callid;
------------------------------------------------------------------------
-------------------- Activity RatingCalculation --------------------
------------------------------------------------------------------------
-- INITIALIZATION:
RatingCalculationTOK: int := 0;
RatingCalculationIN: bool := false;
RatingCalculationOUT: bool := false;
RatingCalculationERR: bool := false;
-- ENABLING NODES:
N1_RatingCalculation_DefaultInitialOUT: bool := false;
N2_RatingCalculation_SendReceive_calculateRatingOUT: bool := false;
N2_RatingCalculation_SendReceive_calculateRatingSTEP: bool := false;
-- Implicit CALLID:
N2_RatingCalculation_callid;
------------------------------------------------------------------------
------------------------- Activity Decision -------------------------
------------------------------------------------------------------------
-- INITIALIZATION:
DecisionTOK: int := 0;
DecisionIN: bool := false;
DecisionOUT: bool := false;
DecisionERR: bool := false;
-- ENABLING NODES:
N1_Decision_DefaultInitialOUT: bool := false;
------------------------------------------------------------------------
------------------------- Activity Approval -------------------------
------------------------------------------------------------------------
-- INITIALIZATION:
ApprovalIN: bool := false;
ApprovalOUT: bool := false;
ApprovalERR: bool := false;
ApprovalTOK: int := 0;
-- ENABLING NODES:
N1_Approval_DefaultInitialOUT: bool := false;
N2_Approval_Send_requestClerkApprovalOUT: bool := false;
N3_Approval_Send_requestSupervisorApprovalOUT: bool := false;
N4_Approval_Receive_approvalResultOUT: bool := false;
-- Implicit CALLID:
N4_Approval_callid;
------------------------------------------------------------------------
------------------------- Activity Accept -------------------------
------------------------------------------------------------------------
-- INITIALIZATION:
AcceptIN: bool := false;
AcceptOUT: bool := false;
AcceptERR: bool := false;
AcceptTOK: int := 0;
-- ENABLING NODES:
N1_Accept_DefaultInitialOUT: bool := false;
N2_Accept_SendReceive_generateOfferOUT: bool := false;
N2_Accept_SendReceive_generateOfferSTEP: bool := false;
N3_Accept_SendReceive_offerToClientOUT: bool := false;
N3_Accept_SendReceive_offerToClientSTEP: bool := false;
```

```
N4_Accept_SendReceive_offerOfferOUT: bool := false;
N4_Accept_SendReceive_offerOfferSTEP: bool := false;
-- LOCAL variables:
VAR_Accept_agreement: int[];
VAR_Accept_accepted: bool;
-- Implicit CALLID:
N2_Accept_callid: int[];
N3_Accept_callid: int[];
N4_Accept_callid: int[];
------------------------------------------------------------------------
------------------------ Activity Decline ------------------------------
------------------------------------------------------------------------
-- INITIALIZATION:
DeclineIN: bool := false;
DeclineOUT: bool := false;
DeclineERR: bool := false;
DeclineTOK: int := 0;
-- ENABLING NODES:
N1_Decline_DefaultInitialOUT: bool := false;
N2_Decline_SendReceive_generateDeclineOUT: bool := false;
N2_Decline_SendReceive_generateDeclineSTEP: bool := false;
N3_Decline_SendReceive_declineToClientOUT: bool := false;
N3_Decline_SendReceive_declineToClientSTEP: bool := false;
-- LOCAL variables:
VAR_Decline_decline: int[];
-- Implicit CALLID:
N2_Decline_callid: int[];
N3_Decline_callid: int[];
------------------------------------------------------------------------
--------------------- COMPENSATIONS MANAGEMENT -------------------------
------------------------------------------------------------------------
-- For every activity, a variable is created containing as value the
-- product of all its immediate subactivities (including exception /
-- compensation handlers). If an activity is a compensation handler,
-- without any other subactivities of this type, then it is assigned
-- a unique prime number > 1. If an activity is *not* a compensation
-- handler, and has no subactivities nor handlers, it is assigned 1.
VAR_CreditRequest: int := 6;
VAR_Initialize: int := 1;
VAR_Main: int := 6;
  VAR_Creation: int := 2;
    VAR_Initialization: int := 1;
    VAR_CompensateCreditManagementInitialization: int := 2;
  VAR_HandleBalanceSecurity: int := 3;
    VAR_withoutname1: int := 1;
    VAR_withoutname2: int := 3;
  VAR_RatingCalculation: int := 1;
  VAR_Decision: int := 1;
    VAR_Approval: int := 1;
    VAR_Accept: int := 1;
      VAR_Decline: int := 1;
  VAR_Finalize: int := 1;
  VAR_Cancel: int :=1;
  VAR_MainFault: int :=1;
  -- the global ordered vector of installed compensation handlers:
  ALLINSTALLED: int[];
  -- the vector of requested compensations:
  COMPENSATIONSREQUESTED: int[];
  -- the vector of return points (who requests the compensations):
  RETURNTO: int[];
  -- the vector of completed compensations:
  COMPENSATIONSDONE: int[];
  -- the vector of modes of completed compensations:
  COMPENSATIONSMODE: bool[];
  ------------------------------------------------------------------------
  -------------------------- STATE STRUCTURE -----------------------------
  ------------------------------------------------------------------------
State Top = s1

Defers
  cancel(callid,customerData),
  createNewCreditRequest(callid,creditData),
  initialize(callid,userData),
  tau

Transitions:
  ------------------------------------------------------------------------
  -------------------------- SYSTEM STARTUP ------------------------------
  ------------------------------------------------------------------------
  s1 -> s1
    { - [Startup = true] /
        self.tau;
        Startup := false;
        CreditRequestIN := true;
    }
  ------------------------------------------------------------------------
  --------------------- ACTIVITY CREDITREQUEST -------------------------
  ------------------------------------------------------------------------
  s1 -> s1
    { - [CreditRequestIN = true] /
        CreditRequestIN := false;
        CreditRequestTOK :=1 ;
        N1_CreditRequest_DefaultInitialOUT := true;
        Priority := NormalPriority;
    }
  -- DEFAULTINITIAL --> <<SERVICE ACTIVITY>> INITIALIZE:
  s1 -> s1
    { tau [N1_CreditRequest_DefaultInitialOUT = true] /
          N1_CreditRequest_DefaultInitialOUT := false;
          InitializeIN := true;
```

```
        Priority := NormalPriority + 10;
        self.tau;
  }
-- <<SERVICE ACTIVITY>> INITIALIZE --> <<SERVICE ACTIVITY>> MAIN:
s1 -> s1
  { tau [InitializeOUT = true]  /
        InitializeOUT := false;
        MainIN := true;
        Priority := NormalPriority + 10;
        self.tau;
  }
-- <<SERVICE ACTIVITY>> MAIN --> <<SERVICE ACTIVITY>> FINALIZE:
s1 -> s1
  { tau [MainOUT = true] /
        MainOUT := false;
        FinalizeIN := true;
        Priority := NormalPriority + 10;
        self.tau;
  }
-- <<SERVICE ACTIVITY>> FINALIZE --> ACTIVITYFINAL:
s1 -> s1
  { tau [FinalizeOUT = true] /
        FinalizeOUT := false;
        VAR_CreditRequest_customerData := null;
        CreditRequestTOK := 0;
        CreditRequestOUT := true;
        self.tau;
  }
-- ERRORS --> CREDITREQUEST_ERROR:
s1 -> s1
  { - [(FinalizeERR = true) or
       (MainERR = true) or
       (InitializeERR = true)] /
      FinalizeERR := false;
      MainERR := false;
      InitializeERR := false;
      VAR_CreditRequest_customerData := null;
      if CreditRequestTOK > 1 then {
        CreditRequestTOK := 0;
      };
      CreditRequestERR := true;
      CreditRequestTOK := 0;
  }
-- NOTE: the nodes "Cancel" and "MainFault" are considered part of
-- the activity of "Main" and thus not at top level.
------------------------------------------------------------------------
---------------------- ACTIVITY INITIALIZE --------------------------
------------------------------------------------------------------------
s1 -> s1
{ tau [InitializeIN = true] /
      InitializeIN := false;
      InitializeTOK :=1 ;
      N1_Initialize_DefaultInitialOUT := true;
      Priority := NormalPriority;
      self.tau;
}
-- DEFAULTINITIAL --> <<RECEIVE>> INITIALIZE:
s1 -> s1
  { initialize(callid,userData)
      [N1_Initialize_DefaultInitialOUT = true] /
      N1_Initialize_DefaultInitialOUT := false;
      LNK_portalService := callid[0];
      VAR_Initialize_userData := userData;
      N2_Initialize_callid := callid;
      N2_Initialize_Receive_initializeOUT := true;
  }
-- <<RECEIVE>> INITIALIZE --> <<SEND&...>> CHECKUSER:
s1 -> s1
  { tau [N2_Initialize_Receive_initializeOUT = true] /
        N2_Initialize_Receive_initializeOUT := false;
        N3_Initialize_callid := [self,7];
        LNK_customerManagementService.checkUser(N3_Initialize_callid,
          VAR_Initialize_userData);
        self.tau;
        N3_Initialize_SendReceive_checkUserSTEP := true;
  }
-- <<SEND&...>> CHECKUSER --> <<...&RECEIVE>> CHECKUSER:
s1 -> s1
  { checkUser_return(callid,result: bool)
      [(callid = N3_Initialize_callid) and
       (N3_Initialize_SendReceive_checkUserSTEP = true)] /
      N3_Initialize_SendReceive_checkUserSTEP := false;
      VAR_Initialize_userOK := result;
      N3_Initialize_SendReceive_checkUserOUT := true;
  }
-- <<SEND&RECEIVE>> CHECKUSER --> <<REPLY>> INITIALIZE:
s1 ->s1
  { tau [N3_Initialize_SendReceive_checkUserOUT = true] /
        N3_Initialize_SendReceive_checkUserOUT := false;
        LNK_portalService.initialize_return(N2_Initialize_callid,
          VAR_Initialize_userOK);
        N4_Initialize_Reply_initializeOUT := true;
        self.tau;
  }
-- <<REPLY>> INITIALIZE --> CHOICE [userOK] -->
--   <<SEND&...>> GETCUSTOMERDATA:
s1 -> s1
  { tau [(N4_Initialize_Reply_initializeOUT = true) and
         (VAR_Initialize_userOK = true)] /
        N4_Initialize_Reply_initializeOUT := false;
```

```
        N5_Initialize_callid := [self,5];
        LNK_customerManagementService.getCustomerdata
          (N5_Initialize_callid,VAR_Initialize_userData);
        N5_Initialize_SendReceive_getCustomerDataSTEP := true;
        self.tau;
  }
-- <<SEND&...>> GETCUSTOMERDATA --> <<...&RECEIVE>> GETCUSTOMERDATA:
s1 -> s1
  { getCustomerdata_return(callid,result)
      [(N5_Initialize_SendReceive_getCustomerDataSTEP = true) and
       (callid = N5_Initialize_callid)] /
      N5_Initialize_SendReceive_getCustomerDataSTEP := false;
      VAR_CreditRequest_customerData := result;
      N5_Initialize_SendReceive_getCustomerDataOUT := true;
  }
-- <<REPLY>> INITIALIZE --> CHOICE [not userOK] -->
--   <<RAISEEXCEPTION>> ABORT:
s1 -> s1
  { tau [(N4_Initialize_Reply_initializeOUT = true) and
         (not (VAR_Initialize_userOK = true))] /
        N4_Initialize_Reply_initializeOUT := false;
        VAR_Initialize_userData := null;
        VAR_Initialize_userOK := false;
        N2_Initialize_callid := null;
        N3_Initialize_callid := null;
        N5_Initialize_callid := null;
        InitializeTOK := 0;
        InitializeERR := true;
        self.tau;
  }
-- MERGE --> ACTIVITYFINAL:
s1 -> s1
  { tau [(N5_Initialize_SendReceive_getCustomerDataOUT = true) or
         (N6_Initialize_raise_abortOUT = true)] / -- note: NEVER true!
        N5_Initialize_SendReceive_getCustomerDataOUT := False;
        N6_Initialize_raise_abortOUT := False;
        VAR_Initialize_userData := null;
        VAR_Initialize_userOK := false;
        N2_Initialize_callid := null;
        N3_Initialize_callid := null;
        N5_Initialize_callid := null;
        InitializeTOK := 0;
        InitializeOUT := true;
        self.tau;
  }
-------------------------------------------------------------------
-------------------------- ACTIVITY MAIN --------------------------
-------------------------------------------------------------------
s1 -> s1
  { - [MainIN = true] /
      MainIN := false;
      MainTOK := 1;
      N1_Main_DefaultInitialOUT := true;
      Priority := NormalPriority;
  }
-- MERGE1 (DEFAULTINITIAL --> <<ACTIVITY>> CREATION &
--   CHOICE [updateDesired] --> <<ACTIVITY>> CREATION):
s1 -> s1
  { tau [(N1_Main_DefaultInitialOUT = true)] /
        N1_Main_DefaultInitialOUT := false;
        CreationIN := true;
        Priority := NormalPriority + 10;
        self.tau;
  }
-- MERGE2 (DEFAULTINITIAL --> <<ACTIVITY>> CREATION &
--   CHOICE [updateDesired] --> <<ACTIVITY>> CREATION):
s1 -> s1
  { tau [(VAR_CreditRequest_updateDesired = true) and
         (DecisionOUT = true)] /
        VAR_CreditRequest_updateDesired := false;
        DecisionOUT := false;
        CreationIN := true;
        Priority := NormalPriority + 10;
        self.tau;
  }
-- Note: We assume updateDesired = False by default
-- <<ACTIVITY>> CREATION --> <<ACTIVITY>> HANDLEBALANCESECURITYDATA:
s1 -> s1
  { tau [CreationOUT = true] /
        CreationOUT := false;
        HandleBalanceSecurityDataIN := true;
        Priority := NormalPriority + 10;
        self.tau;
  }
-- <<ACTIVITY>> HANDLEBALANCESECURITYDATA -->
--   <<ACTIVITY>> RATINGCALCULATION:
s1 -> s1
  { tau [HandleBalanceSecurityDataOUT = true] /
        HandleBalanceSecurityDataOUT := false;
        RatingCalculationIN := true;
        Priority := NormalPriority + 10;
        self.tau;
  }
-- <<ACTIVITY>> RATINGCALCULATION --> <<ACTIVITY>> DECISION:
s1 -> s1
  { tau [RatingCalculationOUT = true] /
        RatingCalculationOUT := false;
        DecisionIN := true;
        Priority := NormalPriority + 10;
        self.tau;
```

```
  }
-- [not updateDesired] FLOWFINAL Main:
s1 -> s1
  { tau [(DecisionOUT = true) and
         not (VAR_CreditRequest_updateDesired = true)] /
        DecisionOUT := false;
        MainTOK := MainTOK - 1;
        if  (MainTOK=0) {
          MainOUT := true;
        };
        self.tau;
  }
-- ANY_ERROR --> MAINFAULT:
s1 ->  s1
  { - [(CreationERR = true) or
       (HandleBalanceSecurityDataERR = true) or
       (RatingCalculationERR = true) or
       (DecisionERR = true)] /
      CreationERR := false;
      HandleBalanceSecurityDataERR := false;
      RatingCalculationERR := false;
      DecisionERR := false;
      MainTOK := 0;
      CancelERR := false;
      CancelIN := false;
      CancelOUT := false;
      CancelTOK := 0;
      N2_Cancel_Receive_cancelOUT := false;
      N3_Cancel_Raise_processCanceledOUT := false;
      N2_Cancel_callid := null;
      MainFaultIN := true;
      Priority := NormalPriority + 10;
  }
-- CANCEL_ERROR --> MAINFAULT:
s1 ->  s1
  { - [ CancelERR = true ] /
      CancelERR := false;
      MainTOK := 0;
      N1_Main_DefaultInitialOUT := false;
      if CreationTOK > 0 {
        if InitializationTOK > 0 {
          N1_Initialization_DefaultInitialOUT := false;
          N2_Initialization_Receive_createNewCreditRequestOUT := false;
          N3_Initialization_SendReceive_initCreditDataOUT := false;
          N3_Initialization_SendReceive_initCreditDataSTEP := false;
          N4_Initialization_Reply_createNewCreditRequestOUT := false;
        };
        InitializationIN := false;
        InitializationOUT := false;
        InitializationERR := false;
        InitializationTOK := 0;
        if CompensateCreditManagementInitializationTOK > 0 {
          N1_CompensateCreditManagementInitialization_DefaultInitialOUT
            := false;
          N2_CompensateCreditManagementInitialization_SendReceive_
            removeDataOUT := false;
          N2_CompensateCreditManagementInitialization_SendReceive_
            removeDataSTEP := false;
        };
        CompensateCreditManagementInitializationIN := false;
        CompensateCreditManagementInitializationOUT := false;
        CompensateCreditManagementInitializationERR := false;
        CompensateCreditManagementInitializationTOK := 0;
      };
      CreationTOK := 0;
      CreationIN := false;
      CreationERR := false;
      CreationOUT := false;
      if HandleBalanceSecurityDataTOK > 0 {
        N1_HandleBalanceSecurityData_DefaultInitialOUT := false;
        if withoutname1TOK > 0 {
          N1_withoutname1_DefaultInitialOUT := false;
          N2_withoutname1_fork1OUT := false;
          N2_withoutname1_fork2OUT := false;
          N3_withoutname1_SendReceive_enterBalanceDataOUT := false;
          N3_withoutname1_SendReceive_enterBalanceDataSTEP := false;
          N4_withoutname1_SendReceive_enterSecurityDataOUT := false;
          N4_withoutname1_SendReceive_enterSecurityDataSTEP := false;
          N5_withoutname1_SendReceive_updateBalanceRatingOUT := false;
          N5_withoutname1_SendReceive_updateBalanceRatingSTEP := false;
          N6_withoutname1_SendReceive_updateSecurityRatingOUT := false;
          N6_withoutname1_SendReceive_updateSecurityRatingSTEP := false;
        };
        withoutname1IN := false;
        withoutname1OUT := false;
        withoutname1ERR := false;
        withoutname1TOK := 0;
        if withoutname2TOK > 0 {
          N1_withoutname2_DefaultInitialOUT := false;
          N2_withoutname2_fork1OUT := false;
          N2_withoutname2_fork2OUT := false;
          N3_withoutname2_SendReceive_clearDataOUT := false;
          N3_withoutname2_SendReceive_clearDataSTEP := false;
          N4_withoutname2_SendReceive_clearDataOUT := false;
          N4_withoutname2_SendReceive_clearDataSTEP := false;
        };
        withoutname2IN:= false;
        withoutname2OUT := false;
        withoutname2ERR := false;
        withoutname2TOK := 0;
```

```
};
HandleBalanceSecurityDataTOK := 0;
HandleBalanceSecurityDataIN := false;
HandleBalanceSecurityDataERR := false;
HandleBalanceSecurityDataOUT := false;
if RatingCalculationTOK > 0 {
  N1_RatingCalculation_DefaultInitialOUT := false;
  N2_RatingCalculation_SendReceive_calculateRatingOUT := false;
  N2_RatingCalculation_SendReceive_calculateRatingSTEP := false;
};
RatingCalculationTOK := 0;
RatingCalculationIN := false;
RatingCalculationERR := false;
RatingCalculationOUT := false;
if DecisionTOK > 0 {
  N1_Decision_DefaultInitialOUT := false;
  if ApprovalTOK > 0 {
    N1_Approval_DefaultInitialOUT := false;
    N2_Approval_Send_requestClerkApprovalOUT := false;
    N3_Approval_Send_requestSupervisorApprovalOUT := false;
    N4_Approval_Receive_approvalResultOUT := false;
  };
  ApprovalIN := false;
  ApprovalOUT := false;
  ApprovalERR := false;
  ApprovalTOK := 0;
  if AcceptTOK > 0 {
    N1_Accept_DefaultInitialOUT := false;
    N2_Accept_SendReceive_generateOfferOUT := false;
    N2_Accept_SendReceive_generateOfferSTEP := false;
    N3_Accept_SendReceive_offerToClientOUT := false;
    N3_Accept_SendReceive_offerToClientSTEP := false;
    N4_Accept_SendReceive_offerOfferOUT := false;
    N4_Accept_SendReceive_offerOfferSTEP := false;
  };
  AcceptIN:= false;
  AcceptOUT := false;
  AcceptERR := false;
  AcceptTOK := 0;
  if DeclineTOK >0 {
    N1_Decline_DefaultInitialOUT := false;
    N2_Decline_SendReceive_generateDeclineOUT := false;
    N2_Decline_SendReceive_generateDeclineSTEP := false;
    N3_Decline_SendReceive_declineToClientOUT := false;
    N3_Decline_SendReceive_declineToClientSTEP := false;
  };
  DeclineIN := false;
  DeclineOUT := false;
  DeclineERR:= false;
  DeclineTOK := 0;
      };
      DecisionTOK := 0;
      DecisionIN := false;
      DecisionERR := false;
      DecisionOUT := false;
      MainFaultIN := true;
      Priority := NormalPriority + 10;
  }
-- Cancel Handler Completion:
s1 -> s1
  { - [CancelOUT = true] /
      CancelOUT := false;
      MainTOK := MainTOK - 1;
      if (MainTOK = 0 ) {
        MainOUT := true;
      };
  }
-- MAINFAULT -> MAINERROR:
s1 -> s1
  { - [MainFaultERR = true] /
      MainFaultERR := false;
      MainERR := true;
      MainTOK := 0;
  }
-- MAINFAULT -> RECOVERY:
s1 -> s1
  { - [MainFaultOUT = true] /
      MainFaultOUT := false;
      MainOUT := true;
      MainTOK := 0;
  }
-------------------------------------------------------------------------
------------------------ ACTIVITY FINALIZE --------------------------
-------------------------------------------------------------------------
s1 -> s1
  { - [FinalizeIN = true] /
      FinalizeIN := false;
      FinalizeTOK := 1;
      N1_Finalize_DefaultInitialOUT := true;
      Priority := NormalPriority;
  }
-- DEFAULTINITIAL --> <<SEND>> GOODBYE:
s1 -> s1
  { tau [N1_Finalize_DefaultInitialOUT = true] /
        N1_Finalize_DefaultInitialOUT := false;
        LNK_portalService.goodbye([self],VAR_CreditRequest_customerData);
        N2_Finalize_Send_goodbyeOUT := true;
        self.tau;
  }
-- <<SEND>> GOODBYE --> ACTIVITYFINAL:
```

```
s1 -> s1
  { tau [N2_Finalize_Send_goodbyeOUT = true]  /
        N2_Finalize_Send_goodbyeOUT := false;
        FinalizeTOK := 0;
        FinalizeOUT := true;
        self.tau;
  }
-------------------------------------------------------------------------
------------------------- ACTIVITY CANCEL ------------------------------
-------------------------------------------------------------------------
-- DEFAULTINITIAL --> <<RECEIVE>> CANCEL:
s1 -> s1
  { cancel(callid,customerData)
      [MainTOK > 0] /
      CancelIN := true;
      CancelTOK := 1;
      MainTOK := MainTOK+1;
      VAR_Cancel_customerData := customerData;
      N2_Cancel_callid := callid;
      N2_Cancel_Receive_cancelOUT := true;
  }
-- <<RECEIVE>> CANCEL --> <<RAISEEXCEPTION>> PROCESSCANCELED:
s1 -> s1
  { tau [N2_Cancel_Receive_cancelOUT = true] /
        N2_Cancel_Receive_cancelOUT := false;
        N2_Cancel_callid := null;
        VAR_Cancel_customerData := null;
        CancelERR := true;
        CancelTOK :=0;
        self.tau;
  }
-- <<RAISEEXCEPTION>> PROCESSCANCELED --> ACTIVITYFINAL (NEVER occurs!):
s1 -> s1
  { tau [N3_Cancel_Raise_processCanceledOUT = true] /
        N3_Cancel_Raise_processCanceledOUT := false;
        N2_Cancel_callid := null;
        VAR_Cancel_customerData := null;
        CancelTOK := 0;
        CancelOUT := true;
        self.tau;
  }
-------------------------------------------------------------------------
------------------------ ACTIVITY MAINFAULT -----------------------------
-------------------------------------------------------------------------
s1 -> s1
  { - [MainFaultIN = true] /
      MainFaultIN := false;
      MainFaultTOK := 1;
      N1_MainFault_DefaultInitialOUT := true;
      Priority := NormalPriority;
  }
-- DEFAULTINITIAL --> <<COMPENSATEALL>> CLEANUPDATA:
s1 -> s1
  { tau [N1_MainFault_DefaultInitialOUT = true] /
        N1_MainFault_DefaultInitialOUT := false;
        N2_MainFault_CompensateAll_InProgress := true;
        self.tau;
  }
s1 -> s1
  { - [N2_MainFault_CompensateAll_InProgress] /
      N2_MainFault_CompensateAll_InProgress := false;
      RES: bool;
      RES := false;
      TMP: int[];
      TMP := [];
      CURRENT: int;
      DIV: int;
      for index1 in 0 .. ALLINSTALLED.length - 1 {
      -- copy all items except first nested in VAR_Main into TMP:
        CURRENT := ALLINSTALLED[index1];
        DIV := VAR_Main mod CURRENT;
        if ((RES = false) and (DIV /= 0)) {
          TMP := TMP + [CURRENT];
        };
        if (RES = true) {
          TMP := TMP + [CURRENT];
        };
        if ((RES = false) and (DIV = 0)) {
          COMPENSATIONSREQUESTED := COMPENSATIONSREQUESTED + [CURRENT];
          RETURNTO := RETURNTO + [VAR_Main];
          RES := true;
        };
      };
      ALLINSTALLED := TMP;
      TMP := [];
      if (RES = false) {
        VAR_MainFault_COMPENSATEALL_DONE := true;
      };
  }
-- If the requested compensation is done, COMPENSATIONSDONE.head
-- contains the returnpoint so we execute compensateall once more:
s1 -> s1
  { - [(COMPENSATIONSDONE.head = VAR_Main) and
       (COMPENSATIONSMODE.head = true)] /
      COMPENSATIONSDONE := COMPENSATIONSDONE.tail;
      COMPENSATIONSMODE := COMPENSATIONSMODE.tail;
      -- return to compensateall to see if there are other handlers:
      N2_MainFault_CompensateAll_InProgress := true;
  }
-- In case of failure of the handler, we memorize this:
```

```
s1 -> s1
  { - [(COMPENSATIONSDONE.head = VAR_Main) and
       (COMPENSATIONSMODE.head = false)] /
      COMPENSATIONSDONE := COMPENSATIONSDONE.tail;
      COMPENSATIONSMODE := COMPENSATIONSMODE.tail;
      VAR_MainFault_COMPENSATEALL_FAIL := true;
      N2_MainFault_CompensateAll_InProgress := true;
  }
-- If all compensations are done without failures, we proceed normally:
s1 -> s1
  { tau [(VAR_MainFault_COMPENSATEALL_DONE = true) and
         (VAR_MainFault_COMPENSATEALL_FAIL = false)] /
        VAR_MainFault_COMPENSATEALL_DONE := false;
        N2_MainFault_CompensateAll_cleanUpDataOUT := true;
        self.tau;
  }
-- If all compensations are done, but there was a failure, we proceed
-- propagating the exception:
s1 -> s1
  { tau [(VAR_MainFault_COMPENSATEALL_DONE = true) and
         (VAR_MainFault_COMPENSATEALL_FAIL = true)] /
        VAR_MainFault_COMPENSATEALL_DONE := true;
        VAR_MainFault_COMPENSATEALL_FAIL := false;
        N2_MainFault_CompensateAll_cleanUpDataERR := true;
        self.tau;
  }
-- <<COMPENSATEALL>> CLEANUPDATA --> <<SEND&...>> ABORTPROCESS:
s1 -> s1
  { tau [N2_MainFault_CompensateAll_cleanUpDataOUT = true] /
        N2_MainFault_CompensateAll_cleanUpDataOUT := false;
        N2_Cancel_callid := [self,2];
        LNK_portalService.abortProcess(N2_Cancel_callid);
        N3_MainFault_Send_abortProcessOUT := true;
        self.tau;
  }
-- <<...&RECEIVE>> ABORTPROCESS --> ACTIVITYFINAL:
s1 -> s1
  { tau [N3_MainFault_Send_abortProcessOUT = true] /
        N3_MainFault_Send_abortProcessOUT := false;
        MainFaultTOK := MainFaultTOK - 1;
        N2_Cancel_callid := null;
        MainFaultOUT := true;
        MainFaultTOK := 0;
        self.tau;
  }
-- INTERNALERRORS --> FAIL:
s1 -> s1
  { - [N2_MainFault_CompensateAll_cleanUpDataERR = true] /
      N2_MainFault_CompensateAll_cleanUpDataERR := false;
      N2_Cancel_callid := null;
      MainFaultERR := true;
      MainFaultTOK := 0;
  }
-----------------------------------------------------------------------
-------------------------- ACTIVITY CREATION --------------------------
-----------------------------------------------------------------------
s1 -> s1
  { - [CreationIN = true] /
      CreationIN := false;
      CreationTOK := 1;
      N1_Creation_DefaultInitialOUT := true;
      Priority := NormalPriority;
  }
-- DEFAULTINITIAL --> ACTIVITY INITIALIZATION:
s1 -> s1
  { tau [N1_Creation_DefaultInitialOUT = true] /
        N1_Creation_DefaultInitialOUT := false;
        InitializationIN := true;
        Priority := NormalPriority + 10;
        self.tau;
  }
-- INITIALIZATION COMPLETION --> ACTIVITYFINAL:
s1 -> s1
  { tau [InitializationOUT = true] /
        InitializationOUT := false;
        CreationOUT := true;
        CreationTOK := 0;
        self.tau;
  }
-- CREATION FAILURE (without EXCP HANDLER):
s1 -> s1
  { - [InitializationERR = true] /
      InitializationERR := false;
      CreationTOK := 0;
      CreationERR := true;
  }
-- ACTIVATION OF COMPENSATION HANDLER:
s1 -> s1
  { - [COMPENSATIONSREQUESTED.head =
         VAR_CompensateCreditManagementInitialization] /
      COMPENSATIONSREQUESTED := COMPENSATIONSREQUESTED.tail;
      VAR_CompensateCreditManagementInitialization_RETURNTO :=
        RETURNTO.head;
      RETURNTO := RETURNTO.tail;
      CompensateCreditManagementInitializationIN := true;
      Priority := NormalPriority + 10;
  }
-- COMPLETION OF COMPENSATION HANDLER:
s1 -> s1
  { - [CompensateCreditManagementInitializationOUT = true ] /
```

```
      CompensateCreditManagementInitializationOUT := false;
      COMPENSATIONSDONE := COMPENSATIONSDONE +
        [VAR_CompensateCreditManagementInitialization_RETURNTO];
      COMPENSATIONSMODE := COMPENSATIONSMODE + [true];
      VAR_CompensateCreditManagementInitialization_RETURNTO := null;
  }
-- FAILURE OF COMPENSATION HANDLER:
s1 -> s1
  { - [CompensateCreditManagementInitializationERR = true ] /
      CompensateCreditManagementInitializationERR := false;
      COMPENSATIONSDONE := COMPENSATIONSDONE +
        [VAR_CompensateCreditManagementInitialization_RETURNTO];
      COMPENSATIONSMODE := COMPENSATIONSMODE + [false];
      VAR_CompensateCreditManagementInitialization_RETURNTO := null;
  }
-----------------------------------------------------------------------
---------------------- ACTIVITY INITIALIZATION ------------------------
-----------------------------------------------------------------------
s1 -> s1
  { - [InitializationIN = true] /
      InitializationIN := false;
      InitializationTOK := 1;
      N1_Initialization_DefaultInitialOUT := true;
      Priority := NormalPriority;
  }
-- DEFAULTINITIAL --> <<RECEIVE>> CREATENEWCREDITREQUEST:
s1 -> s1
  { createNewCreditRequest(callid,creditData)
      [N1_Initialization_DefaultInitialOUT = true] /
      N1_Initialization_DefaultInitialOUT := false;
      N2_Initialization_callid := callid;
      VAR_CreditRequest_creditData := creditData;
      N2_Initialization_Receive_createNewCreditRequestOUT := true;
  }
-- <<RECEIVE>> CREATENEWCREDITREQUEST --> <<SEND&...>> INITCREDITDATA:
s1 -> s1
  { tau [N2_Initialization_Receive_createNewCreditRequestOUT = true] /
        N2_Initialization_Receive_createNewCreditRequestOUT := false;
        N3_Initialization_callid := [self,2];
        LNK_creditManagementService.initCreditData
          (N3_Initialization_callid,VAR_CreditRequest_creditData);
        N3_Initialization_SendReceive_initCreditDataSTEP := true;
        self.tau;
  }
-- <<SEND&...>> INITCREDITDATA --> <<...&RECEIVE>> INITCREDITDATA:
s1 -> s1
  { initCreditData_return(callid)
      [(N3_Initialization_SendReceive_initCreditDataSTEP = true) and
       (callid = N3_Initialization_callid)] /
      N3_Initialization_SendReceive_initCreditDataSTEP := false;
      N3_Initialization_SendReceive_initCreditDataOUT := true;
  }
-- <<...&RECEIVE>> INITCREDITDATA --> <<REPLY>> CREATENEWCREDITREQUEST:
s1 -> s1
  { tau [N3_Initialization_SendReceive_initCreditDataOUT = true] /
        N3_Initialization_SendReceive_initCreditDataOUT := false;
        LNK_portalService.createNewCreditRequest_return
          (N2_Initialization_callid);
        N4_Initialization_Reply_createNewCreditRequestOUT := true;
        self.tau;
  }
-- <<REPLY>> CREATENEWCREDITREQUEST --> FLOWFINAL:
s1 -> s1
  { tau [N4_Initialization_Reply_createNewCreditRequestOUT = true] /
        N4_Initialization_Reply_createNewCreditRequestOUT := false;
        InitializationTOK := InitializationTOK - 1;
        if (InitializationTOK = 0) {
          N2_Initialization_callid := null;
          N3_Initialization_callid := null;
          InitializationOUT := true;
          -- Set completion handler:
          ALLINSTALLED := [VAR_CompensateCreditManagementInitialization]
            + ALLINSTALLED;
        };
        self.tau;
  }
-----------------------------------------------------------------------
--------- ACTIVITY COMPENSATECREDITMANAGEMENTINITIALIZATION ----------
-----------------------------------------------------------------------
s1 -> s1
  { - [CompensateCreditManagementInitializationIN = true] /
      CompensateCreditManagementInitializationIN := false;
      CompensateCreditManagementInitializationTOK := 1;
      N1_CompensateCreditManagementInitialization_DefaultInitialOUT :=
        true;
  }
-- DEFAULTINITIAL --> <<SEND&...>> INITCREDITDATA:
s1 -> s1
  { tau [N1_CompensateCreditManagementInitialization_DefaultInitialOUT =
          true] /
        N1_CompensateCreditManagementInitialization_DefaultInitialOUT :=
          false;
        N2_CompensateCreditManagementInitialization_callid := [self,2];
        LNK_creditManagementService.removeData
          (N2_CompensateCreditManagementInitialization_callid,
            VAR_CreditRequest_creditData);
        N2_CompensateCreditManagementInitialization_SendReceive_
          removeDataSTEP := true;
        self.tau;
  }
```

```
-- <<SEND&...>> INITCREDITDATA --> <<...&RECEIVE>> INITCREDITDATA:
s1 -> s1
  { removeData_return(callid)
      [(N2_CompensateCreditManagementInitialization_SendReceive_
          removeDataSTEP = true) and
       (callid = N2_CompensateCreditManagementInitialization_callid)] /
      N2_CompensateCreditManagementInitialization_SendReceive_
        removeDataSTEP := false;
      N2_CompensateCreditManagementInitialization_SendReceive_
        removeDataOUT := true;
  }
-- <<...&RECEIVE>> INITCREDITDATA --> FLOWFINAL:
s1 -> s1
  { tau [N2_CompensateCreditManagementInitialization_SendReceive_
           removeDataOUT = true] /
         N2_CompensateCreditManagementInitialization_SendReceive_
           removeDataOUT := false;
         CompensateCreditManagementInitializationTOK :=
           CompensateCreditManagementInitializationTOK - 1;
         if (CompensateCreditManagementInitializationTOK = 0) {
           N2_CompensateCreditManagementInitialization_callid := null;
           CompensateCreditManagementInitializationOUT := true;
         };
         self.tau;
  }
-----------------------------------------------------------------------
------------------ ACTIVITY HANDLEBALANCESECURITY --------------------
-----------------------------------------------------------------------
s1 -> s1
  { - [HandleBalanceSecurityDataIN = true] /
      HandleBalanceSecurityDataIN := false;
      HandleBalanceSecurityDataTOK := 1;
      N1_HandleBalanceSecurityData_DefaultInitialOUT := true;
      Priority := NormalPriority;
  }
-- DEFAULTINITIAL --> ACTIVITY withoutname1:
s1 -> s1
  { tau [N1_HandleBalanceSecurityData_DefaultInitialOUT = true] /
        N1_HandleBalanceSecurityData_DefaultInitialOUT := false;
        withoutname1IN := true;
        Priority := NormalPriority + 10;
        self.tau;
  }
-- ACTIVITY withoutname1 --> ACTIVITYFINAL:
s1 -> s1
  { tau [withoutname1OUT = true] /
        withoutname1OUT := false;
        HandleBalanceSecurityDataOUT := true;
        HandleBalanceSecurityDataTOK := 0;
        self.tau;
  }
-- HANDLEBALANCESECURITY FAILURE (without EXCP HANDLER):
s1 -> s1
  { - [withoutname1ERR = true] /
      withoutname1ERR := false;
      HandleBalanceSecurityDataTOK := 0;
      HandleBalanceSecurityDataERR := true;
  }
-- ACTIVATION OF COMPENSATION HANDLER:
s1 -> s1
  { - [COMPENSATIONSREQUESTED.head = VAR_withoutname2] /
      COMPENSATIONSREQUESTED := COMPENSATIONSREQUESTED.tail;
      VAR_withoutname2_RETURNTO := RETURNTO.head;
      RETURNTO := RETURNTO.tail;
      withoutname2IN := true;
      Priority := NormalPriority + 10;
  }
-- COMPLETION OF COMPENSATION HANDLER:
s1 -> s1
  { - [withoutname2OUT = true] /
      withoutname2OUT := false;
      COMPENSATIONSDONE := COMPENSATIONSDONE +
        [VAR_withoutname2_RETURNTO];
      COMPENSATIONSMODE := COMPENSATIONSMODE + [true];
      VAR_withoutname2_RETURNTO := null;
  }
-- FAILURE OF COMPENSATION HANDLER:
s1 -> s1
  { - [withoutname2ERR = true] /
      withoutname2ERR := false;
      COMPENSATIONSDONE := COMPENSATIONSDONE +
        [VAR_withoutname2_RETURNTO];
      COMPENSATIONSMODE := COMPENSATIONSDONE + [false];
      VAR_withoutname2_RETURNTO := null;
  }
-----------------------------------------------------------------------
----------------------- ACTIVITY withoutname1 -------------------------
-----------------------------------------------------------------------
s1 -> s1
  { - [withoutname1IN = true] /
      withoutname1IN := false;
      withoutname1TOK := 1;
      N1_withoutname1_DefaultInitialOUT := true;
      Priority := NormalPriority;
  }
-- DEFAULTINITIAL --> FORK1,FORK2:
s1 -> s1
  { tau [N1_withoutname1_DefaultInitialOUT = true] /
        N1_withoutname1_DefaultInitialOUT := false;
        N2_withoutname1_fork1OUT := true;
```

```
        N2_withoutname1_fork2OUT := true;
        withoutname1TOK := withoutname1TOK + 1; -- for the fork
        self.tau;
  }
-- FORK1 --> <<SEND&...>> ENTERBALANCEDATA:
s1 -> s1
  { tau [N2_withoutname1_fork1OUT = true] /
        N2_withoutname1_fork1OUT := false;
        N3_withoutname1_callid := [self,3];
        LNK_portalService.enterBalanceData(N3_withoutname1_callid,
          VAR_CreditRequest_customerData);
        N3_withoutname1_SendReceive_enterBalanceDataSTEP := true;
        self.tau;
  }
-- <<SEND&...>> ENTERBALANCEDATA --> <<...&RECEIVE>> ENTERBALANCEDATA:
s1 -> s1
  { enterBalanceData_return(callid,result)
      [(N3_withoutname1_SendReceive_enterBalanceDataSTEP = true) and
       (callid = N3_withoutname1_callid)] /
      N3_withoutname1_SendReceive_enterBalanceDataSTEP := false;
      VAR_withoutname1_balancePackage := result;
      N3_withoutname1_SendReceive_enterBalanceDataOUT := true;
  }
-- FORK2 --> <<SEND&...>> ENTERSECURITYDATA:
s1 -> s1
  { tau [N2_withoutname1_fork2OUT = true] /
        N2_withoutname1_fork2OUT := false;
        N4_withoutname1_callid := [self,4];
        LNK_portalService.enterSecurityData(N4_withoutname1_callid,
          VAR_CreditRequest_customerData);
        N4_withoutname1_SendReceive_enterSecurityDataSTEP := true;
        self.tau;
  }
-- <<SEND&...>> ENTERSECURITYDATA --> <<...&RECEIVE>> ENTERSECURITYDATA:
s1 -> s1
  { enterSecurityData_return(callid,result)
      [(N4_withoutname1_SendReceive_enterSecurityDataSTEP = true) and
       (callid = N4_withoutname1_callid)] /
      N4_withoutname1_SendReceive_enterSecurityDataSTEP := false;
      VAR_withoutname1_securityPackage := result;
      N4_withoutname1_SendReceive_enterSecurityDataOUT := true;
  }
-- <<...&RECEIVE>> ENTERBALANCEDATA --> <<SEND&...>> UPDATEBALANCERATING:
s1 -> s1
  { tau [N3_withoutname1_SendReceive_enterBalanceDataOUT = true] /
        N3_withoutname1_SendReceive_enterBalanceDataOUT := false;
        N5_withoutname1_callid := [self,5];
        LNK_balanceService.updateBalanceRating(N5_withoutname1_callid,
          VAR_CreditRequest_customerData,
            VAR_withoutname1_balancePackage);
        N5_withoutname1_SendReceive_updateBalanceRatingSTEP := true;
        self.tau;
  }
-- <<SEND&...>> UPDATEBALANCERATING -->
--   <<...&RECEIVE>> UPDATEBALANCERATING:
s1 -> s1
  { updateBalanceRating_return(callid)
      [(N5_withoutname1_SendReceive_updateBalanceRatingSTEP = true) and
       (callid = N5_withoutname1_callid)] /
      N5_withoutname1_SendReceive_updateBalanceRatingSTEP := false;
      N5_withoutname1_SendReceive_updateBalanceRatingOUT := true;
  }
-- <<...&RECEIVE>> ENTERSECURITYDATA -->
--   <<SEND&...>> UPDATESECURITYRATING:
s1 -> s1
  { tau [N4_withoutname1_SendReceive_enterSecurityDataOUT = true] /
        N4_withoutname1_SendReceive_enterSecurityDataOUT := false;
        N6_withoutname1_callid := [self,6];
        LNK_securityService.updateSecurityRating(N6_withoutname1_callid,
          VAR_CreditRequest_customerData,
            VAR_withoutname1_securityPackage);
        N6_withoutname1_SendReceive_updateSecurityRatingSTEP := true;
        self.tau;
  }
-- <<SEND&...>> UPDATESECURITYRATING -->
--   <<...&RECEIVE>> UPDATESECURITYRATING:
s1 -> s1
  { updateSecurityRating_return(callid)
      [(N6_withoutname1_SendReceive_updateSecurityRatingSTEP = true) and
       (callid = N6_withoutname1_callid)] /
      N6_withoutname1_SendReceive_updateSecurityRatingSTEP := false;
      N6_withoutname1_SendReceive_updateSecurityRatingOUT := true;
  }
-- JOIN --> FLOWFINAL:
s1 -> s1
  { tau [(N5_withoutname1_SendReceive_updateBalanceRatingOUT = true) and
         (N6_withoutname1_SendReceive_updateSecurityRatingOUT = true)] /
        N5_withoutname1_SendReceive_updateBalanceRatingOUT := false;
        N6_withoutname1_SendReceive_updateSecurityRatingOUT := false;
        withoutname1TOK := withoutname1TOK - 1; -- for the join
        withoutname1TOK := withoutname1TOK - 1; -- for the flowfinal
        if (withoutname1TOK = 0) {
          N3_withoutname1_callid := null;
          N4_withoutname1_callid := null;
          N5_withoutname1_callid := null;
          N6_withoutname1_callid := null;
          -- compensation handlers:
          ALLINSTALLED := [VAR_withoutname2]  + ALLINSTALLED;
          withoutname1OUT := true;
        };
```

```
        self.tau;
  }
-- When an activity WITH A COMPENSATION HANDLER ends succesfully (ie.
-- inside: FLOW_FINAL, ACTIVITY_FINAL, SUCCESSFUL_EXCEPTION_HANDLER),
-- then its compensation handler is installed in the following way:
--   ALLINSTALLED := [VAR_CompensationHandlerName] + ALLINSTALLED;
-----------------------------------------------------------------------
---------------------- ACTIVITY withoutname2 ------------------------
-----------------------------------------------------------------------
s1 -> s1
  { - [withoutname2IN = true] /
      withoutname2IN := false;
      withoutname2TOK := 1;
      N1_withoutname2_DefaultInitialOUT := true;
      Priority := NormalPriority;
  }
-- DEFAULTINITIAL --> FORK1,FORK2:
s1 -> s1
  { - [N1_withoutname2_DefaultInitialOUT = true] /
      N1_withoutname2_DefaultInitialOUT := false;
      N2_withoutname2_fork1OUT := true;
      N2_withoutname2_fork2OUT := true;
      withoutname2TOK := withoutname2TOK + 1; -- for the fork
  }
-- FORK1 --> <<SEND&...>> CLEARDATA:
s1 -> s1
  { tau [N2_withoutname2_fork1OUT = true] /
        N2_withoutname2_fork1OUT := false;
        N3_withoutname2_callid := [self,3];
        LNK_balanceService.clearData(N3_withoutname2_callid,
          VAR_CreditRequest_customerData);
        N3_withoutname2_SendReceive_clearDataSTEP := true;
        self.tau;
  }
-- <<SEND&...>> CLEARDATA --> <<...&RECEIVE>> CLEARDATA:
s1 -> s1
  { clearData_return(callid)
      [(N3_withoutname2_SendReceive_clearDataSTEP = true) and
       (callid = N3_withoutname2_callid)] /
      N3_withoutname2_SendReceive_clearDataSTEP := false;
      N3_withoutname2_SendReceive_clearDataOUT := true;
  }
-- FORK2 --> <<SEND&...>> CLEARDATA:
s1 -> s1
  { tau [N2_withoutname2_fork2OUT = true] /
        N2_withoutname2_fork2OUT := false;
        N4_withoutname2_callid := [self,4];
        LNK_securityService.clearData(N4_withoutname2_callid,
          VAR_CreditRequest_customerData);
        N4_withoutname2_SendReceive_clearDataSTEP := true;
        self.tau;
  }
-- <<SEND&...>> CLEARDATA --> <<...&RECEIVE>> CLEARDATA:
s1 -> s1
  { clearData_return(callid)
      [(N4_withoutname2_SendReceive_clearDataSTEP = true) and
       (callid = N4_withoutname2_callid)] /
      N4_withoutname2_SendReceive_clearDataSTEP := false;
      N4_withoutname2_SendReceive_clearDataOUT := true;
  }
-- ACTIVITY withoutname2 --> FLOWFINAL:
s1 -> s1
  { tau [(N3_withoutname2_SendReceive_clearDataOUT = true) and
         (N4_withoutname2_SendReceive_clearDataOUT = true)] /
        N3_withoutname2_SendReceive_clearDataOUT := false;
        N4_withoutname2_SendReceive_clearDataOUT := false;
        withoutname2TOK := withoutname2TOK - 1; -- for the join
        withoutname2TOK := withoutname2TOK - 1; -- for the flowfinal
        if (withoutname2TOK = 0 ) {
          N3_withoutname2_callid := null;
          N4_withoutname2_callid := null;
          withoutname2OUT := true;
          withoutname2TOK := 0;
        };
        self.tau;
  }
-----------------------------------------------------------------------
---------------------- ACTIVITY RATINGCALCULATION --------------------
-----------------------------------------------------------------------
s1 -> s1
  { - [RatingCalculationIN = true] /
      RatingCalculationIN := false;
      RatingCalculationTOK := 1;
      N1_RatingCalculation_DefaultInitialOUT := true;
      Priority := NormalPriority;
  }
-- DEFAULTINITIAL --> <<SEND&...>> CALCULATERATING:
s1 -> s1
  { tau [N1_RatingCalculation_DefaultInitialOUT = true] /
        N1_RatingCalculation_DefaultInitialOUT := false;
        N2_RatingCalculation_callid := [self,2];
        LNK_ratingService.calculateRating(N2_RatingCalculation_callid,
          VAR_CreditRequest_customerData);
        N2_RatingCalculation_SendReceive_calculateRatingSTEP := true;
        self.tau;
  }
-- <<SEND&...>> CALCULATERATING --> <<...&RECEIVE>> CALCULATERATING:
s1 -> s1
  { calculateRating_return(callid,result:Token)
      [(N2_RatingCalculation_SendReceive_calculateRatingSTEP = true) and
```

```
       (callid = N2_RatingCalculation_callid)] /
      N2_RatingCalculation_SendReceive_calculateRatingSTEP := false;
      VAR_CreditRequest_ratingData := [0,0,0,0,0,0,0,result,0];
      N2_RatingCalculation_SendReceive_calculateRatingOUT := true;
  }
-- N3 <<...&RECEIVE>> CALCULATERATING --> ACTIVITYFINAL:
s1 -> s1
  { tau [N2_RatingCalculation_SendReceive_calculateRatingOUT = true]  /
        N2_RatingCalculation_SendReceive_calculateRatingOUT := false;
        RatingCalculationTOK := 0;
        RatingCalculationOUT := true;
        self.tau;
  }
-------------------------------------------------------------------------
------------------------ ACTIVITY DECISION ------------------------------
-------------------------------------------------------------------------
s1 -> s1
  { - [DecisionIN = true] /
      DecisionIN := false;
      DecisionTOK := 1;
      N1_Decision_DefaultInitialOUT := true;
      Priority := NormalPriority;
  }
-- DEFAULTINITIAL --> ACTIVITY APPROVAL:
s1 -> s1
  { tau [(N1_Decision_DefaultInitialOUT = true) and
         not (VAR_CreditRequest_ratingData[7] = AAA)] / -- 7 = result
        N1_Decision_DefaultInitialOUT := false;
        ApprovalIN := true;
        Priority := NormalPriority + 10;
        self.tau;
  }
-- ACTIVITY APPROVAL, DEFAULTINITIAL --> ACCEPT:
s1 -> s1
  { tau [((VAR_CreditRequest_ratingData[7] = AAA) or -- 7 = result, 8 =
           (VAR_CreditRequest_ratingData[8] = true)) -- manualAcceptance
         and (((VAR_CreditRequest_ratingData[7] = AAA) and
               (N1_Decision_DefaultInitialOUT = true)) or
              ((not VAR_CreditRequest_ratingData[7] = AAA) and
               ( ApprovalOUT = true)))] /
        N1_Decision_DefaultInitialOUT := false;
        ApprovalOUT := false;
        AcceptIN := true;
        Priority := NormalPriority + 10;
        self.tau;
  }
-- ACTIVITY APPROVAL, DEFAULTINITIAL --> DECLINE:
s1 -> s1
  { tau [(not ((VAR_CreditRequest_ratingData[7] = AAA) or -- 7 = result,
               (VAR_CreditRequest_ratingData[8] = true))) -- 8 = manual-
          and (((VAR_CreditRequest_ratingData[7] = AAA) and -- Acceptance
                (N1_Decision_DefaultInitialOUT = true)) or
               ((not VAR_CreditRequest_ratingData[7] = AAA) and
                (ApprovalOUT = true)))] /
        N1_Decision_DefaultInitialOUT := false;
        ApprovalOUT := false;
        DeclineIN := true;
        Priority := NormalPriority + 10;
        self.tau;
  }
-- ACTIVITY ACCEPT/DECLINE --> ACTIVITYFINAL:
s1 -> s1
  { tau [(AcceptOUT = true) or
         (DeclineOUT = true)] /
        AcceptOUT := false;
        DeclineOUT := false;
        DecisionOUT := true;
        DecisionTOK := 0;
        self.tau;
  }
-- ERRORS --> ACTIVITY DECISION FAILURE:
s1 -> s1
  { - [(AcceptERR = true) or (DeclineERR = true) or (ApprovalERR)] /
      ApprovalERR := false;
      AcceptERR := false;
      DeclineERR := false;
      DecisionERR := true;
      DecisionTOK := 0;
  }
-------------------------------------------------------------------------
------------------------- ACTIVITY APPROVAL -----------------------------
-------------------------------------------------------------------------
s1 -> s1
  { - [ApprovalIN = true] /
      ApprovalIN := false;
      ApprovalTOK := 1;
      N1_Approval_DefaultInitialOUT := true;
      Priority := NormalPriority;
  }
-- DEFAULTINITIAL --> <<SEND>> REQUESTCLERKAPPROVAL:
s1 -> s1
  { tau [(VAR_CreditRequest_ratingData[7] = BBB) and -- 7 = result
         (N1_Approval_DefaultInitialOUT = true)] /
        N1_Approval_DefaultInitialOUT := false;
        LNK_portalService.requestClerkApproval([self],
          VAR_CreditRequest_ratingData);
        N2_Approval_Send_requestClerkApprovalOUT := true;
        self.tau;
  }
-- DEFAULTINITIAL --> <<SEND>> REQUESTSUPERVISORAPPROVAL:
```

```
s1 -> s1
  { tau [(not VAR_CreditRequest_ratingData[7] = BBB) and -- 7 = result
         (N1_Approval_DefaultInitialOUT = true)] /
        N1_Approval_DefaultInitialOUT := false;
        LNK_portalService.requestSupervisorApproval([self],
          VAR_CreditRequest_ratingData);
        N3_Approval_Send_requestSupervisorApprovalOUT := true;
        self.tau;
  }
-- <<SEND>> REQUESTSUPERVISORAPPROVAL --> <<RECEIVE>> APPROVALRESULT:
s1 -> s1
  { approvalResult(callid,ratingData)
      [(N3_Approval_Send_requestSupervisorApprovalOUT = true) or
       (N2_Approval_Send_requestClerkApprovalOUT = true)] /
      N3_Approval_Send_requestSupervisorApprovalOUT := false;
      N2_Approval_Send_requestClerkApprovalOUT := false;
      VAR_CreditRequest_ratingData := ratingData;
      N4_Approval_callid := callid;
      N4_Approval_Receive_approvalResultOUT := true;
  }
-- <<RECEIVE>> APPROVALRESULT --> ACTIVITYFINAL:
s1 -> s1
  { tau [N4_Approval_Receive_approvalResultOUT = true] /
        N4_Approval_Receive_approvalResultOUT := false;
        ApprovalTOK := 0;
        ApprovalOUT := true;
        self.tau;
  }
-----------------------------------------------------------------------
------------------------- ACTIVITY ACCEPT -----------------------------
-----------------------------------------------------------------------
s1 -> s1
  { - [AcceptIN = true] /
      AcceptIN := false;
      AcceptTOK := 1;
      N1_Accept_DefaultInitialOUT := true;
      Priority := NormalPriority;
  }
-- DEFAULTINITIAL --> <<SEND&...>> GENERATEOFFER:
s1 -> s1
  { tau [N1_Accept_DefaultInitialOUT = true] /
        N1_Accept_DefaultInitialOUT := false;
        N2_Accept_callid := [self,2];
        LNK_creditManagementService.generateOffer(N2_Accept_callid,
          VAR_CreditRequest_ratingData);
        N2_Accept_SendReceive_generateOfferSTEP := true;
        self.tau;
  }
-- <<SEND&...>> GENERATEOFFER --> <<...&RECEIVE>> GENERATEOFFER:
s1 -> s1
  { generateOffer_return(callid,agreement)
      [(N2_Accept_SendReceive_generateOfferSTEP = true) and
       (callid = N2_Accept_callid)] /
      N2_Accept_SendReceive_generateOfferSTEP := false;
      VAR_Accept_agreement := agreement;
      N2_Accept_SendReceive_generateOfferOUT := true;
  }
-- <<...&RECEIVE>> GENERATEOFFER --> <<SEND&...>> OFFERTOCLIENT:
s1 -> s1
  { tau [N2_Accept_SendReceive_generateOfferOUT = true] /
        N2_Accept_SendReceive_generateOfferOUT := false;
        N3_Accept_callid := [self,3];
        LNK_portalService.offerToClient(N3_Accept_callid,
          VAR_Accept_agreement);
        N3_Accept_SendReceive_offerToClientSTEP := true;
        self.tau;
  }
-- <<SEND&...>> OFFERTOCLIENT --> <<...&RECEIVE>> OFFERTOCLIENT:
s1 -> s1
  { offerToClient_return(callid,accepted)
      [(N3_Accept_SendReceive_offerToClientSTEP = true) and
       (callid = N3_Accept_callid)] /
      N3_Accept_SendReceive_offerToClientSTEP := false;
      VAR_Accept_accepted := accepted;
      N3_Accept_SendReceive_offerToClientOUT := true;
  }
-- <<...&RECEIVE>> OFFERTOCLIENT --> <<SEND&...>> ACCEPTOFFER:
s1 -> s1
  { tau [(VAR_Accept_accepted = true) and
         (N3_Accept_SendReceive_offerToClientOUT = true) ] /
        N3_Accept_SendReceive_offerToClientOUT := false;
        N4_Accept_callid := [self,4];
        LNK_creditManagementService.acceptOffer(N4_Accept_callid,
          VAR_Accept_agreement);
        N4_Accept_SendReceive_offerOfferSTEP := true;
        self.tau;
  }
-- <<SEND&...>> ACCEPTOFFER --> <<...&RECEIVE>> ACCEPTOFFER:
s1 -> s1
  { acceptOffer_return(callid)
      [(N4_Accept_SendReceive_offerOfferSTEP = true) and
       (callid = N4_Accept_callid)] /
      N4_Accept_SendReceive_offerOfferSTEP := false;
      N4_Accept_SendReceive_offerOfferOUT := true;
}
-- <<...&RECEIVE>> ACCEPTOFFER --> ACTIVITYFINAL:
s1 -> s1
  { tau [(N4_Accept_SendReceive_offerOfferOUT = true) or
         ((N3_Accept_SendReceive_offerToClientOUT = true) and
          (not VAR_Accept_accepted = true))] /
```

```
        N4_Accept_SendReceive_offerOfferOUT := false;
        N3_Accept_SendReceive_offerToClientOUT := false;
        N2_Accept_callid:= null;
        N3_Accept_callid:= null;
        N4_Accept_callid := null;
        VAR_Accept_agreement := null;
        VAR_Accept_accepted := null;
        AcceptOUT := true;
        AcceptTOK := 0;
        self.tau;
  }
-------------------------------------------------------------------------
------------------------- ACTIVITY DECLINE ------------------------------
-------------------------------------------------------------------------
s1 -> s1
  { - [DeclineIN = true] /
      DeclineIN := false;
      DeclineTOK := 1;
      N1_Decline_DefaultInitialOUT := true;
      Priority := NormalPriority;
  }
-- DEFAULTINITIAL --> <<SEND&...>> GENERATEDECLINE:
s1 -> s1
  { tau [N1_Decline_DefaultInitialOUT = true] /
        N1_Decline_DefaultInitialOUT := false;
        N2_Decline_callid := [self,2];
        LNK_creditManagementService.generateDecline(N2_Decline_callid,
          VAR_CreditRequest_ratingData);
        N2_Decline_SendReceive_generateDeclineSTEP := true;
        self.tau;
  }
-- <<SEND&...>> GENERATEDECLINE --> <<...&RECEIVE>> GENERATEDECLINE:
s1 -> s1
  { generateDecline_return(callid,decline)
      [(N2_Decline_SendReceive_generateDeclineSTEP = true) and
       (callid = N2_Decline_callid)] /
      N2_Decline_SendReceive_generateDeclineSTEP := false;
      VAR_Decline_decline := decline;
      N2_Decline_SendReceive_generateDeclineOUT := true;
  }
-- <<...&RECEIVE>> GENERATEDECLINE --> <<SEND&...>> DECLINETOCLIENT:
s1 -> s1
  { tau [N2_Decline_SendReceive_generateDeclineOUT = true] /
        N2_Decline_SendReceive_generateDeclineOUT := false;
        N3_Decline_callid := [self,3];
        LNK_portalService.declineToClient(N3_Decline_callid,
          VAR_Decline_decline);
        N3_Decline_SendReceive_declineToClientSTEP := true;
        self.tau;
  }
-- <<SEND&...>> DECLINETOCLIENT --> <<...&RECEIVE>> DECLINETOCLIENT:
s1 -> s1
  { declineToClient_return(callid,updateDesired)
      [(N3_Decline_SendReceive_declineToClientSTEP = true) and
       (callid = N3_Decline_callid)] /
      N3_Decline_SendReceive_declineToClientSTEP := false;
      VAR_CreditRequest_updateDesired := updateDesired;
      N3_Decline_SendReceive_declineToClientOUT := true;
  }
-- <<...&RECEIVE>> DECLINETOCLINET --> ACTIVITYFINAL:
s1 -> s1
  { tau [N3_Decline_SendReceive_declineToClientOUT = true] /
        N3_Decline_SendReceive_declineToClientOUT := false;
        N2_Decline_callid := null;
        N3_Decline_callid := null;
        VAR_Decline_decline := null;
        DeclineOUT := true;
        DeclineTOK := 0;
       self.tau;
  }

end CreditRequest
-------------------------------------------------------------------------
------------------- CREDIT MANAGEMENT SERVICE ---------------------------
-------------------------------------------------------------------------
Class CreditManagementService is

Signals
  initCreditData(callid,creditData);  -- send&receive from CreditRequest
  removeData(callid,creditData);  -- send&receive from CreditRequest
  generateOffer(callid,ratingData); -- send&receive from CreditRequest
  acceptOffer(agreement); -- send&receive from CreditRequest
  generateDecline(callid,ratingData); -- send&receive from CreditRequest
  acceptOffer(callid,agreement); -- send&receive from CreditRequest

Vars:
  Priority: int :=3;
  VAR_agreement: int[];

State Top = s1,s2

Transitions:
  -- initCreditData(callid,creditData)
  s1 -> s1
    { initCreditData(callid,creditData) /
        VAR_agreement[0] := creditData[0];
        Caller: obj;
        Caller := callid[0];
        Caller.initCreditData_return(callid);
    }
```

```
  -- removeData(callid,creditData)
  s1 -> s2
    { removeData(callid,creditData) /
        VAR_agreement := null;
        Caller: obj;
        Caller := callid[0];
        Caller.removeData_return(callid);
    }
  -- generateOffer(callid,ratingData: RatingData): AgreementData
  s1 -> s1
    { generateOffer(callid,ratingData) /
        Caller: obj;
        Caller := callid[0];
        Caller.generateOffer_return(callid,VAR_agreement);
    }
  -- acceptOffer(agreement: AgreementData)
  s1 -> s2
    { acceptOffer(callid,agreement) /
        Caller: obj;
        Caller := callid[0];
        Caller.acceptOffer_return(callid);
    }
  -- generateDecline(callid,ratingData: RatingData): AgreementData
  s1 -> s1
    { generateDecline(callid,ratingData) /
        Caller: obj;
        Caller := callid[0];
        Caller.generateDecline_return(callid,VAR_agreement);
    }

end CreditManagementService
-------------------------------------------------------------------------
---------------- CUSTOMER MANAGEMENT SERVICE ----------------------------
-------------------------------------------------------------------------
Class CustomerManagementService is

Signals
  checkUser (callid,userData); -- send&receive from CreditRequest
  getCustomerdata (callid,userData); -- send&receive from CreditRequest

Vars:
  Priority: int := 3;
  VAR_customerData: int[];

State Top = s1

Transitions:
  -- checkUser(callid,userData: UserData): Boolean [userOK]
  s1 -> s1
    { checkUser(callid,userData) /
        Caller: obj;
        Caller := callid[0];
        Caller.checkUser_return(callid,true);
    }
  -- checkUser(callid,userData: UserData): Boolean [userNotOK]
  s1 -> s1
    { checkUser(callid,userData) /
        Caller: obj;
        Caller := callid[0];
        Caller.checkUser_return(callid,false);
    }
  -- getCustomerdata(callid,userData: UserData): CustomerData
  s1 -> s1
    { getCustomerdata(callid,userData) /
        Caller: obj;
        Caller := callid[0];
        Caller.getCustomerdata_return(callid,VAR_customerData);
    }

end CustomerManagementService
-------------------------------------------------------------------------
------------------------------- CLIENT ----------------------------------
-------------------------------------------------------------------------
Class Client is

Signals:
  initialize_return(callid, result:bool);
  createNewCreditRequest_return(callid);
  --
  offerToClient(callid,agreement); -- send&receive from Portal service
  declineToClient(callid,agreement); -- send&receive from Portal service
  --
  abortProcess(callid); -- send&receive from Portal service
  goodbye(callid,customerData); -- send from Portal service

Vars:
  Priority: int :=2;
  LNK_portal: obj;
  VAR_userdata := [self];
  VAR_creditData := [self];
  VAR_customerData := [self];

State Top = s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s20,s24,s25,s26,s27

Transitions:
  -- INITIALIZE:
  s1 -> s2
    { - / LNK_portal.initialize([self],VAR_userdata);
    }
  s2 -> s3
```

```
  { initialize_return(callid,result:bool) [result = false] /
  }
s2 -> s4
  { initialize_return(callid,result:bool) [result = true] /
  }
-- CREATENEWCREDITREQUEST:
s4 -> s5
  { - / LNK_portal.createNewCreditRequest([self],VAR_creditData);
  }
s5 -> s6
  { createNewCreditRequest_return(callid)
  }
s4 -> s20
  { - / LNK_portal.cancel([self],VAR_customerData);
  }
-- CANCEL AFTER CREATENEWCREDITREQUEST:
s6 -> s20
  { - / LNK_portal.cancel([self],VAR_customerData);
  }
s6 -> s7
-- offerToClient(callid,agreement:AgreementData): Boolean
s7 -> s8
  { offerToClient(callid,agreement) /
      LNK_portal.offerToClient_return(callid,true);
  }
s7 -> s9
  { offerToClient(callid,agreement) /
      LNK_portal.offerToClient_return(callid,false);
  }
-- CANCEL AFTER offerToClient --> true:
s8 -> s20
  { - / LNK_portal.cancel([self],VAR_customerData);
  }
s8 -> s20
-- CANCEL AFTER offerToClient --> false
-- CANCEL AFTER declineToClient --> false
s9 -> s20
  { - / LNK_portal.cancel([self],VAR_customerData);
  }
s9 -> s20
-- CANCEL AFTER offerToClient --> true:
s7 -> s24
  { declineToClient(callid,agreement) /
      LNK_portal.declineToClient_return(callid,true);
  }
s7 -> s9
  { declineToClient(callid,agreement) /
      LNK_portal.declineToClient_return(callid,false);
  }
-- SECOND CREATENEWCREDITREQUEST:
  s24 -> s25
    { - / LNK_portal.createNewCreditRequest([self],VAR_creditData);
    }
  s25 -> s26
    { createNewCreditRequest_return(callid)
    }
  s24 -> s20
    { - / LNK_portal.cancel([self],VAR_customerData);
    }
  -- CANCEL AFTER CREATENEWCREDITREQUEST:
  s26 -> s20
    { - / LNK_portal.cancel([self],VAR_customerData);
    }
  s26 -> s27
  -- offerToClient(callid,agreement:AgreementData): Boolean
  s27 -> s8
    { offerToClient(callid,agreement) /
        LNK_portal.offerToClient_return(callid,true);
    }
  s27 -> s9
    { offerToClient(callid,agreement) /
        LNK_portal.offerToClient_return(callid,false);
    }
  -- CANCEL AFTER offerToClient --> true:
  s27 -> s9
    { declineToClient(callid,agreement) /
        LNK_portal.declineToClient_return(callid,false);
    }

end Client
------------------------------------------------------------------------
---------------------------- PORTAL SERVICE ----------------------------
------------------------------------------------------------------------
Class PortalService is

Signals
  -- consumer:
  cancel(callid,customerData); -- send from Client
  initialize(callid,userData); -- send&receive from Client
  initialize_return(callid,result); -- reply from CreditRequest
  createNewCreditRequest(callid,creditData); -- send&receive from Client
  createNewCreditRequest_return(callid); -- reply from CreditRequest
  enterSecurityData(callid,customerData); --send&receive from CreditRequest
  enterBalanceData(callid,customerData); -- send&receive from CreditRequest
  requestClerkApproval(callid,ratingData); -- send from CreditRequest
  requestSupervisorApproval(callid,ratingData); -- send from CreditRequest
  offerToClient(callid,agreement); -- send&receive from CreditRequest
  offerToClient_return(callid,accepted:bool); -- reply from CreditRequest
  declineToClient(callid,agreement); -- send&receive from CreditRequest
  declineToClient_return(callid,updateDesired:bool); -- reply from Client
```

```
  -- provider:
  abortProcess(callid); -- send&receive from CreditRequest
  abortProcess_return(callid); -- reply from Client
  goodbye(callid,customerData); -- send from CreditRequest

Vars
   Priority:int := 3;
   LNK_creditRequest;
   client_callid: int[];
   VAR_balancePackage: int[];
   VAR_securityPackage: int[];
   VAR_ratingData: int[];
   VAR_userData: int[];

State Top = s1

Transitions:
  ------------------------------------------------------------------------
  -------------------------- PORTAL CONSUMER -----------------------------
  ------------------------------------------------------------------------
  -- initialize(userData): Boolean
  s1 -> s1
    { initialize(callid,userData) /
        client_callid := callid;
        Tmp: Token[] := callid;
        LNK_creditRequest.initialize([self] + Tmp,userData);
    }
  -- WAITS REPLY FROM INITIALIZE:
  s1 -> s1
    { initialize_return(callid,result) /
        Client: obj;
        Client := callid[1];
        Client.initialize_return(callid.tail, result);
    }
  -- createNewCreditRequest(creditData: CreditData)
  s1 -> s1
    { createNewCreditRequest(callid,creditData) /
        client_callid := callid;
        Tmp: Token[] := callid;
        LNK_creditRequest.createNewCreditRequest([self] + Tmp,creditData);
    }
  -- createNewCreditRequest_return(callid)
  s1 -> s1
    { createNewCreditRequest_return(callid) /
        Client: obj;
        Client := callid[1];
        Client.createNewCreditRequest_return(callid.tail);
    }
  -- cancel(customerData: CustomerData)
  s1 -> s1
    { cancel(callid,customerData) /
        Tmp: Token[] := callid;
        LNK_creditRequest.cancel([self] + Tmp,customerData);
    }
  ------------------------------------------------------------------------
  -------------------------- PORTAL PROVIDER -----------------------------
  ------------------------------------------------------------------------
  -- enterBalanceData(callid,customerData: CustomerData): BalancePackage
  s1 -> s1
    { enterBalanceData(callid,customerData) /
        Caller: obj;
        Caller := callid[0];
        Caller.enterBalanceData_return(callid,VAR_balancePackage);
    }
  -- enterSecurityData(callid,customerData: CustomerData): SecurityPackage
  s1 -> s1
    { enterSecurityData(callid,customerData) /
        Caller: obj;
        Caller := callid[0];
        Caller.enterSecurityData_return(callid,VAR_securityPackage);
    }
  -- requestClerkApproval(ratingData: RatingData)
  s1 -> s1
    { requestClerkApproval(callid,ratingData) /
        VAR_ratingData := ratingData;
        VAR_ratingData[8] := true;
        LNK_creditRequest.approvalResult([self],VAR_ratingData);
    }
  s1 -> s1
    { requestClerkApproval(callid,ratingData) /
        VAR_ratingData := ratingData;
        VAR_ratingData[8] := false;
        LNK_creditRequest.approvalResult([self],VAR_ratingData);
    }
  -- requestSupervisorApproval(ratingData: RatingData)
  s1 -> s1
    { requestSupervisorApproval(callid,ratingData) /
        VAR_ratingData := ratingData;
        VAR_ratingData[8] := true;
        LNK_creditRequest.approvalResult([self],VAR_ratingData);
    }
  s1 -> s1
    { requestSupervisorApproval(callid,ratingData) /
        VAR_ratingData := ratingData;
        VAR_ratingData[8] := false;
        LNK_creditRequest.approvalResult([self],VAR_ratingData);
    }
  -- offerToClient(agreement: AgreementData): Boolean
  s1 -> s1
    { offerToClient(callid,agreement) /
```

```
        Tmp: Token[] := callid;
        TheClient: obj;
        TheClient := agreement[0];
        TheClient.offerToClient([self] + Tmp,agreement);
    }
  s1 -> s1
    { offerToClient_return(callid,accepted) /
        LNK_creditRequest.offerToClient_return(callid.tail,accepted);
    }
  -- declineToClient(agreement: AgreementData): Boolean
  s1 -> s1
    { declineToClient(callid,agreement) /
        Tmp: Token[] := callid;
        TheClient: obj;
        TheClient := agreement[0];
        TheClient.declineToClient([self] + Tmp,agreement)
    }
  s1 -> s1
    { declineToClient_return(callid,updateDesired) /
        LNK_creditRequest.declineToClient_return(callid.tail,
          updateDesired);
    }
  -- goodbye(customerData: CustomerData)
  s1 -> s1
    { goodbye(callid,customerData)
    }
  -- abortProcess()
  s1 -> s1
    { abortProcess(callid)
    }

end PortalService
------------------------------------------------------------------------
---------------------------- BALANCE SERVICE ---------------------------
------------------------------------------------------------------------
Class BalanceService is

Signals
  updateBalanceRating(callid,customerData,balancePackage); -- s&r from CR
  clearData(callid,customerData); -- send&receive from CreditRequest
  calculateBalanceRating(callid,customerData); -- s&r from RatingService

Vars
  Priority:int := 3;
  VAR_balanceRating;

State Top = s1

Transitions
  -- updateBalanceRating(customerData: CustomerData,
  --   balances: BalancePackage)
  s1 -> s1
    { updateBalanceRating(callid,customerData,balancePackage) /
        Caller: obj;
        Caller := callid[0];
        Caller.updateBalanceRating_return(callid);
    }
  -- clearData(customerData: CustomerData)
  s1 -> s1
    { clearData(callid,customerData) /
        Caller: obj;
        Caller := callid[0];
        Caller.clearData_return(callid);
    }
  -- calculateBalanceRating(customerData: CustomerData): int
  s1 -> s1
    { calculateBalanceRating(callid,customerData) /
        Caller: obj;
        Caller := callid[0];
        Caller.calculateBalanceRating_return(callid,VAR_balanceRating);
    }

end BalanceService
------------------------------------------------------------------------
---------------------------- SECURITY SERVICE --------------------------
------------------------------------------------------------------------
Class SecurityService is

Signals
  updateSecurityRating(callid,customerData,securityPackage); -- s&r from CR
  clearData(callid,customerData); -- send&receive from CreditRequest
  calculateSecurityRating(callid,customerData); -- s&r from RatingService

Vars
  Priority: int := 3;
  securityRating;

State Top = s1

Transitions
  -- updateSecurityRating(customerData: CustomerData,
  --   securities: SecurityPackage)
  s1 -> s1
    { updateSecurityRating(callid,customerData,securityPackage) /
        Caller: obj;
        Caller := callid[0];
        Caller.updateSecurityRating_return(callid);
    }
  -- clearData(customerData: CustomerData)
  s1 -> s1
```

```
    { clearData(callid,customerData) /
        Caller: obj;
        Caller := callid[0];
        Caller.clearData_return(callid);
    }
  -- calculateSecurityRating(customerData: CustomerData): int
  s1 -> s1
    { calculateSecurityRating(callid,customerData) /
        Caller: obj;
        Caller := callid[0];
        Caller.calculateSecurityRating_return(callid,securityRating);
    }

end SecurityService
-------------------------------------------------------------------------
-------------------------- CALCULATOR SERVICE ---------------------------
-------------------------------------------------------------------------
Class CalculatorService is

Signals
  performRatingCalculation(callid,balanceRating,
    securityRating); -- send&receive from RatingService

Vars
  Priority: int := 3;

State Top = s1

Transitions:
  -- performRatingCalculation(callid: int,balanceRating: int,
  --   securityRating): RatingData
  s1 -> s1
    { performRatingCalculation(callid,balanceRating,securityRating) /
        Caller: obj;
        Caller := callid[0];
        Caller.performRatingCalculation_return(callid,AAA);
    }
  s1 -> s1
    { performRatingCalculation(callid,balanceRating,securityRating) /
        Caller: obj;
        Caller := callid[0];
        Caller.performRatingCalculation_return(callid,BBB);
    }
  s1 -> s1
    { performRatingCalculation(callid,balanceRating,securityRating) /
        Caller: obj;
        Caller := callid[0];
        Caller.performRatingCalculation_return(callid,OTHER);
    }
```

```
end CalculatorService
-------------------------------------------------------------------------
---------------------------- RATING SERVICE -----------------------------
-------------------------------------------------------------------------
Class Rating is

Signals:
   calculateRating(callid,customerData); -- receive from CreditRequest
   calculateBalanceRating_return(callid,result); -- reply BalanceService
   calculateSecurityRating_return(callid,result); -- reply SecurityService
   performRatingCalculation_return(callid,result: Token); -- reply from CR
   --
   tau;

Vars:
  Priority: int := 10;
  RANDOMQUEUE;
  NormalPriority:int := 2;
  Startup: bool := True;
  -- STATIC LNK of Activity TOP LEVEL:
  LNK_ratingService: obj;
  LNK_balanceService: obj;
  LNK_securityService: obj;
  LNK_calculatorService: obj;
  ------------------------------------------------------------------
  ------------------------- Activity Rating ------------------------
  ------------------------------------------------------------------
  -- INITIALIZATION:
  RatingTOK: int := 0;
  RatingIN: bool := false;
  RatingOUT: bool := false;
  RatingERR: bool := false;
  -- ENABLING NODES:
  N1_Rating_DefaultInitialOUT: bool := false;
  --
  N2_Rating_Receive_calculateRatingOUT: bool:= false;
  --
  N3_Rating_fork1OUT: bool := false;
  N3_Rating_fork2OUT: bool := false;
  --
  N4_Rating_SendReceive_calculateBalanceRatingOUT: bool:= false;
  N4_Rating_SendReceive_calculateBalanceRatingSTEP: bool:= false;
  N5_Rating_SendReceive_calculateSecurityRatingOUT: bool:= false;
  N5_Rating_SendReceive_calculateSecurityRatingSTEP: bool:= false;
  N6_Rating_SendReceive_performRatingCalculationOUT: bool:= false;
  N6_Rating_SendReceive_performRatingCalculationSTEP: bool:= false;
  N7_Rating_Reply_calculateRatingOUT: bool:= false;
  -- LOCAL variables:
  VAR_Rating_customerData: Token[];
  VAR_Rating_balanceRating: int;
```

```
  VAR_Rating_securityRating: int;
  VAR_Rating_overallratingData: Token;
  --
  VAR_Rating_Evaluation: Token;
  -- Implicit CALLID:
  N2_Rating_callid; -- receive&reply
  N4_Rating_callid; -- send&receive
  N5_Rating_callid; -- send&receive
  N6_Rating_callid; -- send&receive
  ---------------------------------------------------------------------
  ------------------------- STATE STRUCTURE ---------------------------
  ---------------------------------------------------------------------
State Top = s1

Defers
  calculateRating(callid,customerData),
  tau

Transitions:
  ---------------------------------------------------------------------
  -------------------------- SYSTEM STARTUP ---------------------------
  ---------------------------------------------------------------------
  s1 -> s1
    { - [Startup = true] /
        Startup := false;
        self.tau;
        RatingIN := true
    }
  ---------------------------------------------------------------------
  -------------------------- ACTIVITY RATING --------------------------
  ---------------------------------------------------------------------
  s1 -> s1
    { - [RatingIN = true] /
        RatingIN := false;
        RatingTOK := 1;
        N1_Rating_DefaultInitialOUT := true;
        Priority := NormalPriority;
    }
  -- DEFAULTINITIAL --> <<RECEIVE>> CALCULATERATING:
  s1 -> s1
    { calculateRating(callid,customerData)
        [N1_Rating_DefaultInitialOUT = true] /
        N1_Rating_DefaultInitialOUT := false;
        LNK_ratingService := callid[0];
        N2_Rating_callid := callid;
        VAR_Rating_customerData := customerData;
        VAR_Rating_Evaluation := VAR_Rating_customerData[7];
        N2_Rating_Receive_calculateRatingOUT := true;
    }
  -- <<RECEIVE>> CALCULATERATING --> FORK1,FORK2:
s1 -> s1
  { tau [N2_Rating_Receive_calculateRatingOUT = true] /
        N2_Rating_Receive_calculateRatingOUT := false;
        N3_Rating_fork1OUT := true;
        N3_Rating_fork2OUT := true;
        RatingTOK := RatingTOK + 1; -- for the fork
        self.tau;
  }
-- FORK1 --> <<SEND&...>> CALCULATEBALANCERATING:
s1 -> s1
  { tau [N3_Rating_fork1OUT = true] /
        N3_Rating_fork1OUT := false;
        N4_Rating_callid := [self,4];
        LNK_balanceService.calculateBalanceRating(N4_Rating_callid,
          VAR_Rating_customerData);
        N4_Rating_SendReceive_calculateBalanceRatingSTEP := true;
        self.tau;
  }
-- <<SEND&...>> CALCULATEBALANCERATING -->
--   <<...&RECEIVE>> CALCULATEBALANCERATING:
s1 -> s1
  { calculateBalanceRating_return(callid,result)
      [(N4_Rating_SendReceive_calculateBalanceRatingSTEP = true) and
       (callid = N4_Rating_callid)] /
      N4_Rating_SendReceive_calculateBalanceRatingSTEP := false;
      VAR_Rating_balanceRating := result;
      N4_Rating_SendReceive_calculateBalanceRatingOUT := true;
  }
-- FORK2 --> <<SEND&...>> CALCULATESECURITYRATING:
s1 -> s1
  { tau [N3_Rating_fork2OUT = true] /
        N3_Rating_fork2OUT := false;
        N5_Rating_callid := [self,5];
        LNK_securityService.calculateSecurityRating(N5_Rating_callid,
          VAR_Rating_customerData);
        N5_Rating_SendReceive_calculateSecurityRatingSTEP := true;
        self.tau;
  }
-- <<SEND&...>> CALCULATESECURITYRATING -->
--   <<...&RECEIVE>> CALCULATESECURITYRATING:
s1 -> s1
  { calculateSecurityRating_return(callid,result)
      [(N5_Rating_SendReceive_calculateSecurityRatingSTEP = true) and
       (callid = N5_Rating_callid)] /
      N5_Rating_SendReceive_calculateSecurityRatingSTEP := false;
      VAR_Rating_securityRating := result;
      N5_Rating_SendReceive_calculateSecurityRatingOUT := true;
  }
-- JOIN1,JOIN2 --> <<SEND&...>> PERFORMRATINGCALCULATION:
s1 -> s1
```

```
  { tau [(N4_Rating_SendReceive_calculateBalanceRatingOUT = true) and
         (N5_Rating_SendReceive_calculateSecurityRatingOUT = true)] /
        N4_Rating_SendReceive_calculateBalanceRatingOUT := false;
        N5_Rating_SendReceive_calculateSecurityRatingOUT := false;
        N6_Rating_callid := [self,6];
        RatingTOK := RatingTOK - 1; -- for the join
        LNK_calculatorService.performRatingCalculation(N6_Rating_callid,
          VAR_Rating_balanceRating,VAR_Rating_securityRating);
        N6_Rating_SendReceive_performRatingCalculationSTEP := true;
        self.tau;
  }
-- <<SEND&...>> PERFORMRATINGCALCULATION -->
--   <<...&RECEIVE>> PERFORMRATINGCALCULATION:
s1 -> s1
  { performRatingCalculation_return(callid,result:Token)
      [(N6_Rating_SendReceive_performRatingCalculationSTEP = true) and
       (callid = N6_Rating_callid)] /
      N6_Rating_SendReceive_performRatingCalculationSTEP := false;
      VAR_Rating_overallratingData := result;
      N6_Rating_SendReceive_performRatingCalculationOUT := true;
  }
-- <<SEND&RECEIVE>> PERFORMRATINGCALCULATION -->
--   <<REPLY>> CALCULATERATING:
s1 -> s1
  { tau [N6_Rating_SendReceive_performRatingCalculationOUT = true] /
        N6_Rating_SendReceive_performRatingCalculationOUT := false;
        LNK_ratingService.calculateRating_return(N2_Rating_callid,
          VAR_Rating_overallratingData);
        N7_Rating_Reply_calculateRatingOUT := true;
        self.tau;
  }
-- <<REPLY>> CALCULATERATING --> ACTIVITYFINAL:
s1 -> s1
  { tau [N7_Rating_Reply_calculateRatingOUT = true] /
        N7_Rating_Reply_calculateRatingOUT := false;
        RatingTOK := 0;
        VAR_Rating_customerData := null;
        VAR_Rating_balanceRating := 0;
        VAR_Rating_securityRating := 0;
        VAR_Rating_overallratingData := null;
        N2_Rating_callid := null;
        N4_Rating_callid := null;
        N5_Rating_callid := null;
        N6_Rating_callid := null;
        RatingOUT := true;
        -- RATING is a permanent service. When completed, it restarts:
        RatingOUT := false;
        RatingIN := true;
        N1_Rating_DefaultInitialOUT := true;
      self.tau;
    }

end Rating
------------------------------------------------------------------------
---------------------- SYSTEM CONFIGURATION ----------------------------
------------------------------------------------------------------------
Objects:
  name, passwd, amount, creditype, monthlyinstallment, rate, securities,
    balance: Token;
  ratingData, AAA, BBB, OTHER: Token;
  --
  TheClient: Client
    (Priority -> 2, LNK_portal -> ThePortal);
  --
  ThePortal: PortalService
    (Priority -> 3, LNK_creditRequest -> TheCredit);
  TheCalculator: CalculatorService
    (Priority -> 3);
  TheCustomerManagement: CustomerManagementService
    (Priority -> 3);
  TheCreditManagement: CreditManagementService
    (Priority -> 3);
  TheBalanceService: BalanceService
    (Priority -> 3);
  TheSecurityService: SecurityService
    (Priority -> 3);
  --
  TheRating: Rating
    (Priority -> 4, NormalPriority -> 2,
       LNK_balanceService -> TheBalanceService,
       LNK_securityService -> TheSecurityService,
       LNK_ratingService -> TheCredit,
       LNK_calculatorService -> TheCalculator);
  --
  TheCredit: CreditRequest
    (Priority -> 5, NormalPriority -> 2,
       LNK_customerManagementService -> TheCustomerManagement,
       LNK_portalService -> ThePortal,
       LNK_creditManagementService -> TheCreditManagement,
       LNK_balanceService -> TheBalanceService,
       LNK_securityService -> TheSecurityService,
       LNK_ratingService -> TheRating);
------------------------------------------------------------------------
--------------------------- States: 39530 ------------------------------
------------------------------------------------------------------------

------------------------------------------------------------------------
-------------------------- Version: 10-02-2010 -------------------------
------------------------------------------------------------------------
```

```
------------------------------------------------------------------------
----------------------- SYSTEM ABSTRACTIONS ----------------------------
------------------------------------------------------------------------
Abstractions {
  State  TheCredit.N1_Initialize_DefaultInitialOUT = true ->
           accepting_requests(initialize)
  State  TheCredit.MainTOK > 0 -> accepting_requests(cancel)
  State  TheCredit.MainTOK > 0 -> activity_in_progress(Main)
  State  TheRating.VAR_Rating_Evaluation = $1 -> rating_evaluation($1)
  --
  Action $1:$2.$3($*) and $1 /= $2 -> receive($1,$3)
  Action $1:accept($2) and $2 /= tau -> send($1,$2)
  Action $1:lostevent($2,$*) -> lostevent($1,$2,$*)
  Action $1.Runtime_Error -> Runtime_Error($1)
  --
  Action assign(CreditRequestIN,*,true) -> start(CreditRequest)
  Action assign(CreditRequestOUT,*,true) -> finish(CreditRequest)
  Action assign(CreditRequestERR,*,true) -> error(CreditRequest)
  Action TheCredit:ThePortal.createNewCreditRequest($*) ->
           accept(createNewCreditRequest)
  Action TheCredit:accept(declineToClient) -> response(declineToClient)
  Action TheCredit:accept(offerToClient) -> response(offerToClient)
  Action TheCredit:ThePortal.cancel($*) -> receive(cancel)
  Action TheCredit:accept(calculateRating) -> request(calculateRating)
  Action TheCredit:accept(clearData) -> send(clearData)
  Action TheCredit:accept(removeData) -> send(removeData)
  --
  Action assign(RatingIN,*,true) -> start(Rating)
  Action assign(RatingOUT,*,true) -> finish(Rating)
  Action assign(RatingERR,*,true) -> error(Rating)
  --
  Action assign(InitializeIN,*,true) -> start(Initialize)
  Action assign(InitializeOUT,*,true) -> finish(Initialize)
  Action assign(InitializeERR,*,true) -> error(Initialize)
  --
  Action assign(MainIN,*,true) -> start(Main)
  Action assign(MainOUT,*,true) -> finish(Main)
  Action assign(MainERR,*,true) -> error(Main)
  --
  Action assign(FinalizeIN,*,true) -> start(Finalize)
  Action assign(FinalizeOUT,*,true) -> finish(Finalize)
  Action assign(FinalizeERR,*,true) -> error(Finalize)
  --
  Action assign(CancelIN,*,true) -> start(Cancel)
  Action assign(CancelOUT,*,true) -> finish(Cancel)
  Action assign(CancelERR,*,true) -> error(Cancel)
  --
  Action assign(MainFaultIN,*,true) -> start(MainFault)
  Action assign(MainFaultOUT,*,true) -> finish(MainFault)
  Action assign(MainFaultERR,*,true) -> error(MainFault)
  --
  Action assign(CreationIN,*,true) -> start(Creation)
  Action assign(CreationOUT,*,true) -> finish(Creation)
  Action assign(CreationERR,*,true) -> error(Creation)
  --
  Action assign(InitializationIN,*,true) -> start(Initialization)
  Action assign(InitializationOUT,*,true) -> finish(Initialization)
  Action assign(InitializationERR,*,true) -> error(Initialization)
  --
  Action assign(CompensateCreditManagementInitializationIN,*,true) ->
    start(CompensateCreditManagementInitialization)
  Action assign(CompensateCreditManagementInitializationOUT,*,true) ->
    finish(CompensateCreditManagementInitialization)
  Action assign(CompensateCreditManagementInitializationERR,*,true) ->
    error(CompensateCreditManagementInitialization)
  --
  Action assign(HandleBalanceSecurityDataIN,*,true) ->
    start(HandleBalanceSecurityData)
  Action assign(HandleBalanceSecurityDataOUT,*,true) ->
    finish(HandleBalanceSecurityData)
  Action assign(HandleBalanceSecurityDataERR,*,true) ->
    error(HandleBalanceSecurityData)
  --
  Action assign(withoutname1IN,*,true) -> start(withoutname1)
  Action assign(withoutname1OUT,*,true) -> finish(withoutname1)
  Action assign(withoutname1ERR,*,true) -> error(withoutname1)
  --
  Action assign(withoutname2IN,*,true) -> start(withoutname2)
  Action assign(withoutname2OUT,*,true) -> finish(withoutname2)
  Action assign(withoutname2ERR,*,true) -> error(withoutname2)
  --
  Action assign(RatingCalculationIN,*,true) -> start(RatingCalculation)
  Action assign(RatingCalculationOUT,*,true) -> finish(RatingCalculation)
  Action assign(RatingCalculationERR,*,true) -> error(RatingCalculation)
  --
  Action assign(DecisionIN,*,true) -> start(Decision)
  Action assign(DecisionOUT,*,true) -> finish(Decision)
  Action assign(DecisionERR,*,true) -> error(Decision)
  --
  Action assign(ApprovalIN,*,true) -> start(Approval)
  Action assign(ApprovalOUT,*,true) -> finish(Approval)
  Action assign(ApprovalERR,*,true) -> error(Approval)
  --
  Action assign(AcceptIN,*,true) -> start(Accept)
  Action assign(AcceptOUT,*,true) -> finish(Accept)
  Action assign(AcceptERR,*,true) -> error(Accept)
  --
  Action assign(DeclineIN,*,true) -> start(Decline)
  Action assign(OUT,*,true) -> finish(Decline)
  Action assign(DeclineERR,*,true) -> error(Decline)
```

```
  }
--------------------------------------------------------------------------
------------------------------ TYPES -------------------------------------
--------------------------------------------------------------------------
-- RatingData
-- 0 (id: int,
-- 1  customerId: int,
-- 2  creditId: int,
-- 3  securityId: int,
-- 4  creditAmount: float,
-- 5  securityResult: String,
-- 6  balanceResult: String,
-- 7  result: String,
-- 8  manualAcceptance: bool
--   )
--
-- SecurityData
-- 0 (id: int,
-- 1  customerId: int,
-- 2  creditId: int,
-- 3  subject: int,
-- 4  nominalAmount: float,
-- 5  calculatedAmount: String
--   )
--
-- BalanceData
-- 0 (id: int,
-- 1  customerId: int,
-- 2  loadedAsXbrFile: bool := false,
-- 3  assets: float,
-- 4  floatingAssets: float,
-- 5  totalAssets: float,
-- 6  equity: float,
-- 7  committedAssets: float,
-- 8  totalLiabilities: float,
-- 9  income: float,
--10  expenditure: float,
--11  businessResult: float,
--12  date: String
--   )
--
-- CreditType (motor, house, security, general, other)
--
-- SecurityType (land, motor, liquid, other)
--
-- UserData
-- 0 (id: int,
-- 1  name: String,
-- 2  password: String
--   )
--
-- CreditData
-- 0 (id: int,
-- 1  customerId: int,
-- 2  creditAmount: float,
-- 3  creditType: int,
-- 4  monthlyInstallment: float,
-- 5  interestRate: float,
-- 6  securityList: String,
-- 7  balanceList: String
--   )
--
-- AgreementData
-- 0 (id: int,
-- 1  customerId: int,
-- 2  securityId: int,
-- 3  ratingId: int,
-- 4  createOffer: bool := false,
-- 5  waitingForCustomer: bool := false,
-- 6  acceptOffer: bool := false,
-- 7  pdfDocument: String
--   )
--
-- CustomerData
-- 0 (id: int,
-- 1  loginName: String,
-- 2  firstName: String,
-- 3  lastName: String
--   )
```