Modeling and Simulation of Selected Operational IT Risks in the Banking Sector (Extended Version)

Christoph Brandt[1]
Technical University of Berlin, Germany email: cbrandt@cs.tu-berlin.de

Francesco Santini[2], Natallia Kokash and Farhad Arbab
Centrum Wiskunde & Informatica, Netherlands
email: F.Santini@cwi.nl, Natallia.Kokash@cwi.nl, Farhad.Arbab@cwi.nl

27 April 2013

KEYWORDS
Operational risks, hybrid simulation, control theory, banking sector

ABSTRACT

International banks need to estimate their operational risks due to external regulations. Based on their estimations they need to provide private capital to cover potential losses caused by these risks. Therefore, operational risks need to be properly measured and managed in order to reduce the required private capital. In this paper we discuss operational risks related to a typical banking business process that is enabled by an IT landscape. We present how risks related to the operational behavior of the IT landscape can be simulated. The simulation results help to estimate risk measures like the expected loss, the value-at-risk and the expected shortfall. We further sketch how control theory can be used to actively manage the dynamic reconfiguration of a service landscape, in order to minimize modeled operational risks. First experimental simulation results illustrate our approach.

Introduction

We present a new approach to modeling and simulating operational risks that shows some potential to successfully address open issues known from today's best-practices. The research question is how to model and simulate operational risks of real-world financial organizations using organizational models (Brandt and Hermann (2013)). The main contribution of our work consists of two parts: The first part is about simulating operational risks using an approach that is bottom up and top down at the same time. The second part is about dynamic reconfiguration of service landscapes using control theory, which helps to actively optimize modeled operational risks. The rest of this paper is organized as follows: We, firstly, reflect on the notion of operational risks, introduce a real-world scenario and present the methods and tools that we use. Secondly, we show our first experimental results of an assumed IT landscape as well as a preliminary risk assessment based on these results, and potential next steps of our study. Finally, we discuss selected related work.

Operational Risks

According to the Basel Committee of Banking Supervision *"operational risk is defined as the risk of loss resulting from inadequate or failed internal processes, people and systems or from external events. This definition includes legal risk, but excludes strategic and reputational risk"* (on Banking Supervision (2011b;a; 2006; 2012)). Operational risks complete the risk portfolio of a bank, which also encompasses credit risks and market risks. Operational risks are largely firm-specific non-systematic risks (Tchernobai (2006)).
In the context of this paper, we focus on the technological risk related to a business process that is enabled by an IT landscape. We measure operational risks by the help of the expected loss, the value-at-risk and the expected shortfall, because these measures nicely integrate with already existing risk frameworks in financial organizations.

A real-world Scenario in Finance Industry

The concrete scenario we consider here was inspired by a study of real-world requirements at Credit Suisse. We decided to go for a business process that is about buying shares over the internet by the help of an e-banking system. In this scenario, people select shares they want to buy, put them into a basket, and finally pay for them using electronic means like credit cards. Once in a while, an account statement is sent to clients by email summarizing all their past transactions.
In order to be able to discuss operational risks related to IT systems, we decided to model a service landscape, which implements and enables this business process. The model is shown in Fig. 1. It comes as a Model-

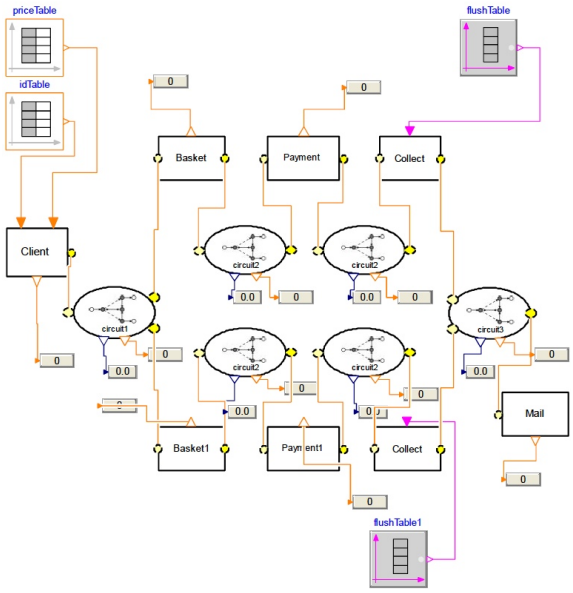

Figure 1: Case 1

ica model, running in a Dymola 2013 system. We call this model "Case 1."

In this model, a stream of client orders flows into the system. Each order has a certain business value in euro cents, comes at a specific point in time, and adds up to a concrete basket until that basket is complete.

The input stream in model "Case 1" is replicated (circuit 1) to be processed by an upper (basket, payment, collect) and a lower branch (basket1, payment1, collect1). Because if a request fails in either one of the branches the other one always provides a potential back up. The streams out of the two branches are merged (circuit 3), collected and forwarded from time to time to a mail service, which sends out account statements by email.

The basket services handle the book-keeping of all incoming requests until their corresponding baskets are complete. Once a basket is complete, the whole basket is sent to the payment service. Both services can fail. This behavior is realized by failure distributions. Each of these services has its own failure distribution. The collect services collect all successfully paid baskets and flush them out to the mail service based on a given time table (flushTable, flushTable1). In addition to that, the payment services and the mail services use different delay distributions. The routing between the different services is realized by circuits that internally implement different Reo connectors (Arbab (2004)), which are ideally suited to support the exogenous coordination of services in an IT landscape. These circuits contain buffers, which can run full. In this case, incoming requests are dropped, and, therefore, lost.

Methods and Tools

In this section we introduce the methods and tools that we use and give a short explanation of how we apply them.

Methods: *Reo* (Arbab (2004)) represents a paradigm for composition of distributed software components and services based on the notion of mobile channels. Reo enforces an exogenous channel-based coordination model that defines how designers can build complex coordinators, called connectors, out of simpler ones. Application designers can use Reo as a "glue code" language for compositional construction of connectors that orchestrate the cooperative behavior of instances of components or services in a component-based system or a service-oriented application.[3] Reo supports loose coupling among components and services, distribution of heterogeneous components, exogenous coordination, compositional construction (which nicely matches with the object oriented modeling approach in Modelica), an arbitrary mix of synchrony and asynchrony, user-defined primitives, dynamic reconfiguration, a formal graphical syntax analogous to electronic circuit diagrams, a formal semantics based on coinductive calculus of flow and (alternatively) on constraint automata, and specification and verification methods using programming logic. We use Reo to model the exogenous coordination between the services in our IT landscape. *Hybrid simulations* (Saouma and Sivaselvan (2008)) encompass discrete and continuous simulation techniques at the same time. Discrete simulations are event driven and state based. Continuous simulations represent continuous processes and are usually encoded using differential equation systems. We run hybrid simulations to implement the behavior of constraint automata, which define the semantics of Reo connectors, and we plan to use differential equation systems in order to codify the semantics of control elements. *Control theory* (Bubnicki (2005)) is a theory that deals with influencing the behavior of dynamical systems. The objective of a control theory is to determine corrective actions that lead to system stability. That means, that the system will stabilize at some point and not oscillate. Differential equations describe the input and outputs of a continuous control systems. We rely on control theory to manage the dynamic reconfiguration of Reo circuits.

Tools: Dymola[4] is a simulation engine that supports the simulation of Modelica models. Modelica[5] is a special purpose language for the specification of hybrid simulation systems. Modelica realizes hierarchical model composition, encompasses libraries of truly reusable components and connectors as well as composite non causal connections. Its modeling methodology emphasizes object orientation and equations. Dymola supports graphical composition of Modelica models, and fast simulation with symbolic pre-processing of these models. We use the Dymola/Modelica bundle to model IT landscapes in

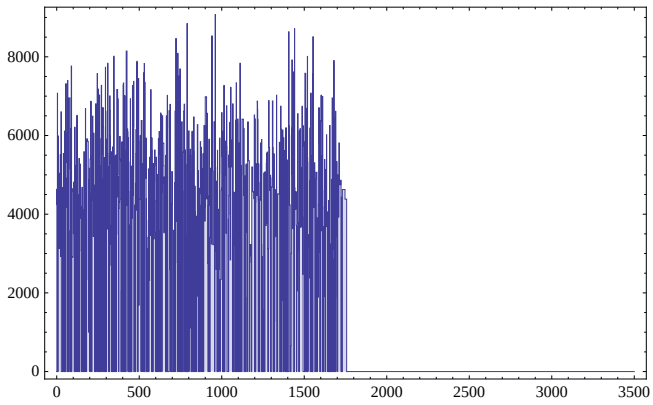

Figure 2: Input Request Stream (x:[s], y:[Euro])

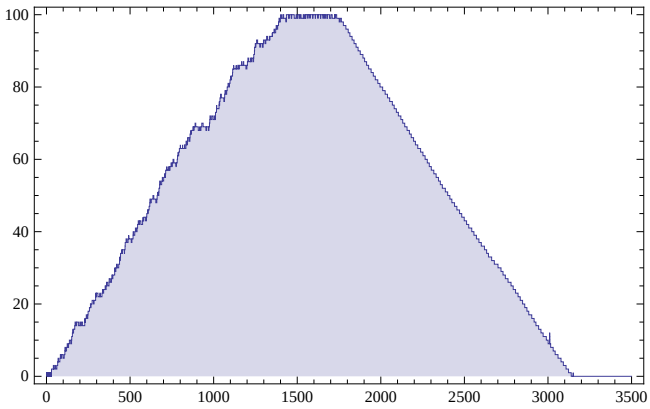

Figure 3: Used FIFO Capacity (x:[s], y:[%])

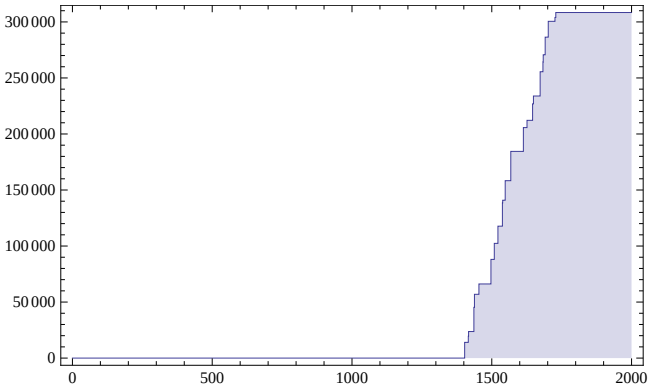

Figure 4: Accumulated FIFO Losses (x:[s], y:[Euro])

an object-oriented and graphical way and we use it to run hybrid simulations. Matlab[6] is a numerical computing environment. Its add-on Simulink[7] provides an extensive library of elements for control theory, which are ready to use off the shelf. We use Matlab/Simulink to run control theory models, which actively manage the dynamic reconfiguration of IT landscapes coordinated by Reo circuits in order to optimize operational risks. Matlab/Simulink integrates nicely with Dymola/Modelica as it is possible to run Modelica models, which have been compiled using Dymola, inside a Matlab/Simulink model. Mathematica[8] is a concrete computer algebra system. It comes with an extensive built-in functionality for statistics. We use the symbolic and numerical methods implemented in Mathematica for the risk analysis in our study.

Experimental Results

In this section we present selected experimental results to further illustrate our approach. These results are currently not based on empirical data, but on some random numbers, which we generated based on the insights gained from the analysis of the requirements derived from the real-world scenario at Credit Suisse[9]. The assumptions are strongly simplified in order to comply with disclosure agreements.

We assess the operational risks related to the IT landscape represented by model "Case 1" by simulating its behavior. Therefore, a request stream of incoming orders from different clients is constructed and pumped into the system. We assume that the value of orders is normally distributed, that two to five orders form a basket and that the interval between incoming orders can be represented as a Poisson distribution (Faisst (2003), p.8, Schäl (2003), p.14, Giacometti et al. (year unkown)).

Fig. 2 shows the incoming business value of each request at a certain point in time during the simulation run. Fig. 3 presents the percentage of the most used up first-in-first-out queue in one of the circuits of model "Case 1". The moment one of the queues is full, which means that the used capacity of one of the fifos is at 100% as it can be seen in the figure, the system starts loosing business requests. We analyze these losses from the point of view of operational risks later in this paper. Their accumulated business value in euro cent is shown in Fig. 4.

In our system, losses do not occur only due to full queues, but also because services fail to operate successfully. This failure can have internal or external reasons. An internal reason may be that a service is defective, an external reason may be that some needed external services do not provide expected results. In the scope of our study we realized service failures as simple failure probabilities. Fig. 5 shows the accumulated business value that was lost during the simulation run due to failures of different services in model "Case 1." As it can be seen, service failures happen from the very beginning, whereas fifo losses start happening once one of the queues is full.

We will see in the following that these different types of losses result in different shapes of loss distributions that are counterintuitive, and which contradict today's assumptions about the shape of loss distributions of operational risks.

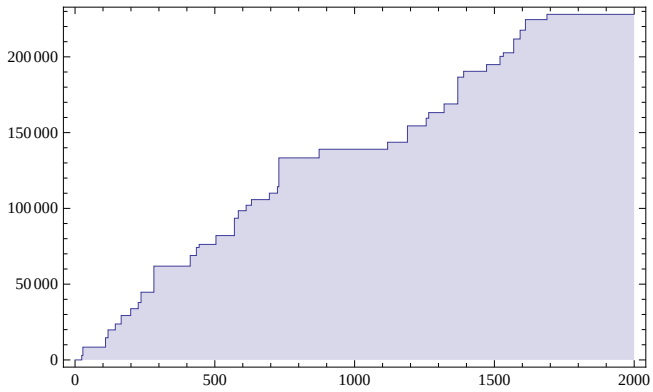

Figure 5: Accumulated Service Losses (x:[s], y:[Euro])

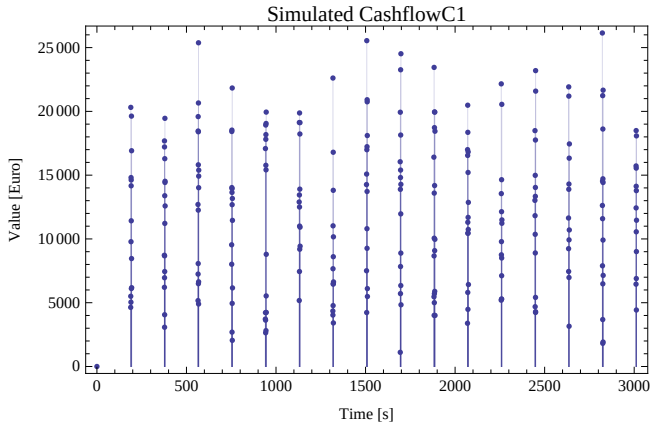

Figure 6: Simulated Cash Flow (x:[s], y:[Euro])

In our model "Case 1" paid baskets are collected in the collect services. Afterwards, account statements are sent out on a regular basis using a time table. Fig. 6 shows the total business value related to these account statements that are sent out at certain points in time.
We defined the accumulated business value as the obtained cash flow of successfully executed business processes.
In line with today's practice for operational risks we present our finding using value and frequency distributions. In Fig. 7 the copula of the value/frequency distribution of the input requests is presented, which are based on generated random numbers. Here, this copula looks as if it is built out of two independent distributions, which is, in fact, in line with the setting of our simulation run, and, therefore, not surprising, the outcoming results correspond to the inserted variables.
Assuming now independent value and frequency distributions, estimating their parameters and constructing a copula leads to the result presented in Fig. 8.
In the given case, we used a normal distribution to approximate the simulated value distribution and a Poisson distribution to approximate the simulated frequency distribution. Both theoretical distributions fit the simulated distributions.

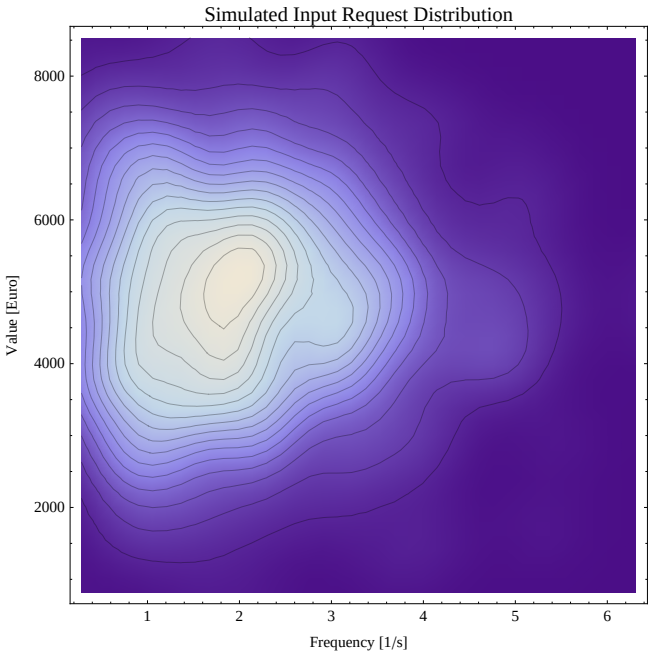

Figure 7: Simulated Input Requests (x:[1/s], y:[Euro])

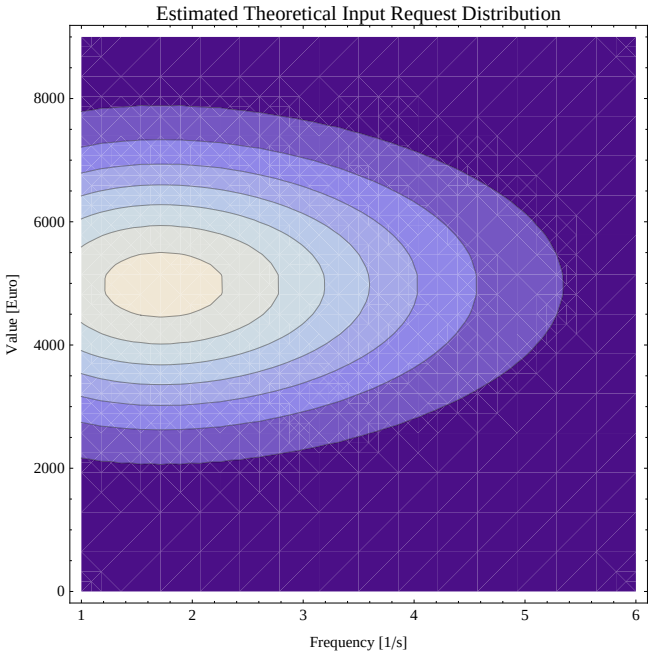

Figure 8: Estimated Input Requests (x:[1/s], y:[Euro])

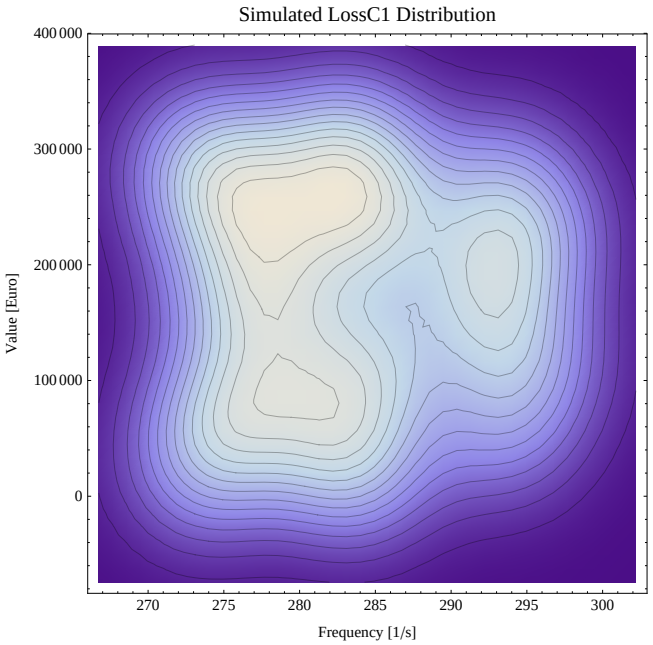

Figure 9: Simulated FIFO Losses (x:[1/s], y:[Euro])

We see a completely different situation in Fig. 9. Here, the copula of simulated losses due to full queues is shown.

The picture indicates that the underlying frequency and value distributions are not independent.

We can see two centers of high value and low value losses at a low frequency and one center of high value losses at a high frequency. This experimental result supports the statement made in Tchernobai (2006) that *"unlike market risk and perhaps credit risk, the [operational] risk factors are largely internal to the bank."* What we see is the structural impact a concrete IT landscape has regarding the observable loss behavior.

In Fig. 10 we present the estimated copula based on the obtained simulation data. It is based on the assumptions of independent value and frequency distributions. As suggested in (Tchernobai (2006)) the value distribution is approximated using an alpha-stable distribution, the frequency distribution is built up using a Poisson distribution. However, as can be seen, the assumptions do not hold. The shape of the estimated copula is significantly different from the shape of the simulated copula. This shape mismatch indicates that top down measurement approaches do not always work well. A much better match between the simulated and the estimated loss distributions related to service failures can be seen in the following. In Fig. 11 the simulated service losses are shown.

In Fig. 12 the estimated theoretical copula is presented based on the same assumptions as before, in Fig. 10. The potential reason for this match is that the overall structure of the IT landscape has much less influence on independent loss events in different services than rout-

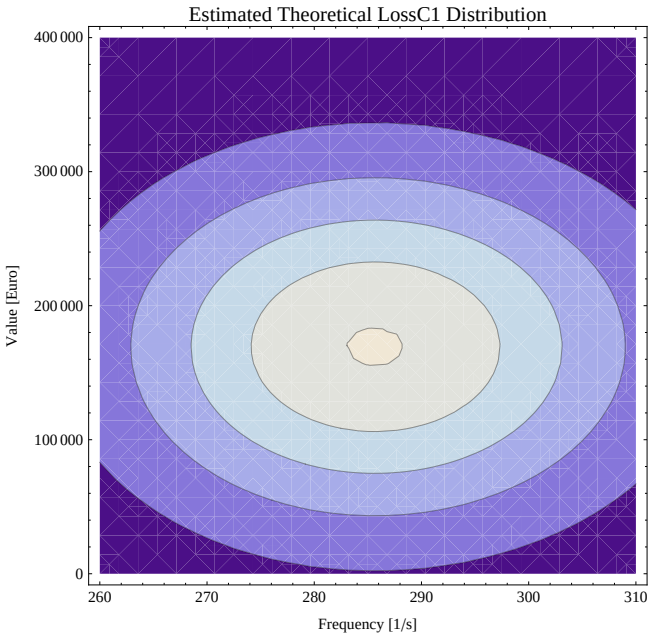

Figure 10: Estimated FIFO Losses (x:[1/s], y:[Euro])

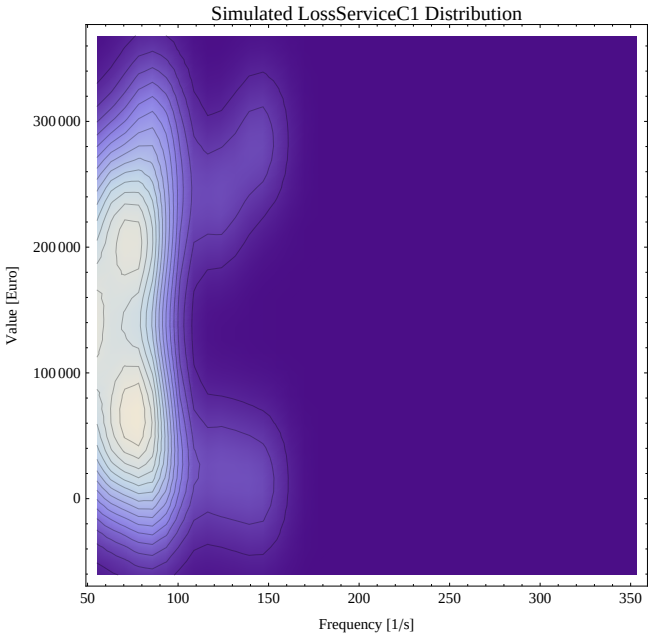

Figure 11: Simulated Service Losses (x:[1/s], y:[Euro])

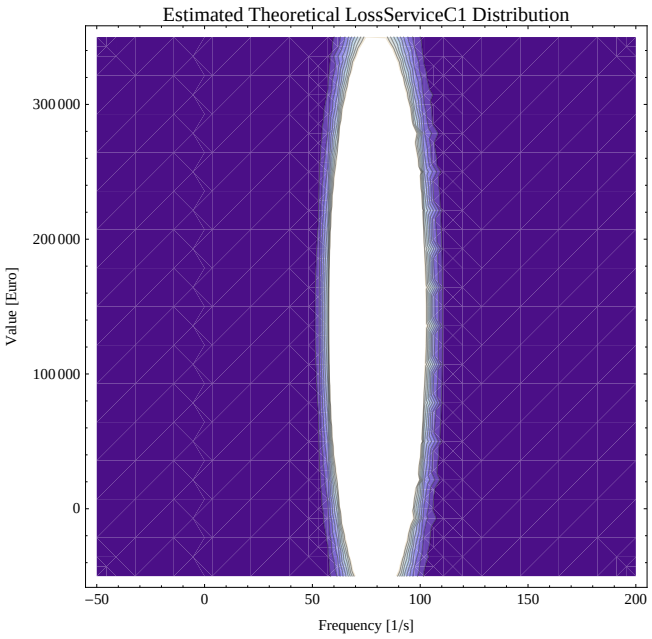

Figure 12: Estimated Service Losses (x:[1/s], y:[Euro])

ing decisions of requests have when losses in queues are analyzed.

However, what we can see is that the value of baskets is no longer normally distributed. There are two centers visible in Fig. 11, and there are apparently two different loss distributions operating, one that causes many losses, and one that cause only few losses. This observation is in line with the settings of our simulation model, the outcoming results correspond to the inserted variables. However, our settings should make sense because, in practice, the individual loss behavior of a specific service application should be specific to that service or application.

Fig. 13 finally shows the simulated copula of the successfully obtained cash flow in the end. We can see four centers. A low frequency/low value, a low frequency/high value, a high frequency/high value and a high frequency/low value centers.

As in previous cases, this phenomenon is caused by the specific qualities of the IT landscape in model "Case 1." Here, again, the estimated theoretical copula, in Fig. 14, oversimplifies the situation. Therefore, the often advertised top down approach recommended to estimate distributions for operational risks does not successfully work here.

Experimental Risk Assessment

In the literature (Tchernobai (2006), Daldrup (2005)) the loss distribution is used to estimate the expected loss, the value-at-risk and the expected shortfall. Whereas the expected loss can be priced into the products and services, the value-at-risk is used to de-

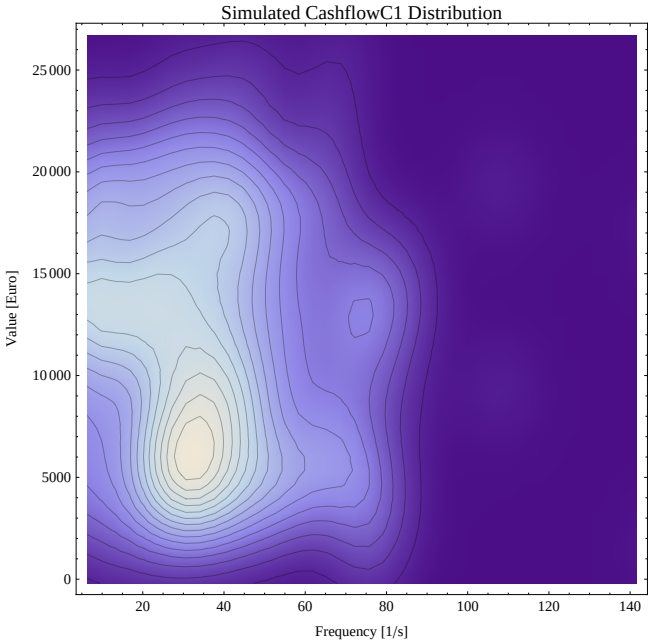

Figure 13: Simulated Cash Flow (x:[1/s], y:[Euro])

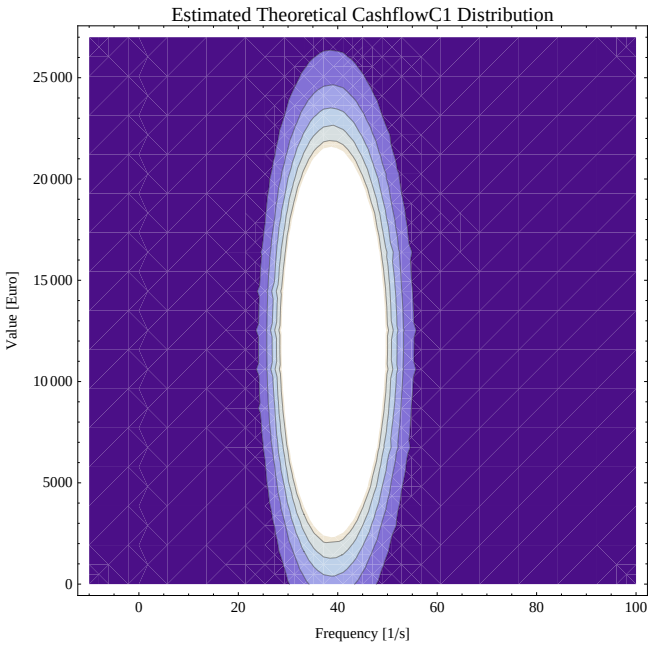

Figure 14: Estimated Cash Flow (x:[1/s], y:[Euro])

termine the needed capital to cover potential, but unforeseen losses. In addition to that, the expected shortfall can be used to optimize a portfolio of operational risks. In the following, we suggest doing this using control theory, whereby a controller governs the automatic re-configuration of Reo circuits that exogenously coordinates services in our assumed IT landscape.

In Fig. 15 an alpha-stable distribution is shown describing the loss behavior of the simulated system. Its parameters have been estimated using bootstrap techniques applied to the simulated data. To take an alpha-stable distribution in order to model operational risks was suggested by (Tchernobai (2006)). Here, the distribution fits the data generated by the simulation and was accepted by all goodness-of-fit tests we checked. From left to right vertical lines represent the expected loss, the value-at-risk and the expected shortfall for the simulated period and for a confidence level of 0.95 on a per business request basis. According to Tchernobai (Tchernobai (2006)) different confidence levels lead to different capital requirements. In order to keep our underlying (numerical) simulation issues simple, we decided to calculate the risk measures for the purpose of demonstration in the scope of this paper for a confidence level of 0.95. However, current regulations (on Banking Supervision (2006) p. 151) require a confidence level of 0.99 or 0.999.

In principle, the same can be done for the cash flow in a symmetric way. The theoretical distributions here are still under investigation and left for future work.

However, the loss distribution covers all losses that occur during the simulation, that is, losses of services and losses because of full queues. As we have seen in earlier figures, some loss distributions are impacted by the structural qualities of the simulated system, whereas others are not. Therefore, the fit of the alpha-stable distribution describing the loss behavior of a system in the given case cannot yet be generalized as it was suggested in Tchernobai (2006).

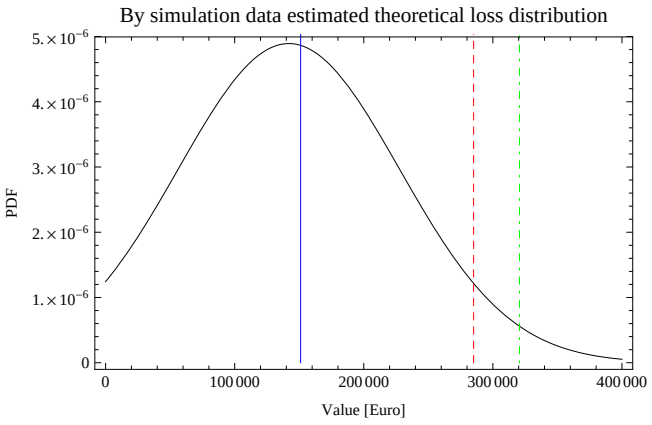

Figure 15: Risk Measurements for Losses (x:[Euro], y:[PDF])

Therefore, the approach we apply is at the same time bottom up and top down. It is bottom up as it generates simulation data of the potential future behavior of a system, which can be used in exchange of past loss data. It is top down as it estimates parameters of theoretical distributions based on the generated simulation data, which are selected based on a priori assumptions. The advantage is clear. Past data do not necessarily reflect the future loss behavior of a system because IT landscapes constantly change. Simulation provides a look-ahead, which fixes this problem. Secondly, today's approaches to estimating operational risks are built on top of data pools. These pools no longer reflect structural information needed to assess operational risks. Here, simulation can help as it is not (necessarily) built up on whole organizations, but on organizational elements, which may appear in different contexts and are more stable than organizations as such. Therefore, we suggest to focus on organizational building blocks, setting them up each to represent a concrete organizational setting, and simulating them in order to derive the necessary data for risk assessment.

Next Steps

The next steps in our study will focusses on the simulation of the model "Case 2". Here, we assume that incoming orders are not replicated and simultaneously processed by an upper and a lower branch, but arbitrarily distributed to the available services in order to maximize the overall capacity of requests the IT landscape can process. Fig. 16 shows the Modelica model of "Case 2" in the Dymola environment.

The main differences compared to model "Case 1" in Fig. 1 are the different Reo circuits that coordinate the used services.

Our assumption is that model "Case 1" is better in terms of operational risks when few requests of high value need to be processed, whereas model "Case 2" excels when many requests of low value need to be handled. The corresponding simulations are left for future work.

Assuming that the nature of the incoming requests may change over time, as for instance, typical business situations at day and at night differ, model "Case 3" in Fig. 17 combines both earlier models. It contains a switch that either selects model "Case 1" or model "Case 2" for processing the incoming requests. From a theoretical point of view, the switch enables the dynamic reconfiguration of the Reo circuits in the given IT landscape by either selecting the first or the second configuration. We plan to control the switch by setting variables of the switch to certain values through a control model. We assume that the control model can at the same time read certain variables indicating the situation of operational risks from model "Case 1" and model "Case 2", respectively.

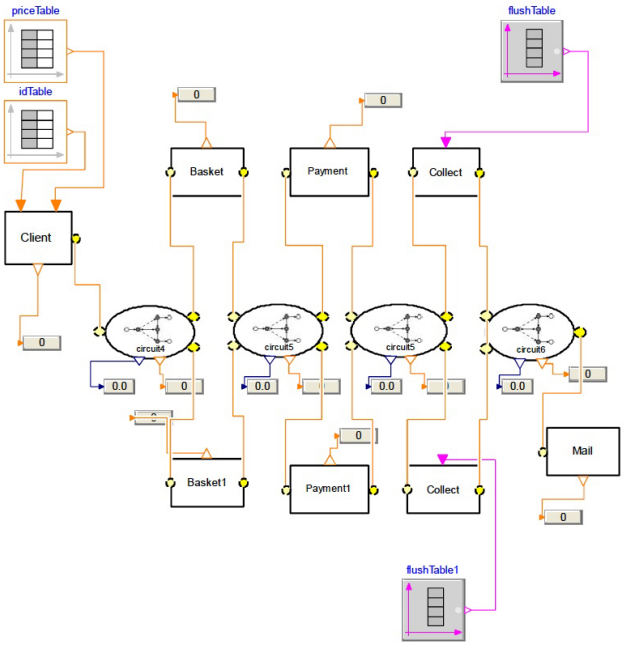

Figure 16: Case 2

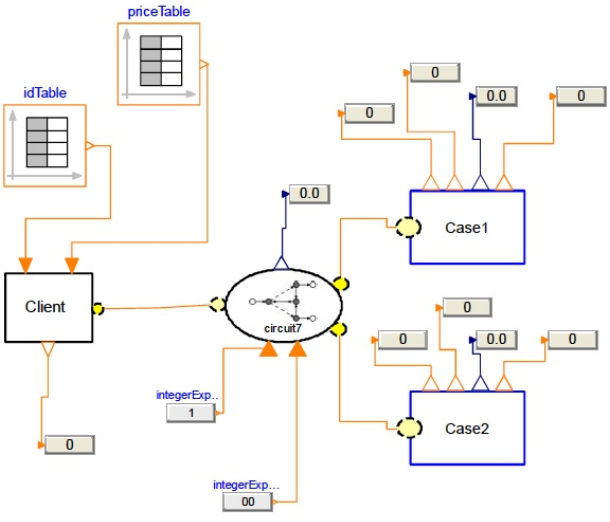

Figure 17: Case 3

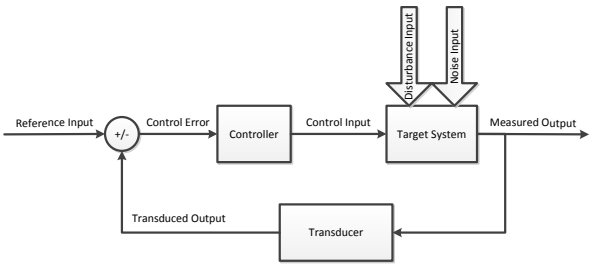

Figure 18: Feedback System

Subsequently, we wrapped model "Case 3" in the Matlab/Simulink environment and defined input and output ports declared in the Dymola/Modelica setting. This enables us to use "Case 3" as a black-box in this environment. Currently, this Matlab/Simulink model just reads generated random data created using Mathematica and writes simulation data into result files. In the future we plan to use the Matlab/Simulink libraries to develop the control model which then either decides to switch to the first or the second Reo configuration. To the best of our knowledge our suggested approach of using a control model in order to switch between Reo configurations of a service landscape in order to optimize the modeled operational risks is new and has not been published thus far. It requires further investigation to determine how to best capture the real operational risks by the help of the modeled operational risks in order to finally optimize the real operational risks.

Fig. 18 presents the general nature of a control model. The target system represents model "Case 3." So, the control model still needs to provide the controller and the transducer. The target system can be disturbed by external noise or stochastic input data. The transducer transforms the measured output from the target system into a transduced output, which is compared with a reference value that can stand for an accepted level of operational risks. The delta or control error is then used by the controller to provide the necessary control input for the target system in order to reduce the control error.

In the appendix of this paper we show all the Modelica code we used in our experiments.

Related Work in Operational Risk Analysis

Three PhD theses have served as a constant source of inspiration during our work on operational risks. The first one is by Anna S. Tchernobai (Tchernobai (2006)). It presents contributions to modeling of operational risks in banks and comes up with an exceptional in-depth statistical treatment of available loss data. The second one is by Anja Hechenblaikner (Hechenblaikner and zu Selhausen (2006)). It presents contributions related to operational risks in banks and provides an outstand-

ing methodological analysis of how to measure operational risks from qualitative and quantitative points of view. The third one is by Britta Kunze who did an outstanding job of analyzing the regulatory sources of operational risks (Kunze and Poddig (2007)). Finally, in his excellent paper, Andre Daldrup (Daldrup (2005)), discusses with impressive clarity different approaches to risk measurement, their strengths, weaknesses and potential uses.

Related Work in Enterprise Architectures

A discussion about the regulatory perspective in enterprise architectures can be found in van Bommel et al. (2007). Here, the question of implementing regulatory requirements into an enterprise architecture or enforcing them using flexible business rules shows up. By anchoring regulatory requirements into an assumed control model we serve both views. The control model is an architectural choice, its parameterization, however, provides room for flexibility in the sense of dynamic business rules.
From the point of view of setting up an Information System Security Risk Management (ISSRM) system, Nicolas Mayer provided a reconstructed domain meta-model that helps to capture all potential risks covered by current industrial standards Nurcan et al. (2010), Mayer (2009). In contrast to that, we covered just selected risks in an assumed IT landscape in order to demonstrate the potential advantages of our simulation approach.

Results and Future Work

Reviewing our main research questions, we were able to provide contributions in the following areas: First, we present an approach to simulating operational risks using hybrid simulation techniques. Our simulated data enabled us to look ahead instead of looking back. Further, the simulation preserved the structural information of the system in the simulated distributions. The simulated data can be used to estimate parameters of given theoretical distributions. Second, we combine control models with Reo models to enact dynamic reconfiguration of service landscapes with the aim to minimize the overall operational risks over time in an actively managed way. In the appendix of this paper we show all the Modelica code we used in our experiments.
Potential future work entails extending the given scenario by external service providers that offer additional capacity to the system landscape. In this context, market prices for services show up. The handling of shared resources like services that run on shared servers, needs to be studied and the competitiveness of "Case 3" in combination with a control model needs to be validated versus "Case 1" and "Case 2". Likewise, the effectiveness and efficiency of our approach must be compared with today's best practices. In addition to that, it appears useful to check if the presented approach is appropriate to fulfill the AMA criteria, and to check which impulses for the daily business of a bank could be generated using it. It may also be interesting to look deeper into the question of what has to be done in order to integrate our approach in the risk bearing ability in Pillar II of Basel 2, and to think about using the results to support the construction of computational methods, as well as the design of non-deceivable risk regulations, that can be applied by regulation authorities.

Acknowledgements

We like to thank the anonymous reviewers for their helpful comments and Prof Dr Thomas Engel of the SnT at the University of Luxembourg (`http://wwwen.uni.lu/snt`) for his constructive feedback about the obtained results, especially for the numerous discussions about potential follow-up research questions.

Notes

[1]This work has been partially sponsored by the *Fonds National de la Recherche Luxembourg* (`www.fnr.lu`), via the PEARL programme.

[2]This work was carried out during the tenure of the ERCIM "Alain Bensoussan" Fellowship Programme. This Programme is supported by the Marie Curie Co-funding of Regional, National and International Programmes (COFUND) of the European Commission.

[3]`http://www.reo.project.cwi.nl`

[4]`http://www.3ds.com/products/catia/portfolio/dymola`

[5]`http://modelica.org/`

[6]`http://www.mathworks.nl/products/matlab/`

[7]`http://www.mathworks.nl/products/simulink/`

[8]`http://www.wolfram.com/mathematica/`

[9]Credit Suisse (Luxembourg) S.A., P.O.BOX 40, 56, Grand-Rue, L-2010 Luxembourg.

REFERENCES

Arbab F., 2004. *Reo: a channel-based coordination model for component composition. Mathematical Structures in Computer Science*, 14, no. 3, 329–366.

Brandt C. and Hermann F., 2013. *Conformance Analysis of Organizational Models in a new Enterprise Modeling Framework using Algebraic Graph Transformation. International Journal of Information System Modeling and Design (IJISMD)*, 4, no. 1.

Bubnicki Z., 2005. *Modern control theory.* Springer. ISBN 9783540239512.

Daldrup A., 2005. *Kreditrisikomasse im Vergleich. Universität Göttingen, Wirtschaftsinformatik II - Arbeitsbericht*, , no. 13, 1–33. `http://www2.as.wiwi.uni-goettingen.de/getfile?DateiID=571`.

Faisst U., 2003. *Ein Modell zur Steuerung operationeller Risiken in IT-unterstutzten Bankprozessen.* In

Multikonferenz Wirtschaftsinformatik 2004, Diskussionspapier WI-138. 1–19. http://www.wi-if.de/paperliste/paper/wi-138.pdf.

Giacometti R.; Rachev S.; Chernobai A.; Bertocchi M.; and Consigli G., year unkown. *Practical Operational Risk*. http://statistik.ets.kit.edu/download/doc_secure1/JOR-Practical-Operational-Risk-GRCBC.pdf.

Hechenblaikner A. and zu Selhausen P., 2006. *Operational Risk in Banken: Eine Methodenkritische Analyse Der Messung Von It-Risiken*. Bank- Und Finanzwirtschaft. Deutscher Universitätsverlag. ISBN 9783835004245.

Kunze B. and Poddig P., 2007. *Überwachung operationeller Risiken bei Banken: Interne und externe Akteure im Rahmen qualitativer und quantitativer Überwachung*. Gabler Edition Wissenschaft. Deutscher Universitätsverlag. ISBN 9783835006430.

Mayer N., 2009. *Model-based Management of Information System Security Risk*. Ph.D. thesis, University of Namur. http://www.nmayer.eu/publis/Thesis_Mayer_2.0.pdf.

Nurcan S.; Salinesi C.; Souveyet C.; and Ralyté J., 2010. *Intentional Perspectives on Information Systems Engineering*. Springer. ISBN 9783642125430. URL http://books.google.lu/books?id=4rc4_3gJPOIC.

on Banking Supervision B.C., 2006. *International Convergence of Capital Measurement and Capital Standards*. Bank for International Settlements. http://www.bis.org/publ/bcbs128.pdf.

on Banking Supervision B.C., 2011a. *Operational Risk - Supervisory Guidelines for the Advanced Measurement Approaches*. Bank for International Settlements. ISBN 9789291318568. http://www.bis.org/publ/bcbs196.pdf.

on Banking Supervision B.C., 2011b. *Principles for the Sound Management of Operational Risk*. Bank for International Settlements. ISBN 9789291318575. http://www.bis.org/publ/bcbs195.pdf.

on Banking Supervision B.C., 2012. *Consultative Document - Principles for effective risk data aggregation and risk reporting*. Bank for International Settlements. http://www.bis.org/publ/blbs222.pdf.

Saouma V. and Sivaselvan V., 2008. *Hybrid Simulation: Theory, Implementation and Applications*. Balkema-proceedings and monographs in engineering, water, and earth sciences. Taylor & Francis. ISBN 9780415465687.

Schäl I., 2003. *Die Quantifizierung und Steuerung von operationellen Risiken*. zeb. http://www4.wiwi.uni-karlsruhe.de/MITARBEITER/SCHAEL/OperationelleRisiken.pdf.

Tchernobai A.S., 2006. *Contributions to Modeling of Operational Risks in Banks*. PhD Thesis. University of California Santa Barbara. http://gradworks.umi.com/32/18/3218836.html.

van Bommel P.; Buitenhuis P.; Hoppenbrouwers S.; and Proper E., 2007. *Architecture Principles - A Regulative Perspective on Enterprise Architecture*. In M. Reichert; S. Strecker; and K. Turowski (Eds.), *EMISA*. GI, *LNI*, vol. P-119. ISBN 978-3-88579-213-0, 47–60.

Appendix A

```
within ;
package RiskAnalysis

  connector ServicePort
    Boolean writeFlag(start=false);
    Boolean writeOkFlag(start=false);
    Integer monetaryValue(start=0);
    Integer ID(start=0);
  end ServicePort;

  connector ServicePortIn = ServicePort annotation (Icon(graphics={Ellipse(
          extent={{100,100},{-100,-100}},
          lineColor={0,0,0},
          lineThickness=0.5,
          fillColor={255,255,170},
          fillPattern=FillPattern.Solid,
          pattern=LinePattern.Dash)}));
  connector ServicePortOut = ServicePort annotation (Icon(graphics={Ellipse(
          extent={{100,100},{-100,-100}},
          lineColor={0,0,0},
          lineThickness=0.5,
          fillColor={255,255,0},
          fillPattern=FillPattern.Solid,
          pattern=LinePattern.Dash)}));

  model Client
    Integer currentBusinessValue(start=0);
    Integer accumulatedProfits(start=0);
    Integer state(start=0);
    Integer accumulatedNumberOfSuccessEvents(start = 0);
    Real controlTime(start=0);

    ServicePortOut singleOrderOut annotation (Placement(transformation(extent={
            {80,-10},{100,10}}), iconTransformation(extent={{80,-12},{100,8}})));
    Modelica.Blocks.Interfaces.IntegerInput singleOrderIDIn
      "Receiving the ID of the baskets"       annotation (Placement(transformation(
            extent={{-40,40},{0,80}}), iconTransformation(
          extent={{-13,-13},{13,13}},
          rotation=-90,
          origin={-57,73})));

    Modelica.Blocks.Interfaces.IntegerInput singleOrderPriceIn1
      "Receiving the ID of the baskets"       annotation (Placement(transformation(
            extent={{-40,40},{0,80}}), iconTransformation(
          extent={{-13,-13},{13,13}},
          rotation=-90,
          origin={37,73})));

    Modelica.Blocks.Interfaces.IntegerOutput outputCT annotation (Placement(
          transformation(
          extent={{-15,-15},{15,15}},
          rotation=-90,
          origin={-13,-81}), iconTransformation(
          extent={{-13,-13},{13,13}},
```

```
55            rotation=-90,
56            origin={-11,-73})));
57    equation
58      outputCT = singleOrderOut.monetaryValue;
59
60    algorithm
61      when (pre(state) == 0 and (change(singleOrderIDIn) or change(singleOrderPriceIn1)
          ) and time > controlTime) then
62        singleOrderOut.writeFlag := true;
63        singleOrderOut.ID := singleOrderIDIn;
64        currentBusinessValue := singleOrderPriceIn1;
65        singleOrderOut.monetaryValue := singleOrderPriceIn1;
66        accumulatedProfits := pre(accumulatedProfits) + singleOrderPriceIn1;
67        state := 1;
68        controlTime := time;
69      end when;
70
71      when (pre(state) == 1 and pre(singleOrderOut.writeOkFlag) and time > controlTime)
          then
72        singleOrderOut.writeFlag := false;
73        accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +1;
74        state := 0;
75        controlTime := time;
76      end when;
77
78      annotation (Diagram(graphics={Rectangle(
79              extent={{-100,60},{80,-60}},
80              lineColor={0,0,0},
81              lineThickness=0.5,
82              fillColor={255,255,255},
83              fillPattern=FillPattern.Solid)}), Icon(graphics={Rectangle(
84              extent={{-100,60},{80,-60}},
85              lineColor={0,0,0},
86              lineThickness=0.5,
87              fillColor={255,255,255},
88              fillPattern=FillPattern.Solid), Text(
89              extent={{-70,18},{36,-16}},
90              lineColor={0,0,0},
91              lineThickness=0.5,
92              fillColor={255,255,255},
93              fillPattern=FillPattern.Solid,
94              textString="Client")}));
95    end Client;
96
97    model BasketService
98
99      parameter Integer idLenght = 10000;
100     parameter Integer basketServiceLossesLength= 1000;
101     parameter Integer mX= 1;
102     parameter Integer mY= 1;
103     parameter Integer mZ= 1;
104     String basketServiceFailuresFileName= "sampledLossesForBasketService.txt";
105     Boolean randomSoftwareSuccess(start=false);
106     Integer indexValue(start=0);
107     Integer idTobeServed(start=0);
108     Integer currentBusinessValue(start=0);
109     Integer accumulatedNumberOfLostEvents(start=0);
```

```
110     Integer accumulatedLosses(start=0);
111     Integer accumulatedNumberOfSuccessEvents(start=0);
112     Integer accumulatedProfits(start=0);
113     Integer currentOrderValue(start=0);
114     Integer state(start=0);
115     Real controlTime(start=0);
116
117   equation
118     outAccumulatedLosses = accumulatedLosses;
119     indexValue = integer(ceil(uniformRNG.outPort[1]));
120     basketServiceLossesArray = Modelica.Utilities.Streams.readFile(
            basketServiceFailuresFileName);
121     randomSoftwareSuccess = if (Modelica.Utilities.Strings.scanInteger(
            basketServiceLossesArray[indexValue]) == 0) then false else true;
122
123   public
124     Modelica.Blocks.Interfaces.IntegerOutput outAccumulatedLosses annotation (
            Placement(
125             transformation(
126             extent={{-15,-15},{15,15}},
127             rotation=-90,
128             origin={-3,-71}), iconTransformation(
129             extent={{-13,-13},{13,13}},
130             rotation=90,
131             origin={-1,73})));
132     ServicePortIn singleOrderIn annotation (Placement(transformation(extent={{-102,
133               -10},{-82,10}}), iconTransformation(extent={{-100,-10},{-80,10}})));
134     ServicePortOut basketServiceOut annotation (Placement(transformation(extent={{
135               70,-10},{90,10}}), iconTransformation(extent={{82,-10},{102,10}})));
136
137   public
138      Modelon.Blocks.Sources.RandomNumbers.uniformRNG uniformRNG(nout=1,
139       samplePeriod=0.5,
140       firstSeed=Modelon.Math.RandomNumbers.Seed(
141           x=mX,
142           y=mY,
143           z=mZ),
144       b={999},
145       a={0.01})
146       annotation (Placement(transformation(extent={{-22,-16},{-2,4}})));
147
148   protected
149      Integer[idLenght] partialBasketValue;
150      String[basketServiceLossesLength] basketServiceLossesArray;
151
152   algorithm
153     when (pre(state) == 0 and (not pre(basketServiceOut.writeOkFlag)) and time >
            controlTime) then
154       singleOrderIn.writeOkFlag := true;
155       state := 1;
156       controlTime := time;
157     end when;
158
159     when (pre(state) == 1 and pre(singleOrderIn.writeFlag) and time > controlTime)
          then
160       singleOrderIn.writeOkFlag := false;
161       if (randomSoftwareSuccess) then
```

```
162         idTobeServed := pre(singleOrderIn.ID); //Saving the ID of the received single
                order
163         currentOrderValue := pre(singleOrderIn.monetaryValue);
164         accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                1;
165         accumulatedProfits := pre(accumulatedProfits) + pre(singleOrderIn.
                monetaryValue);
166         currentBusinessValue := pre(currentBusinessValue) + pre(singleOrderIn.
                monetaryValue);
167         state := 2;
168       else
169         accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
170         accumulatedLosses := pre(accumulatedLosses) + pre(singleOrderIn.monetaryValue
                );
171         state := 0;
172       end if;
173       controlTime := time;
174     end when;
175
176     when (pre(state) == 2 and (not pre(singleOrderIn.writeFlag)) and time >
            controlTime) then
177       if (currentOrderValue <> 0) then
178         partialBasketValue[pre(idTobeServed)] := pre(partialBasketValue[pre(
                idTobeServed)]) + pre(currentOrderValue);
179         state := 0;
180       else
181         basketServiceOut.ID := pre(idTobeServed);
182         basketServiceOut.monetaryValue := pre(partialBasketValue[pre(idTobeServed)]);
183         currentBusinessValue := pre(currentBusinessValue) - pre(partialBasketValue[
                pre(idTobeServed)]);
184         basketServiceOut.writeFlag := true;
185         state := 4;
186       end if;
187       controlTime := time;
188     end when;
189
190     when (pre(state) == 4 and pre(basketServiceOut.writeOkFlag)  and time >
            controlTime) then
191       basketServiceOut.writeFlag := false;
192       state := 0;
193       controlTime := time;
194     end when;
195
196     annotation (Diagram(coordinateSystem(extent={{-100,-100},{100,100}},
197           preserveAspectRatio=true),
198                       graphics={Rectangle(
199           extent={{-90,60},{90,-60}},
200           lineColor={0,0,0},
201           lineThickness=0.5,
202           fillColor={255,255,255},
203           fillPattern=FillPattern.Solid)}), Icon(coordinateSystem(extent={{-100,
204             -100},{100,100}}, preserveAspectRatio=true),
205                                                 graphics={Rectangle(
206           extent={{-88,60},{92,-60}},
207           lineColor={0,0,0},
208           lineThickness=0.5,
209           fillColor={255,255,255},
```

```
210             fillPattern=FillPattern.Solid), Text(
211             extent={{-52,18},{54,-16}},
212             lineColor={0,0,0},
213             lineThickness=0.5,
214             fillColor={255,255,255},
215             fillPattern=FillPattern.Solid,
216             textString="Basket")}));
217   end BasketService;
218
219 model BasketService1
220
221   parameter Integer idLenght = 10000;
222   parameter Integer basketServiceLossesLength= 1000;
223   parameter Integer mX= 1;
224   parameter Integer mY= 1;
225   parameter Integer mZ= 1;
226   String basketServiceFailuresFileName= "sampledLossesForBasketService1.txt";
227   Boolean randomSoftwareSuccess(start=false);
228   Integer indexValue(start=0);
229   Integer idTobeServed(start=0);
230   Integer currentBusinessValue(start=0);
231   Integer accumulatedNumberOfLostEvents(start=0);
232   Integer accumulatedLosses(start=0);
233   Integer accumulatedNumberOfSuccessEvents(start=0);
234   Integer accumulatedProfits(start=0);
235   Integer currentOrderValue(start=0);
236   Integer state(start=0);
237   Real controlTime(start=0);
238
239 equation
240   outAccumulatedLosses = accumulatedLosses;
241   indexValue = integer(ceil(uniformRNG.outPort[1]));
242   basketServiceLossesArray = Modelica.Utilities.Streams.readFile(
        basketServiceFailuresFileName);
243   randomSoftwareSuccess = if (Modelica.Utilities.Strings.scanInteger(
        basketServiceLossesArray[indexValue]) == 0) then false else true;
244
245   public
246   Modelica.Blocks.Interfaces.IntegerOutput outAccumulatedLosses annotation (Placement
        (
247         transformation(
248         extent={{-15,-15},{15,15}},
249         rotation=-90,
250         origin={-3,-71}), iconTransformation(
251         extent={{-13,-13},{13,13}},
252         rotation=90,
253         origin={-1,73})));
254   ServicePortIn singleOrderIn annotation (Placement(transformation(extent={{-102,
255             -10},{-82,10}}), iconTransformation(extent={{-100,-10},{-80,10}})));
256   ServicePortOut basketServiceOut annotation (Placement(transformation(extent={{
257             70,-10},{90,10}}), iconTransformation(extent={{82,-10},{102,10}})));
258
259   public
260    Modelon.Blocks.Sources.RandomNumbers.uniformRNG uniformRNG(nout=1,
261     samplePeriod=0.5,
262     firstSeed=Modelon.Math.RandomNumbers.Seed(
263         x=mX,
```

```
264           y=mY,
265           z=mZ),
266         b={999},
267         a={0.01})
268       annotation (Placement(transformation(extent={{-22,-16},{-2,4}})));
269
270     protected
271      Integer[idLenght] partialBasketValue;
272      String[basketServiceLossesLength] basketServiceLossesArray;
273
274   algorithm
275     when (pre(state) == 0 and (not pre(basketServiceOut.writeOkFlag)) and time >
          controlTime) then
276       singleOrderIn.writeOkFlag := true;
277       state := 1;
278       controlTime := time;
279     end when;
280
281     when (pre(state) == 1 and pre(singleOrderIn.writeFlag) and time > controlTime) then
282       singleOrderIn.writeOkFlag := false;
283       if (randomSoftwareSuccess) then
284         idTobeServed := pre(singleOrderIn.ID); //Saving the ID of the received single
              order
285         currentOrderValue := pre(singleOrderIn.monetaryValue);
286         accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) + 1;
287         accumulatedProfits := pre(accumulatedProfits) + pre(singleOrderIn.monetaryValue
              );
288         currentBusinessValue := pre(currentBusinessValue) + pre(singleOrderIn.
              monetaryValue);
289         state := 2;
290       else
291         accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
292         accumulatedLosses := pre(accumulatedLosses) + pre(singleOrderIn.monetaryValue);
293         state := 0;
294       end if;
295       controlTime := time;
296     end when;
297
298     when (pre(state) == 2 and (not pre(singleOrderIn.writeFlag)) and time > controlTime
          ) then
299       if (currentOrderValue <> 0) then
300         partialBasketValue[pre(idTobeServed)] := pre(partialBasketValue[pre(
              idTobeServed)]) + pre(currentOrderValue);
301         state := 0;
302       else
303         basketServiceOut.ID := pre(idTobeServed);
304         basketServiceOut.monetaryValue := pre(partialBasketValue[pre(idTobeServed)]);
305         currentBusinessValue := pre(currentBusinessValue) - pre(partialBasketValue[pre(
              idTobeServed)]);
306         basketServiceOut.writeFlag := true;
307         state := 4;
308       end if;
309       controlTime := time;
310     end when;
311
312     when (pre(state) == 4 and pre(basketServiceOut.writeOkFlag)  and time > controlTime
          ) then
```

```
313       basketServiceOut.writeFlag := false;
314       state := 0;
315       controlTime := time;
316     end when;
317
318     annotation (Diagram(coordinateSystem(extent={{-100,-100},{100,100}},
319             preserveAspectRatio=true),
320                         graphics={Rectangle(
321             extent={{-90,60},{90,-60}},
322             lineColor={0,0,0},
323             lineThickness=0.5,
324             fillColor={255,255,255},
325             fillPattern=FillPattern.Solid)}), Icon(coordinateSystem(extent={{-100,
326               -100},{100,100}}, preserveAspectRatio=true),
327                                                    graphics={Rectangle(
328             extent={{-88,60},{92,-60}},
329             lineColor={0,0,0},
330             lineThickness=0.5,
331             fillColor={255,255,255},
332             fillPattern=FillPattern.Solid), Text(
333             extent={{-52,18},{54,-16}},
334             lineColor={0,0,0},
335             lineThickness=0.5,
336             fillColor={255,255,255},
337             fillPattern=FillPattern.Solid,
338               textString="Basket1")}));
339   end BasketService1;
340
341     model PaymentService
342
343       //parameter Real divideTimeDelay = 100; // The random number of seconds (
              randomPaymentDelay) is divided by this value
344       parameter Integer paymentServiceLossesLength= 1000;
345       parameter Integer paymentServiceDelayLength= 1000;
346       parameter Integer mX= 1;
347       parameter Integer mY= 1;
348       parameter Integer mZ= 1;
349       String paymentServiceLossesFileName = "sampledLossesForPaymentService.txt";
350       String paymentServiceDelayFileName = "sampledDelaysForPaymentService.txt";
351       Integer state(start=0);
352       Real controlTime(start=0);
353       Integer idTobeServed(start=0);
354       Integer indexValue(start=0);
355       Integer currentBusinessValue(start=0);
356       Integer accumulatedNumberOfLostEvents(start=0);
357       Integer accumulatedLosses(start=0);
358       Integer accumulatedNumberOfSuccessEvents(start=0);
359       Integer accumulatedProfits(start=0);
360       Boolean randomPaymentSuccess(start=false);
361       Integer currentBasketValue(start=0);
362       Real delayPayment(start=0);
363
364       ServicePortOut basketOut annotation (Placement(transformation(extent={
365               {80,-10},{100,10}}), iconTransformation(extent={{80,-12},{100,8}})));
366       ServicePortIn basketIn annotation (Placement(transformation(extent={{
367               58,-10},{78,10}}), iconTransformation(extent={{-120,-12},{-100,8}})));
368
```

```
369   public
370     Modelon.Blocks.Sources.RandomNumbers.uniformRNG uniformRNG(nout=1,
371       samplePeriod=0.5,
372       firstSeed=Modelon.Math.RandomNumbers.Seed(
373             x=mX,
374             y=mY,
375             z=mZ),
376       b={999},
377       a={0.01})
378       annotation (Placement(transformation(extent={{-12,-6},{8,14}})));
379
380   public
381     Modelica.Blocks.Interfaces.IntegerOutput outAccumulatedLosses annotation (
          Placement(
382           transformation(
383           extent={{-15,-15},{15,15}},
384           rotation=-90,
385           origin={7,-61}),  iconTransformation(
386           extent={{-13,-13},{13,13}},
387           rotation=90,
388           origin={-9,73})));
389
390   equation
391     outAccumulatedLosses = accumulatedLosses;
392     paymentServiceLossesArray =  Modelica.Utilities.Streams.readFile(
          paymentServiceLossesFileName);
393     paymentServiceDelayArray =   Modelica.Utilities.Streams.readFile(
          paymentServiceDelayFileName);
394     indexValue = integer(ceil(uniformRNG.outPort[1]));
395     randomPaymentSuccess = if (Modelica.Utilities.Strings.scanInteger(
          paymentServiceLossesArray[indexValue]) == 0) then false else true;
396     delayPayment = Modelica.Utilities.Strings.scanReal(paymentServiceDelayArray[
          indexValue]);
397
398   protected
399     String[paymentServiceLossesLength] paymentServiceLossesArray;
400     String[paymentServiceDelayLength] paymentServiceDelayArray;
401
402   algorithm
403     when (pre(state) == 0 and (not pre(basketOut.writeOkFlag)) and time > controlTime
          ) then
404       basketIn.writeOkFlag := true;
405       state := 1;
406       controlTime := time;
407     end when;
408
409     when (pre(state) == 1 and pre(basketIn.writeFlag) and time > controlTime) then
410       basketIn.writeOkFlag := false;
411       idTobeServed:= basketIn.ID;
412       currentBasketValue := pre(basketIn.monetaryValue);
413       currentBusinessValue := pre(basketIn.monetaryValue);
414       state := 2;
415       controlTime := time;
416     end when;
417
418     when (pre(state) == 2 and (not pre(basketIn.writeFlag)) and time > controlTime)
          then
```

```
419         if (randomPaymentSuccess) then
420           accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                1;
421           accumulatedProfits := pre(accumulatedProfits) + pre(currentBasketValue);
422           state := 3;
423         else
424           accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
425           accumulatedLosses := pre(accumulatedLosses) + pre(currentBasketValue);
426           currentBusinessValue := 0;
427           state := 0;
428         end if;
429           controlTime := time + delayPayment;
430       end when;
431
432       when (pre(state) == 3 and time > controlTime) then
433         basketOut.ID := idTobeServed;
434         basketOut.monetaryValue := pre(currentBasketValue);
435         currentBusinessValue := pre(currentBusinessValue) - pre(currentBasketValue);
436         basketOut.writeFlag := true;
437         state := 4;
438         controlTime := time;
439       end when;
440
441       when (pre(state) == 4 and pre(basketOut.writeOkFlag)  and time > controlTime)
            then
442         basketOut.writeFlag := false;
443         state := 0;
444         controlTime := time;
445       end when;
446
447       annotation (Icon(graphics={Rectangle(
448               extent={{-100,60},{80,-60}},
449               lineColor={0,0,0},
450               lineThickness=0.5,
451               fillColor={255,255,255},
452               fillPattern=FillPattern.Solid), Text(
453               extent={{-70,16},{40,-14}},
454               lineColor={0,0,0},
455               lineThickness=0.5,
456               fillColor={255,255,255},
457               fillPattern=FillPattern.Solid,
458               textString="Payment")}), Diagram(graphics));
459   end PaymentService;
460
461   model PaymentService1
462
463   //parameter Real divideTimeDelay = 100; // The random number of seconds (
          randomPaymentDelay) is divided by this value
464   parameter Integer paymentServiceLossesLength= 1000;
465   parameter Integer paymentServiceDelayLength= 1000;
466   parameter Integer mX= 1;
467   parameter Integer mY= 1;
468   parameter Integer mZ= 1;
469   String paymentServiceLossesFileName = "sampledLossesForPaymentService1.txt";
470   String paymentServiceDelayFileName = "sampledDelaysForPaymentService1.txt";
471   Integer state(start=0);
472   Real controlTime(start=0);
```

```
473     Integer idTobeServed(start=0);
474     Integer indexValue(start=0);
475     Integer currentBusinessValue(start=0);
476     Integer accumulatedNumberOfLostEvents(start=0);
477     Integer accumulatedLosses(start=0);
478     Integer accumulatedNumberOfSuccessEvents(start=0);
479     Integer accumulatedProfits(start=0);
480     Boolean randomPaymentSuccess(start=false);
481     Integer currentBasketValue(start=0);
482     Real delayPayment(start=0);
483
484     ServicePortOut basketOut annotation (Placement(transformation(extent={
485               {80,-10},{100,10}}), iconTransformation(extent={{80,-12},{100,8}})));
486     ServicePortIn basketIn annotation (Placement(transformation(extent={{
487               58,-10},{78,10}}), iconTransformation(extent={{-120,-12},{-100,8}})));
488
489     public
490      Modelon.Blocks.Sources.RandomNumbers.uniformRNG uniformRNG(nout=1,
491       samplePeriod=0.5,
492       firstSeed=Modelon.Math.RandomNumbers.Seed(
493             x=mX,
494             y=mY,
495             z=mZ),
496         b={999},
497         a={0.01})
498       annotation (Placement(transformation(extent={{-12,-6},{8,14}})));
499
500     public
501     Modelica.Blocks.Interfaces.IntegerOutput outAccumulatedLosses annotation (Placement
          (
502           transformation(
503           extent={{-15,-15},{15,15}},
504           rotation=-90,
505           origin={7,-61}),  iconTransformation(
506           extent={{-13,-13},{13,13}},
507           rotation=90,
508           origin={-9,73})));
509
510     equation
511     outAccumulatedLosses = accumulatedLosses;
512     paymentServiceLossesArray =  Modelica.Utilities.Streams.readFile(
          paymentServiceLossesFileName);
513     paymentServiceDelayArray =  Modelica.Utilities.Streams.readFile(
          paymentServiceDelayFileName);
514     indexValue = integer(ceil(uniformRNG.outPort[1]));
515     randomPaymentSuccess = if (Modelica.Utilities.Strings.scanInteger(
          paymentServiceLossesArray[indexValue]) == 0) then false else true;
516     delayPayment = Modelica.Utilities.Strings.scanReal(paymentServiceDelayArray[
          indexValue]);
517
518     protected
519     String[paymentServiceLossesLength] paymentServiceLossesArray;
520     String[paymentServiceDelayLength] paymentServiceDelayArray;
521
522     algorithm
523     when (pre(state) == 0 and (not pre(basketOut.writeOkFlag)) and time > controlTime)
          then
```

```
524       basketIn.writeOkFlag := true;
525       state := 1;
526       controlTime := time;
527     end when;
528
529     when (pre(state) == 1 and pre(basketIn.writeFlag) and time > controlTime) then
530       basketIn.writeOkFlag := false;
531       idTobeServed:= basketIn.ID;
532       currentBasketValue := pre(basketIn.monetaryValue);
533       currentBusinessValue := pre(basketIn.monetaryValue);
534       state := 2;
535       controlTime := time;
536     end when;
537
538     when (pre(state) == 2 and (not pre(basketIn.writeFlag)) and time > controlTime)
          then
539       if (randomPaymentSuccess) then
540         accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) + 1;
541         accumulatedProfits := pre(accumulatedProfits) + pre(currentBasketValue);
542         state := 3;
543       else
544         accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
545         accumulatedLosses := pre(accumulatedLosses) + pre(currentBasketValue);
546         currentBusinessValue := 0;
547         state := 0;
548       end if;
549         controlTime := time + delayPayment;
550     end when;
551
552     when (pre(state) == 3 and time > controlTime) then
553       basketOut.ID := idTobeServed;
554       basketOut.monetaryValue := pre(currentBasketValue);
555       currentBusinessValue := pre(currentBusinessValue) - pre(currentBasketValue);
556       basketOut.writeFlag := true;
557       state := 4;
558       controlTime := time;
559     end when;
560
561     when (pre(state) == 4 and pre(basketOut.writeOkFlag)  and time > controlTime) then
562       basketOut.writeFlag := false;
563       state := 0;
564       controlTime := time;
565     end when;
566
567     annotation (Icon(graphics={Rectangle(
568             extent={{-100,60},{80,-60}},
569             lineColor={0,0,0},
570             lineThickness=0.5,
571             fillColor={255,255,255},
572             fillPattern=FillPattern.Solid), Text(
573             extent={{-70,16},{44,-14}},
574             lineColor={0,0,0},
575             lineThickness=0.5,
576             fillColor={255,255,255},
577             fillPattern=FillPattern.Solid,
578               textString="Payment1")}),
579                                       Diagram(graphics));
```

```
580   end PaymentService1;
581
582   model CollectOrderService
583
584     parameter Integer lengthFIFO = 10000;
585     Integer state(start=0);
586     Real controlTime(start=0);
587     Integer idTobeServed(start=0);
588     Integer nElementsFIFO(start=0);
589     Integer inPointerFIFO(start=1);
590     Integer outPointerFIFO(start=1);
591     Integer accumulatedProfits(start=0);
592     Integer currentBusinessValue(start=0);
593     Integer currentBasketValue(start=0);
594     Integer accumulatedNumberOfSuccessEvents(start=0);
595     Boolean check(start=false);
596
597   public
598     ServicePortIn basketIn annotation (Placement(transformation(extent={{-110,-10},
599               {-90,10}}), iconTransformation(extent={{-118,-10},{-98,10}})));
600     ServicePortOut accountStatementOut annotation (Placement(transformation(
601             extent={{6,-48},{26,-28}}), iconTransformation(extent={{62,-12},{82,8}}))
                );
602     Modelica.Blocks.Interfaces.BooleanInput emptyEvent annotation (Placement(
603           transformation(
604           extent={{-20,-20},{20,20}},
605           rotation=-90,
606           origin={-28,74}), iconTransformation(
607           extent={{-13,-13},{13,13}},
608           rotation=-90,
609           origin={-21,73})));
610
611   protected
612     Integer[lengthFIFO] FIFO;
613     Integer[lengthFIFO] prices;
614
615   algorithm
616     when (pre(state) == 0 and (not pre(accountStatementOut.writeOkFlag)) and time >
          controlTime) then
617       basketIn.writeOkFlag := true;
618       state := 1;
619       controlTime := time;
620     end when;
621
622     when (pre(state) == 1 and pre(basketIn.writeFlag) and time > controlTime) then
623       basketIn.writeOkFlag := false;
624       idTobeServed := pre(basketIn.ID); //Saving the received ID of the basket
625       currentBasketValue := pre(basketIn.monetaryValue);
626       state := 2;
627       controlTime := time;
628     end when;
629
630     // To flush out when I receive a signal, but I do not receive basket
631     when (pre(state) == 1 and (not pre(basketIn.writeFlag)) and pre(check) and time >
          controlTime) then
632       state := 5;
633       check := false;
```

```
634         controlTime := time;
635       end when;
636
637       /* adding the ID of the basket to the internal FIFO */
638       when (pre(state) == 2 and (not pre(basketIn.writeFlag)) and time > controlTime)
              then
639
640         FIFO[pre(inPointerFIFO)] := pre(idTobeServed);
641         prices[pre(inPointerFIFO)] := pre(currentBasketValue);
642         nElementsFIFO := pre(nElementsFIFO) + 1;
643         inPointerFIFO := pre(inPointerFIFO) + 1;
644         currentBusinessValue := pre(currentBusinessValue) + pre(currentBasketValue);
645         accumulatedProfits := pre(accumulatedProfits) + pre(currentBasketValue);
646         accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) + 1;
647         if (not pre(check)) then
648           state := 0;
649         else
650           state:= 3;
651           check:= false;
652         end if;
653       end when;
654
655       when (change(emptyEvent) and pre(nElementsFIFO) > 0) then
656          check := true;
657       end when;
658
659       /* Sequence of message to empty the basket */
660       when (pre(state) == 3 and (not pre(accountStatementOut.writeOkFlag)) and time >
              controlTime) then
661         accountStatementOut.ID := FIFO[pre(outPointerFIFO)];
662         accountStatementOut.monetaryValue := prices[pre(outPointerFIFO)];
663         currentBusinessValue := pre(currentBusinessValue) - prices[pre(outPointerFIFO)
              ];
664         nElementsFIFO := pre(nElementsFIFO) - 1;
665         outPointerFIFO := pre(outPointerFIFO) + 1;
666         accountStatementOut.writeFlag := true;
667         state := 4;
668         controlTime := time;
669       end when;
670
671       when (pre(state) == 4 and pre(accountStatementOut.writeOkFlag)  and time >
              controlTime) then
672         accountStatementOut.writeFlag := false;
673         if (pre(nElementsFIFO) > 0) then
674            state := 3;
675         else
676           state := 0;
677           basketIn.writeOkFlag := false;
678           inPointerFIFO := 1;
679           outPointerFIFO := 1;
680         end if;
681         controlTime := time;
682       end when;
683
684        /* Sequence of message to empty the basket */
685       when (pre(state) == 5 and (not pre(accountStatementOut.writeOkFlag)) and time >
              controlTime) then
```

```
686         accountStatementOut.ID := FIFO[pre(outPointerFIFO)];
687         accountStatementOut.monetaryValue := prices[pre(outPointerFIFO)];
688         currentBusinessValue := pre(currentBusinessValue) - prices[pre(outPointerFIFO)
                ];
689         nElementsFIFO := pre(nElementsFIFO) - 1;
690         outPointerFIFO := pre(outPointerFIFO) + 1;
691         accountStatementOut.writeFlag := true;
692         state := 6;
693         controlTime := time;
694       end when;
695
696       when (pre(state) == 6 and pre(accountStatementOut.writeOkFlag)  and time >
              controlTime) then
697         accountStatementOut.writeFlag := false;
698         if (pre(nElementsFIFO) > 0) then
699           state := 5;
700         else
701           state := 0;
702           inPointerFIFO := 1;
703           outPointerFIFO := 1;
704         end if;
705         controlTime := time;
706       end when;
707
708     equation
709       connect(accountStatementOut, accountStatementOut) annotation (Line(
710           points={{16,-38},{12,-38},{12,-38},{16,-38}},
711           color={0,0,0},
712           pattern=LinePattern.Dash,
713           thickness=0.5,
714           smooth=Smooth.None));
715       annotation (Icon(coordinateSystem(extent={{-120,-100},{80,100}},
716               preserveAspectRatio=true),
717                       graphics={Rectangle(
718               extent={{-108,60},{72,-60}},
719               lineColor={0,0,0},
720               lineThickness=0.5,
721               fillColor={255,255,255},
722               fillPattern=FillPattern.Solid), Text(
723               extent={{-72,14},{32,-18}},
724               lineColor={0,0,0},
725               lineThickness=0.5,
726               fillColor={255,255,255},
727               fillPattern=FillPattern.Solid,
728               textString="Collect")}), Diagram(coordinateSystem(extent={{-120,-100},
729                 {80,100}}, preserveAspectRatio=true),
730                                                 graphics));
731     end CollectOrderService;
732
733     model SendMailService
734
735       parameter Integer sendMailServiceDelayLength= 1000;
736       parameter Integer mX= 1;
737       parameter Integer mY= 1;
738       parameter Integer mZ= 1;
739       String sendMailServiceDelayFileName = "sampledDelaysForMailService.txt";
740       Integer indexValue(start=0);
```

```
741     Integer state(start=0);
742     Real controlTime(start=0);
743     Integer idTobeServed(start=0);
744     Integer currentBasketValue(start=0);
745     Integer outAccountStatement(start=0);
746     Integer accumulatedProfits(start=0);
747     Integer accumulatedNumberOfSuccessEvents(start=0);
748     Integer currentBusinessValue(start=0);
749     Integer outAccountMonetaryValue(start=0);
750     Real sendMailDelay(start=0);
751
752   equation
753     outputCT = outAccountMonetaryValue;
754     sendMailServiceDelayArray = Modelica.Utilities.Streams.readFile(
          sendMailServiceDelayFileName);
755     indexValue = integer(ceil(uniformRNG.outPort[1]));
756     sendMailDelay = Modelica.Utilities.Strings.scanReal(sendMailServiceDelayArray[
          indexValue]);
757
758   public
759     ServicePortIn mailServiceIn annotation (Placement(transformation(extent={{
760             60,-8},{80,12}}), iconTransformation(extent={{-100,-12},{-80,8}})));
761
762     Modelica.Blocks.Interfaces.IntegerOutput outputCT annotation (Placement(
763           transformation(
764           extent={{-15,-15},{15,15}},
765           rotation=-90,
766           origin={-3,-71}), iconTransformation(
767           extent={{-13,-13},{13,13}},
768           rotation=-90,
769           origin={11,-73})));
770     Modelon.Blocks.Sources.RandomNumbers.uniformRNG uniformRNG(nout=1,
771       samplePeriod=0.5,
772       firstSeed=Modelon.Math.RandomNumbers.Seed(
773           x=mX,
774           y=mY,
775           z=mZ),
776       b={999},
777       a={0.01})
778       annotation (Placement(transformation(extent={{-12,-6},{8,14}})));
779
780   protected
781     String[sendMailServiceDelayLength] sendMailServiceDelayArray;
782
783   algorithm
784     when (pre(state) == 0 and time > controlTime) then
785       mailServiceIn.writeOkFlag := true;
786       state := 1;
787       controlTime := time;
788     end when;
789
790     when (pre(state) == 1 and pre(mailServiceIn.writeFlag) and time > controlTime)
          then
791       mailServiceIn.writeOkFlag := false;
792       idTobeServed := pre(mailServiceIn.ID);
793       currentBasketValue := pre(mailServiceIn.monetaryValue);
794       currentBusinessValue := pre(mailServiceIn.monetaryValue);
```

```
795         state := 2;
796         controlTime := time;
797       end when;
798
799       when (pre(state) == 2 and (not pre(mailServiceIn.writeFlag)) and time >
              controlTime) then
800         state := 3;
801         controlTime := time + sendMailDelay;
802       end when;
803
804       when (pre(state) == 3  and time > controlTime) then
805         outAccountStatement := pre(idTobeServed);
806         outAccountMonetaryValue := pre(currentBasketValue);
807         accumulatedProfits := pre(accumulatedProfits) + pre(currentBasketValue);
808         accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) + 1;
809         currentBusinessValue := 0;
810         state := 0;
811         controlTime := time;
812       end when;
813
814       annotation (Icon(graphics={Rectangle(
815               extent={{-80,60},{100,-60}},
816               lineColor={0,0,0},
817               lineThickness=0.5,
818               fillColor={255,255,255},
819               fillPattern=FillPattern.Solid), Text(
820               extent={{-54,18},{72,-18}},
821               lineColor={0,0,0},
822               lineThickness=0.5,
823               fillColor={255,255,255},
824               fillPattern=FillPattern.Solid,
825               textString="Mail")}), Diagram(graphics));
826     end SendMailService;
827
828     model ForwardService
829
830       Integer state(start=0);
831       Integer counterMessages(start = 0);
832       Real controlTime(start=0);
833
834       ServicePortOut forwardServiceOut annotation (Placement(transformation(extent={
835               {80,-10},{100,10}}), iconTransformation(extent={{86,-2},{106,18}})));
836       ServicePortIn forwardServiceIn annotation (Placement(transformation(extent={{
837               60,-10},{80,10}}), iconTransformation(extent={{-114,-2},{-94,18}})));
838
839     algorithm
840       when (pre(state) == 0 and (not pre(forwardServiceIn.writeFlag)) and time >
              controlTime) then
841         forwardServiceIn.writeOkFlag := true;
842         state := 1;
843         controlTime := time;
844       end when;
845
846       when (pre(state) == 1 and pre(forwardServiceIn.writeFlag) and time > controlTime
              ) then
847         forwardServiceOut.writeFlag := true;
848         forwardServiceOut.ID := pre(forwardServiceIn.ID);
```

```
849         forwardServiceOut.monetaryValue := pre(forwardServiceIn.monetaryValue);
850         state := 2;
851         controlTime := time;
852       end when;
853
854       when (pre(state) == 2 and pre(forwardServiceOut.writeOkFlag) and time >
              controlTime) then
855         forwardServiceOut.writeFlag := false;
856         state := 3;
857         controlTime := time;
858       end when;
859
860       when (pre(state) == 3 and (not pre(forwardServiceOut.writeOkFlag)) and time >
              controlTime) then
861         forwardServiceIn.writeOkFlag := false;
862         counterMessages := pre(counterMessages) + 1;
863         state := 0;
864         controlTime := time;
865       end when;
866
867     annotation (Icon(graphics={Rectangle(
868             extent={{-94,70},{86,-50}},
869             lineColor={0,0,0},
870             lineThickness=0.5,
871             fillColor={255,255,255},
872             fillPattern=FillPattern.Solid), Text(
873             extent={{-58,28},{48,-6}},
874             lineColor={0,0,0},
875             lineThickness=0.5,
876             fillColor={255,255,255},
877             fillPattern=FillPattern.Solid,
878             textString="FwS")}));
879   end ForwardService;
880
881   model ReoCircuit1
882
883     parameter Integer lengthFIFO1 = 100;
884     parameter Integer lengthFIFO2 = 100;
885     Integer nElementsFIFO1(start=0);
886     Integer nElementsFIFO2(start=0);
887     Integer inPointerFIFO1(start=1);
888     Integer inPointerFIFO2(start=1);
889     Integer outPointerFIFO1(start=1);
890     Integer outPointerFIFO2(start=1);
891     Integer moneyLossTotalFIFO1(start=0);
892     Integer moneyLossTotalFIFO2(start=0);
893     Integer lostEventsFIFO1(start=0);
894     Integer lostEventsFIFO2(start=0);
895     Integer successEventsFIFO1(start=0);
896     Integer successEventsFIFO2(start=0);
897     Integer cBusiness1(start=0);
898     Integer cBusiness2(start=0);
899     Integer aProfits1(start=0);
900     Integer aProfits2(start=0);
901     Integer state(start=0);
902     Real controlTime(start=0);
903
```

```
904     Real freeCapacity(start=0);
905     Real usedCapacity(start=0);
906     Integer accumulatedLosses(start=0);
907     Integer accumulatedNumberOfLostEvents(start=0);
908     Integer accumulatedNumberOfSuccessEvents(start=0);
909     Integer accumulatedProfits(start=0);
910     Integer currentBusinessValue(start=0);
911
912   public
913     ServicePortOut portOut1 annotation (Placement(transformation(extent={{68,24},
914               {88,44}}), iconTransformation(extent={{68,24},{88,44}})));
915     ServicePortOut portOut2 annotation (Placement(transformation(extent={{68,
916               -48},{88,-28}}), iconTransformation(extent={{68,-48},{88,-28}})));
917     ServicePortIn portIn annotation (Placement(transformation(extent={{-100,-12},
918               {-80,8}}), iconTransformation(extent={{-100,-12},{-80,8}})));
919     Modelica.Blocks.Interfaces.RealOutput currentFIFOCapacity annotation (
920         Placement(transformation(extent={{-36,-62},{-16,-42}}),
921           iconTransformation(
922           extent={{-10,-10},{10,10}},
923           rotation=-90,
924           origin={-20,-68})));
925     Modelica.Blocks.Interfaces.IntegerOutput outAccumulatedLosses annotation (
926         Placement(transformation(extent={{-92,-74},{-72,-54}}),
927           iconTransformation(
928           extent={{-10,-10},{10,10}},
929           rotation=-90,
930           origin={20,-68})));
931
932   protected
933     Integer[lengthFIFO1] FIFO1;
934     Integer[lengthFIFO1] price1;
935     Integer[lengthFIFO2] FIFO2;
936     Integer[lengthFIFO2] price2;
937
938   equation
939     usedCapacity =  ((nElementsFIFO1 + nElementsFIFO2) * 100) / (lengthFIFO1 +
          lengthFIFO2);
940     freeCapacity = 100 - usedCapacity;
941     accumulatedLosses = moneyLossTotalFIFO1 + moneyLossTotalFIFO2;
942     accumulatedNumberOfLostEvents =  lostEventsFIFO1 + lostEventsFIFO2;
943     accumulatedNumberOfSuccessEvents = successEventsFIFO1 + successEventsFIFO2;
944     currentBusinessValue = cBusiness1 + cBusiness2;
945     accumulatedProfits = aProfits1 + aProfits2;
946     currentFIFOCapacity = usedCapacity;
947     outAccumulatedLosses = accumulatedLosses;
948
949   algorithm
950     /* The circuit receives data from Client, while both portService1 and
          portService2 are not ready to read*/
951     when (pre(state) == 0 and pre(portIn.writeFlag) and (not pre(portOut1.writeOkFlag
          ) or (not pre(nElementsFIFO1) > 0)) and (not pre(portOut2.writeOkFlag) or (not
           pre(nElementsFIFO2) > 0)) and time > controlTime) then
952       if (pre(nElementsFIFO1) < lengthFIFO1) then
953         FIFO1[pre(inPointerFIFO1)] := portIn.ID;
954         price1[pre(inPointerFIFO1)] := portIn.monetaryValue;
955         inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
956         nElementsFIFO1 := pre(nElementsFIFO1) + 1;
```

```
957           successEventsFIFO1 := pre(successEventsFIFO1) + 1;
958           aProfits1 := pre(aProfits1) + portIn.monetaryValue;
959           cBusiness1 := pre(cBusiness1) + portIn.monetaryValue;
960         else
961           moneyLossTotalFIFO1 := pre(moneyLossTotalFIFO1) + portIn.monetaryValue;
962           lostEventsFIFO1 := pre(lostEventsFIFO1) + 1;
963         end if;
964         if (pre(nElementsFIFO2) < lengthFIFO2) then
965           FIFO2[pre(inPointerFIFO2)] := portIn.ID;
966           price2[pre(inPointerFIFO2)] := portIn.monetaryValue;
967           inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
968           nElementsFIFO2 := pre(nElementsFIFO2) + 1;
969           successEventsFIFO2 := pre(successEventsFIFO2) + 1;
970           aProfits2 := pre(aProfits2) + portIn.monetaryValue;
971           cBusiness2 := pre(cBusiness2) + portIn.monetaryValue;
972         else
973           moneyLossTotalFIFO2 := pre(moneyLossTotalFIFO2) + portIn.monetaryValue;
974           lostEventsFIFO2 := pre(lostEventsFIFO2) + 1;
975         end if;
976         portIn.writeOkFlag := true;
977         state := 1;
978         controlTime := time;
979       end when;
980
981       /* portService1 and portService2 read at the same time */
982       when (pre(state) == 0 and (not pre(portIn.writeFlag)) and pre(portOut1.
              writeOkFlag) and (pre(nElementsFIFO1) > 0) and pre(portOut2.writeOkFlag) and (
              pre(nElementsFIFO2) > 0) and time > controlTime) then
983         /* Reading from FIFO1 */
984         portOut1.ID := FIFO1[pre(outPointerFIFO1)];
985         portOut1.monetaryValue := price1[pre(outPointerFIFO1)];
986         outPointerFIFO1 := mod(pre(outPointerFIFO1), lengthFIFO1) + 1;
987         nElementsFIFO1 := pre(nElementsFIFO1) - 1;
988         cBusiness1 := pre(cBusiness1) - price1[pre(outPointerFIFO1)];
989         /* Reading from FIFO2 */
990         portOut2.ID := FIFO2[pre(outPointerFIFO2)];
991         portOut2.monetaryValue := price2[pre(outPointerFIFO2)];
992         outPointerFIFO2 := mod(pre(outPointerFIFO2), lengthFIFO2) + 1;
993         nElementsFIFO2 := pre(nElementsFIFO2) - 1;
994         cBusiness2 := pre(cBusiness2) - price2[pre(outPointerFIFO2)];
995         /* Setting the flags on the ports */
996         portOut1.writeFlag := true;
997         portOut2.writeFlag := true;
998         state := 2;
999         controlTime := time;
1000      end when;
1001
1002      /* Reading from Client ad writing to portService1 at the same time */
1003      when (pre(state) == 0 and pre(portIn.writeFlag) and pre(portOut1.writeOkFlag) and
              (pre(nElementsFIFO1) > 0) and ((not pre(portOut2.writeOkFlag)) or (not pre(
             nElementsFIFO2) > 0)) and time > controlTime) then
1004        if (pre(nElementsFIFO1) < lengthFIFO1) then
1005          FIFO1[pre(inPointerFIFO1)] := portIn.ID;
1006          price1[pre(inPointerFIFO1)] := portIn.monetaryValue;
1007          inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
1008          successEventsFIFO1 := pre(successEventsFIFO1) + 1;
1009          aProfits1 := pre(aProfits1) + portIn.monetaryValue;
```

```
1010          cBusiness1 := pre(cBusiness1) + portIn.monetaryValue - price1[pre(
                  outPointerFIFO1)];
1011        else
1012          nElementsFIFO1 := pre(nElementsFIFO1) - 1;
1013          moneyLossTotalFIFO1 := pre(moneyLossTotalFIFO1) + portIn.monetaryValue;
1014          lostEventsFIFO1 := pre(lostEventsFIFO1) + 1;
1015          cBusiness1 := pre(cBusiness1) - price1[pre(outPointerFIFO1)];
1016        end if;
1017        if (pre(nElementsFIFO2) < lengthFIFO2) then
1018          FIFO2[pre(inPointerFIFO2)] := portIn.ID;
1019          price2[pre(inPointerFIFO2)] := portIn.monetaryValue;
1020          inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
1021          nElementsFIFO2 := pre(nElementsFIFO2) + 1;
1022          successEventsFIFO2 := pre(successEventsFIFO2) + 1;
1023          aProfits2 := pre(aProfits2) + portIn.monetaryValue;
1024          cBusiness2 := pre(cBusiness2) + portIn.monetaryValue;
1025        else
1026          moneyLossTotalFIFO2 := pre(moneyLossTotalFIFO2) + portIn.monetaryValue;
1027          lostEventsFIFO2 := pre(lostEventsFIFO2) + 1;
1028        end if;
1029        /* Reading from FIFO1 */
1030        portOut1.ID := FIFO1[pre(outPointerFIFO1)];
1031        portOut1.monetaryValue := price1[pre(outPointerFIFO1)];
1032        outPointerFIFO1 := mod(pre(outPointerFIFO1), lengthFIFO1) + 1;
1033        /* Setting the flags on the ports */
1034        portIn.writeOkFlag := true;
1035        portOut1.writeFlag := true;
1036        state := 3;
1037        controlTime := time;
1038      end when;
1039
1040      /* Reading from Client ad writing to portService2 at the same time */
1041      when (pre(state) == 0 and pre(portIn.writeFlag) and ((not pre(portOut1.
              writeOkFlag)) or (not pre(nElementsFIFO1) > 0)) and (pre(portOut2.writeOkFlag)
              ) and (pre(nElementsFIFO2) > 0) and time > controlTime) then
1042        if (pre(nElementsFIFO1) < lengthFIFO1) then
1043          FIFO1[pre(inPointerFIFO1)] := portIn.ID;
1044          price1[pre(inPointerFIFO1)] := portIn.monetaryValue;
1045          inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
1046          nElementsFIFO1 := pre(nElementsFIFO1) + 1;
1047          successEventsFIFO1 := pre(successEventsFIFO1) + 1;
1048          aProfits1 := pre(aProfits1) + portIn.monetaryValue;
1049          cBusiness1 := pre(cBusiness1) + portIn.monetaryValue;
1050        else
1051          moneyLossTotalFIFO1 := pre(moneyLossTotalFIFO1) + portIn.monetaryValue;
1052          lostEventsFIFO1 := pre(lostEventsFIFO1) + 1;
1053        end if;
1054        if (pre(nElementsFIFO2) < lengthFIFO2) then
1055          FIFO2[pre(inPointerFIFO2)] := portIn.ID;
1056          price2[pre(inPointerFIFO2)] := portIn.monetaryValue;
1057          inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
1058          successEventsFIFO2 := pre(successEventsFIFO2) + 1;
1059          aProfits2 := pre(aProfits2) + portIn.monetaryValue;
1060          cBusiness2 := pre(cBusiness2) + portIn.monetaryValue - price2[pre(
                  outPointerFIFO2)];
1061        else
1062          nElementsFIFO2 := pre(nElementsFIFO2) - 1;
```

```
1063            moneyLossTotalFIFO2 := pre(moneyLossTotalFIFO2) + portIn.monetaryValue;
1064            lostEventsFIFO2 := pre(lostEventsFIFO2) + 1;
1065            cBusiness2 := pre(cBusiness2) - price2[pre(outPointerFIFO2)];
1066          end if;
1067          /* Reading from FIFO2 */
1068          portOut2.ID := FIFO2[pre(outPointerFIFO2)];
1069          portOut2.monetaryValue := price2[pre(outPointerFIFO2)];
1070          outPointerFIFO2 := mod(pre(outPointerFIFO2), lengthFIFO2) + 1;
1071          /* Setting the flags on the ports */
1072          portIn.writeOkFlag := true;
1073          portOut2.writeFlag := true;
1074          state := 4;
1075          controlTime := time;
1076        end when;
1077
1078        /* Writing to portService1 */
1079        when (pre(state) == 0 and (not pre(portIn.writeFlag)) and pre(portOut1.
                writeOkFlag) and (pre(nElementsFIFO1) > 0) and ((not pre(portOut2.writeOkFlag)
                ) or (not pre(nElementsFIFO2) > 0)) and time > controlTime) then
1080          /* Reading from FIFO1 */
1081          portOut1.ID := FIFO1[pre(outPointerFIFO1)];
1082          portOut1.monetaryValue := price1[pre(outPointerFIFO1)];
1083          outPointerFIFO1 := mod(pre(outPointerFIFO1), lengthFIFO1) + 1;
1084          nElementsFIFO1 := pre(nElementsFIFO1) - 1;
1085          cBusiness1 := pre(cBusiness1) - price1[pre(outPointerFIFO1)];
1086          /* Setting the flags on the ports */
1087          portOut1.writeFlag := true;
1088          state := 5;
1089          controlTime := time;
1090        end when;
1091
1092        /* Writing to portService2 */
1093        when (pre(state) == 0 and (not pre(portIn.writeFlag)) and ((not pre(portOut1.
                writeOkFlag)) or (not pre(nElementsFIFO1) > 0)) and (pre(portOut2.writeOkFlag)
                ) and (pre(nElementsFIFO2) > 0) and time > controlTime) then
1094          /* Reading from FIFO2 */
1095          portOut2.ID := FIFO2[pre(outPointerFIFO2)];
1096          portOut2.monetaryValue := price2[pre(outPointerFIFO2)];
1097          outPointerFIFO2 := mod(pre(outPointerFIFO2), lengthFIFO2) + 1;
1098          nElementsFIFO2 := pre(nElementsFIFO2) - 1;
1099          cBusiness2 := pre(cBusiness2) - price2[pre(outPointerFIFO2)];
1100          /* Setting the flags on the ports */
1101          portOut2.writeFlag := true;
1102          state := 6;
1103          controlTime := time;
1104        end when;
1105
1106        /* Reading from Client ad writing to portService1 and portService2 at the same
                time */
1107        when (pre(state) == 0 and pre(portIn.writeFlag) and pre(portOut1.writeOkFlag) and
                 (pre(nElementsFIFO1) > 0) and pre(portOut2.writeOkFlag) and pre(
               nElementsFIFO2) > 0 and time > controlTime) then
1108          if (pre(nElementsFIFO1) < lengthFIFO1) then
1109            FIFO1[pre(inPointerFIFO1)] := portIn.ID;
1110            price1[pre(inPointerFIFO1)] := portIn.monetaryValue;
1111            inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
1112            successEventsFIFO1 := pre(successEventsFIFO1) + 1;
```

```
1113          else
1114            nElementsFIFO1 := pre(nElementsFIFO1) - 1;
1115            moneyLossTotalFIFO1 := pre(moneyLossTotalFIFO1) + portIn.monetaryValue;
1116            lostEventsFIFO1 := pre(lostEventsFIFO1) + 1;
1117          end if;
1118          if (pre(nElementsFIFO2) < lengthFIFO2) then
1119            FIFO2[pre(inPointerFIFO2)] := portIn.ID;
1120            price2[pre(inPointerFIFO2)] := portIn.monetaryValue;
1121            inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
1122            successEventsFIFO2 := pre(successEventsFIFO2) + 1;
1123          else
1124            nElementsFIFO2 := pre(nElementsFIFO2) - 1;
1125            moneyLossTotalFIFO2 := pre(moneyLossTotalFIFO2) + portIn.monetaryValue;
1126            lostEventsFIFO2 := pre(lostEventsFIFO2) + 1;
1127          end if;
1128          /* Reading from FIFO1 */
1129          portOut1.ID := FIFO1[pre(outPointerFIFO1)];
1130          portOut1.monetaryValue := price1[pre(outPointerFIFO1)];
1131          outPointerFIFO1 := mod(pre(outPointerFIFO1), lengthFIFO1) + 1;
1132          /* Reading from FIFO2 */
1133          portOut2.ID := FIFO2[pre(outPointerFIFO2)];
1134          portOut2.monetaryValue := price2[pre(outPointerFIFO2)];
1135          outPointerFIFO2 := mod(pre(outPointerFIFO2), lengthFIFO2) + 1;
1136          /* Setting the flags on the ports */
1137          portIn.writeOkFlag := true;
1138          portOut1.writeFlag := true;
1139          portOut2.writeFlag := true;
1140          state := 7;
1141          controlTime := time;
1142        end when;
1143
1144        when (pre(state) == 1 and (not pre(portIn.writeFlag)) and time > controlTime)
               then
1145          portIn.writeOkFlag := false;
1146          state := 0;
1147          controlTime := time;
1148        end when;
1149
1150        when (pre(state) == 2 and (not pre(portOut1.writeOkFlag)) and (not pre(portOut2.
               writeOkFlag)) and time > controlTime) then
1151          portOut1.writeFlag := false;
1152          portOut2.writeFlag := false;
1153          state := 0;
1154          controlTime := time;
1155        end when;
1156
1157        when (pre(state) == 3 and (not pre(portIn.writeFlag)) and (not pre(portOut1.
               writeOkFlag)) and time > controlTime) then
1158          portIn.writeOkFlag := false;
1159          portOut1.writeFlag := false;
1160          state := 0;
1161          controlTime := time;
1162        end when;
1163
1164        when (pre(state) == 4 and (not pre(portIn.writeFlag)) and (not pre(portOut2.
               writeOkFlag)) and time > controlTime) then
1165          portIn.writeOkFlag := false;
```

```
1166        portOut2.writeFlag := false;
1167        state := 0;
1168        controlTime := time;
1169      end when;
1170
1171      when (pre(state) == 5 and (not pre(portOut1.writeOkFlag)) and time > controlTime)
              then
1172        portOut1.writeFlag := false;
1173        state := 0;
1174        controlTime := time;
1175      end when;
1176
1177      when (pre(state) == 6 and (not pre(portOut2.writeOkFlag)) and time > controlTime)
              then
1178        portOut2.writeFlag := false;
1179        state := 0;
1180        controlTime := time;
1181      end when;
1182
1183      when (pre(state) == 7 and (not pre(portIn.writeFlag)) and (not pre(portOut1.
            writeOkFlag)) and (not pre(portOut2.writeOkFlag)) and time > controlTime) then
1184        portIn.writeOkFlag := false;
1185        portOut1.writeFlag := false;
1186        portOut2.writeFlag := false;
1187        state := 0;
1188        controlTime := time;
1189      end when;
1190
1191    equation
1192      connect(portOut2, portOut2) annotation (Line(
1193          points={{78,-38},{79.5,-38},{79.5,-42},{81,-42},{81,-38},{78,-38}},
1194          color={0,0,0},
1195          pattern=LinePattern.Dash,
1196          thickness=0.5,
1197          smooth=Smooth.None));
1198      connect(portOut1, portOut1) annotation (Line(
1199          points={{78,34},{78,34},{78,34},{78,34}},
1200          color={0,0,0},
1201          pattern=LinePattern.Dash,
1202          thickness=0.5,
1203          smooth=Smooth.None));
1204      connect(portIn, portIn) annotation (Line(
1205          points={{-90,-2},{-90,-2}},
1206          color={0,0,0},
1207          pattern=LinePattern.Dash,
1208          thickness=0.5,
1209          smooth=Smooth.None));
1210      annotation (Icon(graphics={
1211            Ellipse(
1212              extent={{80,58},{-80,-62}},
1213              lineColor={0,0,0},
1214              lineThickness=0.5),
1215            Text(
1216              extent={{-36,-30},{40,-50}},
1217              lineColor={0,0,0},
1218              lineThickness=0.5,
1219              fillPattern=FillPattern.Solid,
```

```
1220              textString="circuit1"),
1221            Bitmap(
1222              extent={{-76,38},{70,-28}},
1223              imageSource=
1224                  "/*Put your image*/",
1225              fileName="modelica://RiskAnalysis/reoDymola.png")}), Diagram(
1226            graphics));
1227    end ReoCircuit1;
1228
1229    model ReoCircuit2
1230
1231       parameter Integer lengthFIFO1 = 100;
1232       Integer nElementsFIFO1(start=0);
1233       Integer inPointerFIFO1(start=1);
1234       Integer outPointerFIFO1(start=1);
1235       Integer moneyLossTotalFIFO1(start=0);
1236       Integer state(start=0);
1237       Real controlTime(start=0);
1238
1239       Real freeCapacity(start=0);
1240       Real usedCapacity(start=0);
1241       Integer accumulatedLosses(start=0);
1242       Integer accumulatedNumberOfLostEvents(start=0);
1243       Integer accumulatedNumberOfSuccessEvents(start=0);
1244       Integer accumulatedProfits(start=0);
1245       Integer currentBusinessValue(start=0);
1246
1247    public
1248      ServicePortOut portOut annotation (Placement(transformation(extent={{12,28},
1249                {32,48}}), iconTransformation(extent={{80,-4},{100,16}})));
1250      ServicePortIn portIn annotation (Placement(transformation(extent={{-90,-4},
1251                {-70,16}}), iconTransformation(extent={{-100,-6},{-80,14}})));
1252      Modelica.Blocks.Interfaces.RealOutput currentFIFOCapacity annotation (
1253          Placement(transformation(extent={{-26,-52},{-6,-32}}),
1254            iconTransformation(
1255            extent={{-10,-10},{10,10}},
1256            rotation=-90,
1257            origin={-20,-62})));
1258      Modelica.Blocks.Interfaces.IntegerOutput outAccumulatedLosses annotation (
1259          Placement(transformation(extent={{-70,-56},{-50,-36}}),
1260            iconTransformation(
1261            extent={{-10,-10},{10,10}},
1262            rotation=-90,
1263            origin={22,-62})));
1264
1265    equation
1266      usedCapacity =  (nElementsFIFO1 * 100) / (lengthFIFO1);
1267      freeCapacity = 100 - usedCapacity;
1268      accumulatedLosses = moneyLossTotalFIFO1;
1269      currentFIFOCapacity = usedCapacity;
1270      outAccumulatedLosses = accumulatedLosses;
1271
1272    protected
1273       Integer[lengthFIFO1] FIFO1;
1274       Integer[lengthFIFO1] price1;
1275
1276    algorithm
```

```
1277        /* The circuit receives data in input */
1278        when (pre(state) == 0 and pre(portIn.writeFlag) and (not pre(portOut.
                writeOkFlag) or (not pre(nElementsFIFO1) > 0)) and time > controlTime) then
1279          if (pre(nElementsFIFO1) < lengthFIFO1) then
1280            FIFO1[pre(inPointerFIFO1)] := pre(portIn.ID);
1281            price1[pre(inPointerFIFO1)] := pre(portIn.monetaryValue);
1282            inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
1283            nElementsFIFO1 := pre(nElementsFIFO1) + 1;
1284            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                    1;
1285            accumulatedProfits := pre(accumulatedProfits) + portIn.monetaryValue;
1286            currentBusinessValue := pre(currentBusinessValue) + portIn.monetaryValue;
1287          else
1288            moneyLossTotalFIFO1 := pre(moneyLossTotalFIFO1) + portIn.monetaryValue;
1289            accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
1290          end if;
1291          portIn.writeOkFlag := true;
1292          state := 1;
1293          controlTime := time;
1294        end when;
1295
1296        /* Writing to output */
1297        when (pre(state) == 0 and (not pre(portIn.writeFlag)) and pre(portOut.
                writeOkFlag) and (pre(nElementsFIFO1) > 0)  and time > controlTime) then
1298          /* Reading from FIFO1 */
1299          portOut.ID := FIFO1[pre(outPointerFIFO1)];
1300          portOut.monetaryValue := price1[pre(outPointerFIFO1)];
1301          outPointerFIFO1 := mod(pre(outPointerFIFO1), lengthFIFO1) + 1;
1302          nElementsFIFO1 := pre(nElementsFIFO1) - 1;
1303          currentBusinessValue := pre(currentBusinessValue) - price1[pre(outPointerFIFO1
                )];
1304          /* Setting the flags on the ports */
1305          portOut.writeFlag := true;
1306          state := 2;
1307          controlTime := time;
1308        end when;
1309
1310        /* Reading from input and writing to output */
1311        when (pre(state) == 0 and pre(portIn.writeFlag) and pre(portOut.writeOkFlag) and
                (pre(nElementsFIFO1) > 0) and time > controlTime) then
1312          if (pre(nElementsFIFO1) < lengthFIFO1) then
1313            FIFO1[pre(inPointerFIFO1)] := pre(portIn.ID);
1314            price1[pre(inPointerFIFO1)] := pre(portIn.monetaryValue);
1315            inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
1316            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                    1;
1317            accumulatedProfits := pre(accumulatedProfits) + portIn.monetaryValue;
1318            currentBusinessValue := pre(currentBusinessValue) + portIn.monetaryValue -
                    price1[pre(outPointerFIFO1)];
1319          else
1320            nElementsFIFO1 := pre(nElementsFIFO1) - 1;
1321            moneyLossTotalFIFO1 := pre(moneyLossTotalFIFO1) + portIn.monetaryValue;
1322            accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
1323            currentBusinessValue := pre(currentBusinessValue) - price1[pre(
                    outPointerFIFO1)];
1324          end if;
1325          /* Reading from FIFO1 */
```

```
1326          portOut.ID := FIFO1[pre(outPointerFIFO1)];
1327          portOut.monetaryValue := price1[pre(outPointerFIFO1)];
1328          outPointerFIFO1 := mod(pre(outPointerFIFO1), lengthFIFO1) + 1;
1329          /* Setting the flags on the ports */
1330          portIn.writeOkFlag := true;
1331          portOut.writeFlag := true;
1332          state := 3;
1333          controlTime := time;
1334        end when;
1335
1336        when (pre(state) == 1 and (not pre(portIn.writeFlag)) and time > controlTime)
               then
1337          portIn.writeOkFlag := false;
1338          state := 0;
1339          controlTime := time;
1340        end when;
1341
1342        when (pre(state) == 2 and (not pre(portOut.writeOkFlag)) and time > controlTime)
               then
1343          portOut.writeFlag := false;
1344          state := 0;
1345          controlTime := time;
1346        end when;
1347
1348        when (pre(state) == 3 and (not pre(portIn.writeFlag)) and (not pre(portOut.
             writeOkFlag)) and time > controlTime) then
1349          portIn.writeOkFlag := false;
1350          portOut.writeFlag := false;
1351          state := 0;
1352          controlTime := time;
1353        end when;
1354
1355      equation
1356        connect(portIn, portIn) annotation (Line(
1357            points={{-80,6},{-76,6},{-76,6},{-80,6}},
1358            color={0,0,0},
1359            pattern=LinePattern.Dash,
1360            thickness=0.5,
1361            smooth=Smooth.None));
1362
1363        annotation (Icon(graphics={
1364              Ellipse(
1365                extent={{80,66},{-80,-54}},
1366                lineColor={0,0,0},
1367                lineThickness=0.5),
1368              Text(
1369                extent={{-36,-22},{40,-42}},
1370                lineColor={0,0,0},
1371                lineThickness=0.5,
1372                fillPattern=FillPattern.Solid,
1373                textString="circuit2"),
1374              Bitmap(
1375                extent={{-74,46},{72,-20}},
1376                imageSource=
1377                    "/*Put your image*/",
1378                fileName="modelica://RiskAnalysis/reoDymola.png")}), Diagram(
1379              graphics));
```

```
1380    end ReoCircuit2;
1381
1382    model ReoCircuit3
1383
1384       parameter Integer lengthID1 = 10000;
1385       parameter Integer lengthID2 = 10000;
1386       parameter Integer lengthFIFO = 100;
1387
1388       Integer inPointerFIFO(start=1);
1389       Integer outPointerFIFO(start=1);
1390       Integer inPointerID1(start=1);
1391       Integer inPointerID2(start=1);
1392       Integer nElementsFIFO(start=0);
1393       Integer moneyLossTotalFIFO(start=0);
1394       Integer state(start=0);
1395       Real controlTime(start=0);
1396
1397       Real freeCapacity(start=0);
1398       Real usedCapacity(start=0);
1399       Integer accumulatedLosses(start=0);
1400       Integer accumulatedNumberOfLostEvents(start=0);
1401       Integer accumulatedNumberOfSuccessEvents(start=0);
1402       Integer accumulatedProfits(start=0);
1403       Integer currentBusinessValue(start=0);
1404
1405    public
1406      ServicePortOut portOut annotation (Placement(transformation(extent={{72,0},{92,
1407               20}}), iconTransformation(extent={{72,0},{92,20}})));
1408      ServicePortIn portIn1 annotation (Placement(transformation(extent={{-80,-36},{
1409               -60,-16}}), iconTransformation(extent={{-80,-36},{-60,-16}})));
1410      ServicePortIn portIn2 annotation (Placement(transformation(extent={{-80,32},{-60,
1411               52}}), iconTransformation(extent={{-80,32},{-60,52}})));
1412      Modelica.Blocks.Interfaces.RealOutput currentFIFOCapacity annotation (
1413          Placement(transformation(extent={{-26,-52},{-6,-32}}),
1414            iconTransformation(
1415            extent={{-10,-10},{10,10}},
1416            rotation=-90,
1417            origin={-16,-58})));
1418      Modelica.Blocks.Interfaces.IntegerOutput outAccumulatedLosses annotation (
1419          Placement(transformation(extent={{-82,-64},{-62,-44}}),
1420            iconTransformation(
1421            extent={{-10,-10},{10,10}},
1422            rotation=-90,
1423            origin={22,-58})));
1424
1425    equation
1426      usedCapacity =  (nElementsFIFO * 100) / (lengthFIFO);
1427      freeCapacity =  100 - usedCapacity;
1428      accumulatedLosses = moneyLossTotalFIFO;
1429      currentFIFOCapacity = usedCapacity;
1430      outAccumulatedLosses = accumulatedLosses;
1431
1432    protected
1433       Integer[lengthID1] ID1;
1434       Integer[lengthID2] ID2;
1435       Integer[lengthFIFO] FIFO;
1436       Integer[lengthFIFO] price;
```

```
1437
1438    algorithm
1439      /* The circuit receives data in input from portIn1 */
1440      when (pre(state) == 0 and pre(portIn1.writeFlag) and (not pre(portIn2.writeFlag))
              and ((not pre(portOut.writeOkFlag)) or (not pre(nElementsFIFO) > 0)) and time
              > controlTime) then
1441        portIn1.writeOkFlag := true;
1442        if (Modelica.Math.Vectors.find(pre(portIn1.ID),ID2) == 0 and pre(nElementsFIFO)
              < lengthFIFO) then
1443          ID1[pre(inPointerID1)] := portIn1.ID;
1444          inPointerID1 := mod(pre(inPointerID1), lengthID1) + 1;
1445          FIFO[pre(inPointerFIFO)] := portIn1.ID;
1446          price[pre(inPointerFIFO)] := portIn1.monetaryValue;
1447          inPointerFIFO := mod(pre(inPointerFIFO), lengthFIFO) + 1;
1448          nElementsFIFO := pre(nElementsFIFO) + 1;
1449          accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                1;
1450          accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
1451          currentBusinessValue := pre(currentBusinessValue) + pre(portIn1.
                monetaryValue);
1452        elseif (Modelica.Math.Vectors.find(pre(portIn1.ID),ID2) == 0 and (not (pre(
              nElementsFIFO) < lengthFIFO))) then
1453          ID1[pre(inPointerID1)] := portIn1.ID;
1454          inPointerID1 := mod(pre(inPointerID1), lengthID1) + 1;
1455          moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn1.monetaryValue;
1456          accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
1457        end if;
1458        state := 1;
1459        controlTime := time;
1460      end when;
1461
1462      /* The circuit receives data in input from portIn2 */
1463      when (pre(state) == 0 and (not pre(portIn1.writeFlag)) and pre(portIn2.writeFlag)
              and ((not pre(portOut.writeOkFlag)) or (not pre(nElementsFIFO) > 0)) and time
              > controlTime) then
1464        portIn2.writeOkFlag := true;
1465        if (Modelica.Math.Vectors.find(pre(portIn2.ID),ID1) == 0 and pre(nElementsFIFO)
              < lengthFIFO) then
1466          ID2[pre(inPointerID2)] := portIn2.ID;
1467          inPointerID2 := mod(pre(inPointerID2), lengthID2) + 1;
1468          FIFO[pre(inPointerFIFO)] := portIn2.ID;
1469          price[pre(inPointerFIFO)] := portIn2.monetaryValue;
1470          inPointerFIFO := mod(pre(inPointerFIFO), lengthFIFO) + 1;
1471          nElementsFIFO := pre(nElementsFIFO) + 1;
1472          accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                1;
1473          accumulatedProfits := pre(accumulatedProfits) + portIn2.monetaryValue;
1474          currentBusinessValue := pre(currentBusinessValue) + portIn2.monetaryValue;
1475        elseif (Modelica.Math.Vectors.find(pre(portIn2.ID),ID1) == 0 and (not (pre(
              nElementsFIFO) < lengthFIFO))) then
1476          ID2[pre(inPointerID2)] := portIn2.ID;
1477          inPointerID2 := mod(pre(inPointerID2), lengthID2) + 1;
1478          moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn2.monetaryValue;
1479          accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
1480        end if;
1481        state := 2;
1482        controlTime := time;
```

```
1483      end when;
1484
1485      /* The circuit receives data in input from portIn1 and from portIn2 */
1486      when (pre(state) == 0 and pre(portIn1.writeFlag) and pre(portIn2.writeFlag) and
              ((not pre(portOut.writeOkFlag)) or (not pre(nElementsFIFO) > 0)) and time >
              controlTime) then
1487        if (portIn1.ID == portIn2.ID and pre(nElementsFIFO) < lengthFIFO) then
1488          FIFO[pre(inPointerFIFO)] := portIn1.ID;
1489          price[pre(inPointerFIFO)] := portIn1.monetaryValue;
1490          inPointerFIFO := mod(pre(inPointerFIFO), lengthFIFO) + 1;
1491          nElementsFIFO := pre(nElementsFIFO) + 1;
1492          ID1[pre(inPointerID1)] := portIn1.ID;
1493          inPointerID1 := mod(pre(inPointerID1), lengthID1) + 1;
1494          accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                  1;
1495          accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
1496          currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue;
1497        elseif (portIn1.ID == portIn2.ID and (not pre(nElementsFIFO) < lengthFIFO))
              then
1498          moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn1.monetaryValue;
1499          accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
1500        else
1501          if ((Modelica.Math.Vectors.find(pre(portIn1.ID),ID2) == 0) and (Modelica.
                  Math.Vectors.find(pre(portIn2.ID),ID1)) <> 0) then
1502            ID1[pre(inPointerID1)] := portIn1.ID;
1503            inPointerID1 := mod(pre(inPointerID1), lengthID1) + 1;
1504            if (pre(nElementsFIFO) < lengthFIFO) then
1505              FIFO[pre(inPointerFIFO)] := portIn1.ID;
1506              price[pre(inPointerFIFO)] := portIn1.monetaryValue;
1507              inPointerFIFO := mod(pre(inPointerFIFO), lengthFIFO + 1);
1508              nElementsFIFO := pre(nElementsFIFO) + 1;
1509              accumulatedNumberOfSuccessEvents := pre(
                      accumulatedNumberOfSuccessEvents) + 1;
1510              accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
1511              currentBusinessValue := pre(currentBusinessValue) + portIn1.
                      monetaryValue;
1512            else
1513              moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn1.monetaryValue;
1514              accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) +
                      1;
1515            end if;
1516          end if;
1517          if ((Modelica.Math.Vectors.find(pre(portIn1.ID),ID2) <> 0) and (Modelica.
                  Math.Vectors.find(pre(portIn2.ID),ID1) == 0)) then
1518            ID2[pre(inPointerID2)] := portIn2.ID;
1519            inPointerID2 := mod(pre(inPointerID2), lengthID2) + 1;
1520            if (pre(nElementsFIFO) < lengthFIFO) then
1521              FIFO[pre(inPointerFIFO)] := portIn2.ID;
1522              price[pre(inPointerFIFO)] := portIn2.monetaryValue;
1523              inPointerFIFO := mod(pre(inPointerFIFO), lengthFIFO) + 1;
1524              nElementsFIFO := pre(nElementsFIFO) + 1;
1525              accumulatedNumberOfSuccessEvents := pre(
                      accumulatedNumberOfSuccessEvents) + 1;
1526              accumulatedProfits := pre(accumulatedProfits) + portIn2.monetaryValue;
1527              currentBusinessValue := pre(currentBusinessValue) + portIn2.
                      monetaryValue;
1528            else
```

```
1529                moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn2.monetaryValue;
1530                accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) +
                        1;
1531              end if;
1532            end if;
1533            if ((Modelica.Math.Vectors.find(pre(portIn1.ID),ID2) == 0) and (Modelica.
                    Math.Vectors.find(pre(portIn2.ID),ID1) == 0) and (pre(nElementsFIFO) +1 <
                     lengthFIFO)) then
1534              ID1[pre(inPointerID1)] := portIn1.ID;
1535              inPointerID1 := mod(pre(inPointerID1) + 1, lengthID1);
1536              ID2[pre(inPointerID2)] := portIn2.ID;
1537              inPointerID2 := mod(pre(inPointerID2) + 1, lengthID2);
1538              FIFO[pre(inPointerFIFO)] := portIn1.ID;
1539              price[pre(inPointerFIFO)] := portIn1.monetaryValue;
1540              FIFO[pre(inPointerFIFO)+1] := portIn2.ID;
1541              price[pre(inPointerFIFO)+1] := portIn2.monetaryValue;
1542              if (pre(nElementsFIFO) == (lengthFIFO -2)) then
1543                inPointerFIFO := 1;
1544              else
1545                inPointerFIFO := mod(pre(inPointerFIFO), lengthFIFO) + 2;
1546              end if;
1547              nElementsFIFO := pre(nElementsFIFO) + 2;
1548              accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                      + 2;
1549              accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue +
                      portIn2.monetaryValue;
1550              currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                      + portIn2.monetaryValue;
1551            elseif ((Modelica.Math.Vectors.find(pre(portIn1.ID),ID2) == 0) and (
                    Modelica.Math.Vectors.find(pre(portIn2.ID),ID1) == 0) and pre(
                    nElementsFIFO) < lengthFIFO) then
1552              ID1[pre(inPointerID1)] := portIn1.ID;
1553              inPointerID1 := mod(pre(inPointerID1), lengthID1) + 1;
1554              ID2[pre(inPointerID2)] := portIn2.ID;
1555              inPointerID2 := mod(pre(inPointerID2), lengthID2) + 1;
1556              FIFO[pre(inPointerFIFO)] := portIn1.ID;
1557              price[pre(inPointerFIFO)] := portIn1.monetaryValue;
1558              inPointerFIFO := mod(pre(inPointerFIFO), lengthFIFO) + 1;
1559              nElementsFIFO := pre(nElementsFIFO) + 1;
1560              moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn2.monetaryValue;
1561              accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
1562              accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                      + 1;
1563              accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
1564              currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                      ;
1565            elseif ((Modelica.Math.Vectors.find(pre(portIn1.ID),ID2) == 0) and (
                    Modelica.Math.Vectors.find(pre(portIn2.ID),ID1) == 0) and (not pre(
                    nElementsFIFO) < lengthFIFO)) then
1566              ID1[pre(inPointerID1)] := portIn1.ID;
1567              inPointerID1 := mod(pre(inPointerID1), lengthID1) + 1;
1568              ID2[pre(inPointerID2)] := portIn2.ID;
1569              inPointerID2 := mod(pre(inPointerID2), lengthID2) + 1;
1570              moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn1.monetaryValue +
                      portIn2.monetaryValue;
1571              accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 2;
1572            end if;
```

```
1573        end if;
1574        portIn1.writeOkFlag := true;
1575        portIn2.writeOkFlag := true;
1576        state := 3;
1577        controlTime := time;
1578      end when;
1579
1580      /* The circuit receives data in input from portIn1 and writes to portOut */
1581      when (pre(state) == 0 and pre(portIn1.writeFlag) and (not pre(portIn2.writeFlag))
              and pre(portOut.writeOkFlag) and pre(nElementsFIFO) > 0 and time >
             controlTime) then
1582        portIn1.writeOkFlag := true;
1583
1584        if ((Modelica.Math.Vectors.find(pre(portIn1.ID),ID2) == 0) and pre(
              nElementsFIFO) < lengthFIFO) then
1585          ID1[pre(inPointerID1)] := portIn1.ID;
1586          inPointerID1 := mod(pre(inPointerID1), lengthID1) + 1;
1587          FIFO[pre(inPointerFIFO)] := portIn1.ID;
1588          price[pre(inPointerFIFO)] := portIn1.monetaryValue;
1589          inPointerFIFO := mod(pre(inPointerFIFO), lengthFIFO) + 1;
1590          accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                  1;
1591          accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
1592          currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue -
                  price[pre(outPointerFIFO)];
1593        elseif ((Modelica.Math.Vectors.find(pre(portIn1.ID),ID2) == 0) and (not pre(
              nElementsFIFO) < lengthFIFO)) then
1594          ID1[pre(inPointerID1)] := portIn1.ID;
1595          inPointerID1 := mod(pre(inPointerID1), lengthID1) + 1;
1596          moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn1.monetaryValue;
1597          accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
1598          nElementsFIFO := pre(nElementsFIFO) - 1;
1599          currentBusinessValue := pre(currentBusinessValue) - price[pre(outPointerFIFO
                  )];
1600        else
1601          nElementsFIFO := pre(nElementsFIFO) - 1;
1602        end if;
1603        // Writing the buffers for the output
1604        portOut.ID := FIFO[pre(outPointerFIFO)];
1605        portOut.monetaryValue := price[pre(outPointerFIFO)];
1606        outPointerFIFO := mod(pre(outPointerFIFO), lengthFIFO) + 1;
1607        portOut.writeFlag:= true;
1608        state := 4;
1609        controlTime := time;
1610      end when;
1611
1612      /* The circuit receives data in input from portIn2 and writes to portOut */
1613      when (pre(state) == 0 and (not pre(portIn1.writeFlag)) and pre(portIn2.writeFlag)
              and pre(portOut.writeOkFlag) and pre(nElementsFIFO) > 0 and time >
             controlTime) then
1614        portIn2.writeOkFlag := true;
1615        if ((Modelica.Math.Vectors.find(pre(portIn2.ID),ID1) == 0) and (pre(
              nElementsFIFO) < lengthFIFO)) then
1616          ID2[pre(inPointerID2)] := portIn2.ID;
1617          inPointerID2 := mod(pre(inPointerID2), lengthID2) + 1;
1618          FIFO[pre(inPointerFIFO)] := portIn2.ID;
1619          price[pre(inPointerFIFO)] := portIn2.monetaryValue;
```

```
1620            inPointerFIFO := mod(pre(inPointerFIFO), lengthFIFO) + 1;
1621            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                    1;
1622            accumulatedProfits := pre(accumulatedProfits) + portIn2.monetaryValue;
1623            currentBusinessValue := pre(currentBusinessValue) + portIn2.monetaryValue -
                    price[pre(outPointerFIFO)];
1624          elseif ((Modelica.Math.Vectors.find(pre(portIn2.ID),ID1) == 0) and (not (pre(
                  nElementsFIFO) < lengthFIFO))) then
1625            ID2[pre(inPointerID2)] := portIn2.ID;
1626            inPointerID2 := mod(pre(inPointerID2), lengthID2) + 1;
1627            moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn2.monetaryValue;
1628            accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
1629            nElementsFIFO := pre(nElementsFIFO) - 1;
1630            currentBusinessValue := pre(currentBusinessValue) - price[pre(outPointerFIFO
                    )];
1631          else
1632            nElementsFIFO := pre(nElementsFIFO) - 1;
1633          end if;
1634          // Writing the buffers for the output
1635          portOut.ID := FIFO[pre(outPointerFIFO)];
1636          portOut.monetaryValue := price[pre(outPointerFIFO)];
1637          outPointerFIFO := mod(pre(outPointerFIFO), lengthFIFO) + 1;
1638          portOut.writeFlag:= true;
1639          state := 5;
1640          controlTime := time;
1641        end when;
1642
1643        /* The circuit  writes to portOut */
1644        when (pre(state) == 0 and (not pre(portIn1.writeFlag)) and (not pre(portIn2.
                writeFlag)) and pre(portOut.writeOkFlag) and pre(nElementsFIFO) > 0 and time >
                 controlTime) then
1645          portOut.ID := FIFO[pre(outPointerFIFO)];
1646          portOut.monetaryValue := price[pre(outPointerFIFO)];
1647          outPointerFIFO := mod(pre(outPointerFIFO), lengthFIFO) + 1;
1648          nElementsFIFO := pre(nElementsFIFO) - 1;
1649          currentBusinessValue := pre(currentBusinessValue) - price[pre(outPointerFIFO)];
1650          portOut.writeFlag:= true;
1651          state := 6;
1652          controlTime := time;
1653        end when;
1654
1655        /* The circuit receives data in input from portIn1 and from portIn2 and writes to
                 portOut */
1656        when (pre(state) == 0 and pre(portIn1.writeFlag) and pre(portIn2.writeFlag) and
                pre(portOut.writeOkFlag) and pre(nElementsFIFO) > 0 and time > controlTime)
                then
1657          if (portIn1.ID == portIn2.ID and pre(nElementsFIFO) < lengthFIFO) then
1658            FIFO[pre(inPointerFIFO)] := portIn1.ID;
1659            price[pre(inPointerFIFO)] := portIn1.monetaryValue;
1660            inPointerFIFO := mod(pre(inPointerFIFO), lengthFIFO) + 1;
1661            nElementsFIFO := pre(nElementsFIFO) + 1;
1662            ID1[pre(inPointerID1)] := portIn1.ID;
1663            inPointerID1 := mod(pre(inPointerID1), lengthID1) + 1;
1664            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                    1;
1665            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
```

```
1666            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue -
                    price[pre(outPointerFIFO)];
1667        elseif (portIn1.ID == portIn2.ID and (not pre(nElementsFIFO) < lengthFIFO))
                then
1668            moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn1.monetaryValue;
1669            accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
1670            currentBusinessValue := pre(currentBusinessValue) - price[pre(outPointerFIFO
                    )];
1671        else
1672            if ((Modelica.Math.Vectors.find(pre(portIn1.ID),ID2) == 0) and (Modelica.
                    Math.Vectors.find(pre(portIn2.ID),ID1)) <> 0) then
1673                ID1[pre(inPointerID1)] := portIn1.ID;
1674                inPointerID1 := mod(pre(inPointerID1), lengthID1) + 1;
1675                if (pre(nElementsFIFO) < lengthFIFO) then
1676                    FIFO[pre(inPointerFIFO)] := portIn1.ID;
1677                    price[pre(inPointerFIFO)] := portIn1.monetaryValue;
1678                    inPointerFIFO := mod(pre(inPointerFIFO), lengthFIFO) + 1;
1679                    accumulatedNumberOfSuccessEvents := pre(
                            accumulatedNumberOfSuccessEvents) + 1;
1680                    accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
1681                    currentBusinessValue := pre(currentBusinessValue) + portIn1.
                            monetaryValue - price[pre(outPointerFIFO)];
1682                else
1683                    moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn1.monetaryValue;
1684                    accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) +
                            1;
1685                    currentBusinessValue := pre(currentBusinessValue) - price[pre(
                            outPointerFIFO)];
1686                end if;
1687            end if;
1688            if ((Modelica.Math.Vectors.find(pre(portIn1.ID),ID2) <> 0) and (Modelica.
                    Math.Vectors.find(pre(portIn2.ID),ID1) == 0)) then
1689                ID2[pre(inPointerID2)] := portIn2.ID;
1690                inPointerID2 := mod(pre(inPointerID2), lengthID2) + 1;
1691                if (pre(nElementsFIFO) < lengthFIFO) then
1692                    FIFO[pre(inPointerFIFO)] := portIn2.ID;
1693                    price[pre(inPointerFIFO)] := portIn2.monetaryValue;
1694                    inPointerFIFO := mod(pre(inPointerFIFO), lengthFIFO) + 1;
1695                    accumulatedNumberOfSuccessEvents := pre(
                            accumulatedNumberOfSuccessEvents) + 1;
1696                    accumulatedProfits := pre(accumulatedProfits) + portIn2.monetaryValue;
1697                    currentBusinessValue := pre(currentBusinessValue) + portIn2.
                            monetaryValue - price[pre(outPointerFIFO)];
1698                else
1699                    moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn2.monetaryValue;
1700                    accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) +
                            1;
1701                    currentBusinessValue := pre(currentBusinessValue) - price[pre(
                            outPointerFIFO)];
1702                end if;
1703            end if;
1704            if ((Modelica.Math.Vectors.find(pre(portIn1.ID),ID2) <> 0) and (Modelica.
                    Math.Vectors.find(pre(portIn2.ID),ID1) <> 0)) then
1705                nElementsFIFO := pre(nElementsFIFO) - 1;
1706            end if;
1707            if ((Modelica.Math.Vectors.find(pre(portIn1.ID),ID2) == 0) and (Modelica.
                    Math.Vectors.find(pre(portIn2.ID),ID1) == 0) and (pre(nElementsFIFO) +1 <
```

```
                lengthFIFO)) then
1708            ID1[pre(inPointerID1)] := portIn1.ID;
1709            inPointerID1 := mod(pre(inPointerID1), lengthID1) + 1;
1710            ID2[pre(inPointerID2)] := portIn2.ID;
1711            inPointerID2 := mod(pre(inPointerID2), lengthID2) + 1;
1712            FIFO[pre(inPointerFIFO)] := portIn1.ID;
1713            price[pre(inPointerFIFO)] := portIn1.monetaryValue;
1714            FIFO[pre(inPointerFIFO)+1] := portIn2.ID;
1715            price[pre(inPointerFIFO)+1] := portIn2.monetaryValue;
1716            if (pre(nElementsFIFO) == (lengthFIFO -2)) then
1717              inPointerFIFO := 1;
1718            else
1719              inPointerFIFO := mod(pre(inPointerFIFO), lengthFIFO) + 2;
1720            end if;
1721            nElementsFIFO := pre(nElementsFIFO) + 1;
1722            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                  + 2;
1723            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue +
                  portIn2.monetaryValue;
1724            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                  + portIn2.monetaryValue - price[pre(outPointerFIFO)];
1725          elseif ((Modelica.Math.Vectors.find(pre(portIn1.ID),ID2) == 0) and (
                Modelica.Math.Vectors.find(pre(portIn2.ID),ID1) == 0) and pre(
                nElementsFIFO) < lengthFIFO) then
1726            ID1[pre(inPointerID1)] := portIn1.ID;
1727            inPointerID1 := mod(pre(inPointerID1), lengthID1) + 1;
1728            ID2[pre(inPointerID2)] := portIn2.ID;
1729            inPointerID2 := mod(pre(inPointerID2), lengthID2) + 1;
1730            FIFO[pre(inPointerFIFO)] := portIn1.ID;
1731            price[pre(inPointerFIFO)] := portIn1.monetaryValue;
1732            inPointerFIFO := mod(pre(inPointerFIFO), lengthFIFO) + 1;
1733            moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn2.monetaryValue;
1734            accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
1735            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                  + 1;
1736            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
1737            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                  - price[pre(outPointerFIFO)];
1738          elseif ((Modelica.Math.Vectors.find(pre(portIn1.ID),ID2) == 0) and (Modelica
                .Math.Vectors.find(pre(portIn2.ID),ID1) == 0) and (not pre(nElementsFIFO)
                < lengthFIFO)) then
1739            ID1[pre(inPointerID1)] := portIn1.ID;
1740            inPointerID1 := mod(pre(inPointerID1), lengthID1) + 1;
1741            ID2[pre(inPointerID2)] := portIn2.ID;
1742            inPointerID2 := mod(pre(inPointerID2) + 1, lengthID2);
1743            moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn1.monetaryValue +
                  portIn2.monetaryValue;
1744            accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 2;
1745            nElementsFIFO := pre(nElementsFIFO) - 1;
1746          end if;
1747        end if;
1748        // Writing in the buffers for the output
1749        portOut.ID := FIFO[pre(outPointerFIFO)];
1750        portOut.monetaryValue := price[pre(outPointerFIFO)];
1751        outPointerFIFO := mod(pre(outPointerFIFO), lengthFIFO) + 1;
1752
1753        portOut.writeFlag:= true;
```

```
1754          portIn1.writeOkFlag := true;
1755          portIn2.writeOkFlag := true;
1756          state := 7;
1757          controlTime := time;
1758        end when;
1759
1760        when (pre(state) == 1 and (not pre(portIn1.writeFlag)) and time > controlTime)
                then
1761          portIn1.writeOkFlag := false;
1762          state := 0;
1763          controlTime := time;
1764        end when;
1765
1766        when (pre(state) == 2 and (not pre(portIn2.writeFlag)) and time > controlTime)
                then
1767          portIn2.writeOkFlag := false;
1768          state := 0;
1769          controlTime := time;
1770           end when;
1771
1772         when (pre(state) == 3 and (not pre(portIn1.writeFlag)) and (not pre(portIn2.
               writeFlag)) and time > controlTime) then
1773          portIn1.writeOkFlag := false;
1774          portIn2.writeOkFlag := false;
1775          state := 0;
1776          controlTime := time;
1777         end when;
1778
1779         when (pre(state) == 4 and (not pre(portIn1.writeFlag)) and (not pre(portOut.
               writeOkFlag)) and time > controlTime) then
1780          portIn1.writeOkFlag := false;
1781          portOut.writeFlag := false;
1782          state := 0;
1783          controlTime := time;
1784         end when;
1785
1786         when (pre(state) == 5 and (not pre(portIn2.writeFlag)) and (not pre(portOut.
               writeOkFlag)) and time > controlTime) then
1787          portIn2.writeOkFlag := false;
1788          portOut.writeFlag := false;
1789          state := 0;
1790          controlTime := time;
1791         end when;
1792
1793         when (pre(state) == 6 and (not pre(portOut.writeOkFlag)) and time > controlTime)
                then
1794          portOut.writeFlag := false;
1795          state := 0;
1796          controlTime := time;
1797         end when;
1798
1799         when (pre(state) == 7 and (not pre(portIn1.writeFlag)) and (not pre(portIn2.
               writeFlag)) and (not pre(portOut.writeOkFlag)) and time > controlTime) then
1800          portIn1.writeOkFlag := false;
1801          portIn2.writeOkFlag := false;
1802          portOut.writeFlag := false;
1803          state := 0;
```

```
1804          controlTime := time;
1805        end when;
1806
1807      equation
1808        connect(portIn1, portIn1) annotation (Line(
1809            points={{-70,-26},{-68,-26},{-68,-26},{-70,-26}},
1810            color={0,0,0},
1811            pattern=LinePattern.Dash,
1812            thickness=0.5,
1813            smooth=Smooth.None));
1814          annotation (Icon(graphics={
1815              Ellipse(
1816                extent={{82,70},{-78,-50}},
1817                lineColor={0,0,0},
1818                lineThickness=0.5),
1819              Text(
1820                extent={{-34,-18},{42,-38}},
1821                lineColor={0,0,0},
1822                lineThickness=0.5,
1823                fillPattern=FillPattern.Solid,
1824                textString="circuit3"),
1825              Bitmap(
1826                extent={{-70,50},{76,-16}},
1827                imageSource=
1828                    "/*Put your image*/",
1829                fileName="modelica://RiskAnalysis/reoDymola.png")}), Diagram(graphics));
1830    end ReoCircuit3;
1831
1832    model ReoCircuit4
1833
1834        parameter Integer lengthID = 10000;
1835        parameter Integer lengthFIFO1 = 100;
1836        parameter Integer lengthFIFO2 = 100;
1837        Integer nElementsFIFO1(start=0);
1838        Integer nElementsFIFO2(start=0);
1839        Integer inPointerFIFO1(start=1);
1840        Integer inPointerFIFO2(start=1);
1841        Integer outPointerFIFO1(start=1);
1842        Integer outPointerFIFO2(start=1);
1843        Integer moneyLossTotalFIFO(start=0);
1844        Integer state(start=0);
1845        Real controlTime(start=0);
1846
1847        Real freeCapacity(start=0);
1848        Real usedCapacity(start=0);
1849        Integer accumulatedLosses(start=0);
1850        Integer accumulatedNumberOfLostEvents(start=0);
1851        Integer accumulatedNumberOfSuccessEvents(start=0);
1852        Integer accumulatedProfits(start=0);
1853        Integer currentBusinessValue(start=0);
1854
1855        Modelon.Blocks.Sources.RandomNumbers.uniformRNG uniformRNG(nout=1,
1856           samplePeriod=0.1,
1857          firstSeed=Modelon.Math.RandomNumbers.Seed(
1858                x=111,
1859                y=88,
1860                z=7))
```

```
1861        annotation (Placement(transformation(extent={{-38,-44},{-18,-24}})));
1862
1863    public
1864      ServicePortOut portOut1 annotation (Placement(transformation(extent={{72,22},{
1865              92,42}}), iconTransformation(extent={{72,22},{92,42}})));
1866      ServicePortOut portOut2 annotation (Placement(transformation(extent={{72,-48},
1867              {92,-28}}), iconTransformation(extent={{72,-48},{92,-28}})));
1868      ServicePortIn portIn annotation (Placement(transformation(extent={{-96,-12},{-76,
1869              8}}), iconTransformation(extent={{-96,-12},{-76,8}})));
1870
1871    protected
1872       Integer[lengthID] IDVector;
1873       Integer[lengthFIFO1] FIFO1;
1874       Integer[lengthFIFO2] FIFO2;
1875       Integer[lengthFIFO1] price1;
1876       Integer[lengthFIFO2] price2;
1877
1878    public
1879      Modelica.Blocks.Interfaces.RealOutput currentFIFOCapacity annotation (
1880          Placement(transformation(extent={{-6,-32},{14,-12}}),
1881            iconTransformation(
1882            extent={{-10,-10},{10,10}},
1883            rotation=-90,
1884            origin={-14,-70})));
1885      Modelica.Blocks.Interfaces.IntegerOutput outAccumulatedLosses annotation (
1886          Placement(transformation(extent={{-72,-54},{-52,-34}}),
1887            iconTransformation(
1888            extent={{-10,-10},{10,10}},
1889            rotation=-90,
1890            origin={22,-70})));
1891
1892    equation
1893      usedCapacity = ((nElementsFIFO1 + nElementsFIFO1) * 100) / (lengthFIFO1 +
             lengthFIFO2);
1894      freeCapacity = 100 - usedCapacity;
1895      accumulatedLosses = moneyLossTotalFIFO;
1896      currentFIFOCapacity = usedCapacity;
1897      outAccumulatedLosses = accumulatedLosses;
1898
1899    algorithm
1900       /* The circuit receives data from Client, while both portService1 and
              portService2 are not ready to read*/
1901       when (pre(state) == 0 and pre(portIn.writeFlag) and (not pre(portOut1.
             writeOkFlag) or (not pre(nElementsFIFO1) > 0)) and (not pre(portOut2.
             writeOkFlag) or (not pre(nElementsFIFO2) > 0)) and time > controlTime) then
1902         if (IDVector[pre(portIn.ID)] == 0) then
1903           if ((pre(nElementsFIFO1) < lengthFIFO1) and (not pre(nElementsFIFO2) <
                 lengthFIFO2)) then
1904             FIFO1[pre(inPointerFIFO1)] := portIn.ID;
1905             price1[pre(inPointerFIFO1)] := portIn.monetaryValue;
1906             inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
1907             nElementsFIFO1 := pre(nElementsFIFO1) + 1;
1908             accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 1;
1909             accumulatedProfits := pre(accumulatedProfits) + portIn.monetaryValue;
1910             currentBusinessValue := pre(currentBusinessValue) + portIn.monetaryValue;
1911             IDVector[pre(portIn.ID)] := 1;
```

```
1912            end if;
1913            if (pre(nElementsFIFO2) < lengthFIFO2 and (not pre(nElementsFIFO1) <
                    lengthFIFO1)) then
1914              FIFO2[pre(inPointerFIFO2)] := portIn.ID;
1915              price2[pre(inPointerFIFO2)] := portIn.monetaryValue;
1916              inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
1917              nElementsFIFO2 := pre(nElementsFIFO2) + 1;
1918              accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                      + 1;
1919              accumulatedProfits := pre(accumulatedProfits) + portIn.monetaryValue;
1920              currentBusinessValue := pre(currentBusinessValue) + portIn.monetaryValue;
1921              IDVector[pre(portIn.ID)] := 2;
1922            end if;
1923            if ((pre(nElementsFIFO1) < lengthFIFO1) and (pre(nElementsFIFO2) <
                    lengthFIFO2) and uniformRNG.outPort[1] >= 0.5) then
1924              FIFO1[pre(inPointerFIFO1)] := portIn.ID;
1925              price1[pre(inPointerFIFO1)] := portIn.monetaryValue;
1926              inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
1927              nElementsFIFO1 := pre(nElementsFIFO1) + 1;
1928              accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                      + 1;
1929              accumulatedProfits := pre(accumulatedProfits) + portIn.monetaryValue;
1930              currentBusinessValue := pre(currentBusinessValue) + portIn.monetaryValue;
1931              IDVector[pre(portIn.ID)] := 1;
1932            end if;
1933            if ((pre(nElementsFIFO2) < lengthFIFO2) and (pre(nElementsFIFO1) <
                    lengthFIFO1) and (not uniformRNG.outPort[1] >= 0.5)) then
1934              FIFO2[pre(inPointerFIFO2)] := portIn.ID;
1935              price2[pre(inPointerFIFO2)] := portIn.monetaryValue;
1936              inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
1937              nElementsFIFO2 := pre(nElementsFIFO2) + 1;
1938              accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                      + 1;
1939              accumulatedProfits := pre(accumulatedProfits) + portIn.monetaryValue;
1940              currentBusinessValue := pre(currentBusinessValue) + portIn.monetaryValue;
1941              IDVector[pre(portIn.ID)] := 2;
1942            end if;
1943            if (not (pre(nElementsFIFO1) < lengthFIFO1) and (not pre(nElementsFIFO2) <
                    lengthFIFO2)) then
1944              moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn.monetaryValue;
1945              accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
1946            end if;
1947          elseif (IDVector[pre(portIn.ID)] == 1) then
1948            if (pre(nElementsFIFO1) < lengthFIFO1) then
1949              FIFO1[pre(inPointerFIFO1)] := portIn.ID;
1950              price1[pre(inPointerFIFO1)] := portIn.monetaryValue;
1951              inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
1952              nElementsFIFO1 := pre(nElementsFIFO1) + 1;
1953              accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                      + 1;
1954              accumulatedProfits := pre(accumulatedProfits) + portIn.monetaryValue;
1955              currentBusinessValue := pre(currentBusinessValue) + portIn.monetaryValue;
1956            else
1957              moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn.monetaryValue;
1958              accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
1959            end if;
1960          elseif (IDVector[pre(portIn.ID)] == 2) then
```

```
1961            if (pre(nElementsFIFO2) < lengthFIFO2) then
1962              FIFO2[pre(inPointerFIFO2)] := portIn.ID;
1963              price2[pre(inPointerFIFO2)] := portIn.monetaryValue;
1964              inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
1965              nElementsFIFO2 := pre(nElementsFIFO2) + 1;
1966              accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                      + 1;
1967              accumulatedProfits := pre(accumulatedProfits) + portIn.monetaryValue;
1968              currentBusinessValue := pre(currentBusinessValue) + portIn.monetaryValue;
1969            else
1970              moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn.monetaryValue;
1971              accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
1972            end if;
1973          end if;
1974          portIn.writeOkFlag := true;
1975          state := 1;
1976          controlTime := time;
1977        end when;
1978
1979        /* No input from the Client but portService1 and portService2 read at the same
               time */
1980        when (pre(state) == 0 and (not pre(portIn.writeFlag)) and pre(portOut1.
               writeOkFlag) and (pre(nElementsFIFO1) > 0) and pre(portOut2.writeOkFlag) and
               (pre(nElementsFIFO2) > 0) and time > controlTime) then
1981          /* Reading from FIFO1 */
1982          portOut1.ID := FIFO1[pre(outPointerFIFO1)];
1983          portOut1.monetaryValue := price1[pre(outPointerFIFO1)];
1984          outPointerFIFO1 := mod(pre(outPointerFIFO1), lengthFIFO1) + 1;
1985          nElementsFIFO1 := pre(nElementsFIFO1) - 1;
1986          /* Reading from FIFO2 */
1987          portOut2.ID := FIFO2[pre(outPointerFIFO2)];
1988          portOut2.monetaryValue := price2[pre(outPointerFIFO2)];
1989          outPointerFIFO2 := mod(pre(outPointerFIFO2), lengthFIFO2) + 1;
1990          nElementsFIFO2 := pre(nElementsFIFO2) - 1;
1991          /* Setting the flags on the ports */
1992          currentBusinessValue := pre(currentBusinessValue) - price1[pre(outPointerFIFO1
                   )] - price2[pre(outPointerFIFO2)];
1993          portOut1.writeFlag := true;
1994          portOut2.writeFlag := true;
1995          state := 2;
1996          controlTime := time;
1997        end when;
1998
1999        /* Reading from Client ad writing to portService1 at the same time */
2000        when (pre(state) == 0 and pre(portIn.writeFlag) and pre(portOut1.writeOkFlag)
               and (pre(nElementsFIFO1) > 0) and ((not pre(portOut2.writeOkFlag)) or (not
               pre(nElementsFIFO2) > 0)) and time > controlTime) then
2001          if (IDVector[pre(portIn.ID)] == 0) then
2002            if ((pre(nElementsFIFO1) < lengthFIFO1) and (not pre(nElementsFIFO2) <
                     lengthFIFO2)) then
2003              FIFO1[pre(inPointerFIFO1)] := portIn.ID;
2004              price1[pre(inPointerFIFO1)] := portIn.monetaryValue;
2005              inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
2006              accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                      + 1;
2007              accumulatedProfits := pre(accumulatedProfits) + portIn.monetaryValue;
```

```
2008            currentBusinessValue := pre(currentBusinessValue) + portIn.monetaryValue -
                    price1[pre(outPointerFIFO1)];
2009            IDVector[pre(portIn.ID)] := 1;
2010          end if;
2011          if (pre(nElementsFIFO2) < lengthFIFO2 and (not pre(nElementsFIFO1) <
                  lengthFIFO1)) then
2012            FIFO2[pre(inPointerFIFO2)] := portIn.ID;
2013            price2[pre(inPointerFIFO2)] := portIn.monetaryValue;
2014            inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
2015            nElementsFIFO2 := pre(nElementsFIFO2) + 1;
2016            nElementsFIFO1 := pre(nElementsFIFO1) - 1;
2017            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 1;
2018            accumulatedProfits := pre(accumulatedProfits) + portIn.monetaryValue;
2019            currentBusinessValue := pre(currentBusinessValue) + portIn.monetaryValue -
                    price1[pre(outPointerFIFO1)];
2020            IDVector[pre(portIn.ID)] := 2;
2021          end if;
2022          if ((pre(nElementsFIFO1) < lengthFIFO1) and (pre(nElementsFIFO2) <
                  lengthFIFO2) and uniformRNG.outPort[1] >= 0.5) then
2023            FIFO1[pre(inPointerFIFO1)] := portIn.ID;
2024            price1[pre(inPointerFIFO1)] := portIn.monetaryValue;
2025            inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
2026            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 1;
2027            accumulatedProfits := pre(accumulatedProfits) + portIn.monetaryValue;
2028            currentBusinessValue := pre(currentBusinessValue) + portIn.monetaryValue -
                    price1[pre(outPointerFIFO1)];
2029            IDVector[pre(portIn.ID)] := 1;
2030          end if;
2031          if ((pre(nElementsFIFO2) < lengthFIFO2) and (pre(nElementsFIFO1) <
                  lengthFIFO1) and (not uniformRNG.outPort[1] >= 0.5)) then
2032            FIFO2[pre(inPointerFIFO2)] := portIn.ID;
2033            price2[pre(inPointerFIFO2)] := portIn.monetaryValue;
2034            inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
2035            nElementsFIFO2 := pre(nElementsFIFO2) + 1;
2036            nElementsFIFO1 := pre(nElementsFIFO1) - 1;
2037            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 1;
2038            accumulatedProfits := pre(accumulatedProfits) + portIn.monetaryValue;
2039            currentBusinessValue := pre(currentBusinessValue) + portIn.monetaryValue -
                    price1[pre(outPointerFIFO1)];
2040            IDVector[pre(portIn.ID)] := 2;
2041          end if;
2042          if ((not pre(nElementsFIFO1) < lengthFIFO1) and (not pre(nElementsFIFO2) <
                  lengthFIFO2)) then
2043            nElementsFIFO1 := pre(nElementsFIFO1) - 1;
2044            moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn.monetaryValue;
2045            accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
2046            currentBusinessValue := pre(currentBusinessValue) - price1[pre(
                    outPointerFIFO1)];
2047          end if;
2048        elseif (IDVector[pre(portIn.ID)] == 1) then
2049          if (pre(nElementsFIFO1) < lengthFIFO1) then
2050            FIFO1[pre(inPointerFIFO1)] := portIn.ID;
2051            price1[pre(inPointerFIFO1)] := portIn.monetaryValue;
2052            inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
```

```
2053            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 1;
2054            accumulatedProfits := pre(accumulatedProfits) + portIn.monetaryValue;
2055            currentBusinessValue := pre(currentBusinessValue) + portIn.monetaryValue -
                     price1[pre(outPointerFIFO1)];
2056          else
2057            nElementsFIFO1 := pre(nElementsFIFO1) - 1;
2058            moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn.monetaryValue;
2059            accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
2060            currentBusinessValue := pre(currentBusinessValue) - price1[pre(
                    outPointerFIFO1)];
2061          end if;
2062        elseif (IDVector[pre(portIn.ID)] == 2) then
2063          if (pre(nElementsFIFO2) < lengthFIFO2) then
2064            FIFO2[pre(inPointerFIFO2)] := portIn.ID;
2065            price2[pre(inPointerFIFO2)] := portIn.monetaryValue;
2066            inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
2067            nElementsFIFO2 := pre(nElementsFIFO2) + 1;
2068            nElementsFIFO1 := pre(nElementsFIFO1) - 1;
2069            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 1;
2070            accumulatedProfits := pre(accumulatedProfits) + portIn.monetaryValue;
2071            currentBusinessValue := pre(currentBusinessValue) + portIn.monetaryValue -
                     price1[pre(outPointerFIFO1)];
2072          else
2073            nElementsFIFO1 := pre(nElementsFIFO1) - 1;
2074            moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn.monetaryValue;
2075            accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
2076            currentBusinessValue := pre(currentBusinessValue) - price1[pre(
                    outPointerFIFO1)];
2077          end if;
2078        end if;
2079        /* Reading from FIFO1 */
2080        portOut1.ID := FIFO1[pre(outPointerFIFO1)];
2081        portOut1.monetaryValue := price1[pre(outPointerFIFO1)];
2082        outPointerFIFO1 := mod(pre(outPointerFIFO1), lengthFIFO1) + 1;
2083        /* Setting the flags on the ports */
2084        portIn.writeOkFlag := true;
2085        portOut1.writeFlag := true;
2086        state := 3;
2087        controlTime := time;
2088      end when;
2089
2090      /* Reading from Client ad writing to portService2 at the same time */
2091      when (pre(state) == 0 and pre(portIn.writeFlag) and ((not pre(portOut1.
           writeOkFlag)) or (not pre(nElementsFIFO1) > 0)) and (pre(portOut2.writeOkFlag
           )) and (pre(nElementsFIFO2) > 0) and time > controlTime) then
2092        if (IDVector[pre(portIn.ID)] == 0) then
2093          if ((pre(nElementsFIFO1) < lengthFIFO1) and (not pre(nElementsFIFO2) <
                 lengthFIFO2)) then
2094            FIFO1[pre(inPointerFIFO1)] := portIn.ID;
2095            price1[pre(inPointerFIFO1)] := portIn.monetaryValue;
2096            inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
2097            nElementsFIFO1 := pre(nElementsFIFO1) + 1;
2098            nElementsFIFO2 := pre(nElementsFIFO2) - 1;
2099            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 1;
```

```
2100            accumulatedProfits := pre(accumulatedProfits) + portIn.monetaryValue;
2101            currentBusinessValue := pre(currentBusinessValue) + portIn.monetaryValue -
                    price2[pre(outPointerFIFO2)];
2102            IDVector[pre(portIn.ID)] := 1;
2103          end if;
2104          if (pre(nElementsFIFO2) < lengthFIFO2 and (not pre(nElementsFIFO1) <
                  lengthFIFO1)) then
2105            FIFO2[pre(inPointerFIFO2)] := portIn.ID;
2106            price2[pre(inPointerFIFO2)] := portIn.monetaryValue;
2107            inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
2108            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 1;
2109            accumulatedProfits := pre(accumulatedProfits) + portIn.monetaryValue;
2110            currentBusinessValue := pre(currentBusinessValue) + portIn.monetaryValue -
                    price2[pre(outPointerFIFO2)];
2111            IDVector[pre(portIn.ID)] := 2;
2112          end if;
2113          if ((pre(nElementsFIFO1) < lengthFIFO1) and (pre(nElementsFIFO2) <
                  lengthFIFO2) and uniformRNG.outPort[1] >= 0.5) then
2114            FIFO1[pre(inPointerFIFO1)] := portIn.ID;
2115            price1[pre(inPointerFIFO1)] := portIn.monetaryValue;
2116            inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
2117            nElementsFIFO1 := pre(nElementsFIFO1) + 1;
2118            nElementsFIFO2 := pre(nElementsFIFO2) - 1;
2119            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 1;
2120            accumulatedProfits := pre(accumulatedProfits) + portIn.monetaryValue;
2121            currentBusinessValue := pre(currentBusinessValue) + portIn.monetaryValue -
                    price2[pre(outPointerFIFO2)];
2122            IDVector[pre(portIn.ID)] := 1;
2123          end if;
2124          if ((pre(nElementsFIFO2) < lengthFIFO2) and (pre(nElementsFIFO1) <
                  lengthFIFO1) and (not uniformRNG.outPort[1] >= 0.5)) then
2125            FIFO2[pre(inPointerFIFO2)] := portIn.ID;
2126            price2[pre(inPointerFIFO2)] := portIn.monetaryValue;
2127            inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
2128            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 1;
2129            accumulatedProfits := pre(accumulatedProfits) + portIn.monetaryValue;
2130            currentBusinessValue := pre(currentBusinessValue) + portIn.monetaryValue -
                    price2[pre(outPointerFIFO2)];
2131            IDVector[pre(portIn.ID)] := 2;
2132          end if;
2133          if ((not pre(nElementsFIFO1) < lengthFIFO1) and (not pre(nElementsFIFO2) <
                  lengthFIFO2)) then
2134            nElementsFIFO2 := pre(nElementsFIFO2) - 1;
2135            moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn.monetaryValue;
2136            accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
2137            currentBusinessValue := pre(currentBusinessValue) - price2[pre(
                    outPointerFIFO2)];
2138          end if;
2139        elseif (IDVector[pre(portIn.ID)] == 1) then
2140          if (pre(nElementsFIFO1) < lengthFIFO1) then
2141            FIFO1[pre(inPointerFIFO1)] := portIn.ID;
2142            price1[pre(inPointerFIFO1)] := portIn.monetaryValue;
2143            inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
2144            nElementsFIFO1 := pre(nElementsFIFO1) + 1;
```

```
2145            nElementsFIFO2 := pre(nElementsFIFO2) - 1;
2146            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 1;
2147            accumulatedProfits := pre(accumulatedProfits) + portIn.monetaryValue;
2148            currentBusinessValue := pre(currentBusinessValue) + portIn.monetaryValue -
                     price2[pre(outPointerFIFO2)];
2149          else
2150            nElementsFIFO2 := pre(nElementsFIFO2) - 1;
2151            moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn.monetaryValue;
2152            accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
2153            currentBusinessValue := pre(currentBusinessValue) - price2[pre(
                    outPointerFIFO2)];
2154          end if;
2155        elseif (IDVector[pre(portIn.ID)] == 2) then
2156          if (pre(nElementsFIFO2) < lengthFIFO2) then
2157            FIFO2[pre(inPointerFIFO2)] := portIn.ID;
2158            price2[pre(inPointerFIFO2)] := portIn.monetaryValue;
2159            inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
2160            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 1;
2161            accumulatedProfits := pre(accumulatedProfits) + portIn.monetaryValue;
2162            currentBusinessValue := pre(currentBusinessValue) + portIn.monetaryValue -
                     price2[pre(outPointerFIFO2)];
2163          else
2164            nElementsFIFO2 := pre(nElementsFIFO2) - 1;
2165            moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn.monetaryValue;
2166            accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
2167            currentBusinessValue := pre(currentBusinessValue) - price2[pre(
                    outPointerFIFO2)];
2168          end if;
2169        end if;
2170        /* Reading from FIFO2 */
2171        portOut2.ID := FIFO2[pre(outPointerFIFO2)];
2172        portOut2.monetaryValue := price2[pre(outPointerFIFO2)];
2173        outPointerFIFO2 := mod(pre(outPointerFIFO2), lengthFIFO2) + 1;
2174        /* Setting the flags on the ports */
2175        portIn.writeOkFlag := true;
2176        portOut2.writeFlag := true;
2177        state := 4;
2178        controlTime := time;
2179      end when;
2180
2181      /* Writing to portService1 */
2182      when (pre(state) == 0 and (not pre(portIn.writeFlag)) and pre(portOut1.
             writeOkFlag) and (pre(nElementsFIFO1) > 0) and ((not pre(portOut2.writeOkFlag
             )) or (not pre(nElementsFIFO2) > 0)) and time > controlTime) then
2183        /* Reading from FIFO1 */
2184        portOut1.ID := FIFO1[pre(outPointerFIFO1)];
2185        portOut1.monetaryValue := price1[pre(outPointerFIFO1)];
2186        outPointerFIFO1 := mod(pre(outPointerFIFO1), lengthFIFO1) + 1;
2187        nElementsFIFO1 := pre(nElementsFIFO1) - 1;
2188        currentBusinessValue := pre(currentBusinessValue) - price1[pre(outPointerFIFO1
               )];
2189        /* Setting the flags on the ports */
2190        portOut1.writeFlag := true;
2191        state := 5;
2192        controlTime := time;
```

```
2193        end when;
2194
2195        /* Writing to portService2 */
2196        when (pre(state) == 0 and (not pre(portIn.writeFlag)) and ((not pre(portOut1.
                writeOkFlag)) or (not pre(nElementsFIFO1) > 0)) and (pre(portOut2.writeOkFlag
                )) and (pre(nElementsFIFO2) > 0) and time > controlTime) then
2197          /* Reading from FIFO1 */
2198          portOut2.ID := FIFO2[pre(outPointerFIFO2)];
2199          portOut2.monetaryValue := price2[pre(outPointerFIFO2)];
2200          outPointerFIFO2 := mod(pre(outPointerFIFO2), lengthFIFO2) + 1;
2201          nElementsFIFO2 := pre(nElementsFIFO2) - 1;
2202          currentBusinessValue := pre(currentBusinessValue) - price2[pre(outPointerFIFO2
                )];
2203          /* Setting the flags on the ports */
2204          portOut2.writeFlag := true;
2205          state := 6;
2206          controlTime := time;
2207        end when;
2208
2209        // OK
2210
2211        /* Reading from Client ad writing to portService1 and portService2 at the same
                time */
2212        when (pre(state) == 0 and pre(portIn.writeFlag) and pre(portOut1.writeOkFlag)
            and (pre(nElementsFIFO1) > 0) and pre(portOut2.writeOkFlag) and pre(
            nElementsFIFO2) > 0 and time > controlTime) then
2213          if (IDVector[pre(portIn.ID)] == 0) then
2214            if ((pre(nElementsFIFO1) < lengthFIFO1) and (not pre(nElementsFIFO2) <
                    lengthFIFO2)) then
2215              FIFO1[pre(inPointerFIFO1)] := portIn.ID;
2216              price1[pre(inPointerFIFO1)] := portIn.monetaryValue;
2217              inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
2218              nElementsFIFO2 := pre(nElementsFIFO2) - 1;
2219              accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                        + 1;
2220              accumulatedProfits := pre(accumulatedProfits) + portIn.monetaryValue;
2221              currentBusinessValue := pre(currentBusinessValue) + portIn.monetaryValue -
                         price1[pre(outPointerFIFO1)] - price2[pre(outPointerFIFO2)];
2222              IDVector[pre(portIn.ID)] := 1;
2223            end if;
2224            if (pre(nElementsFIFO2) < lengthFIFO2 and (not pre(nElementsFIFO1) <
                    lengthFIFO1)) then
2225              FIFO2[pre(inPointerFIFO2)] := portIn.ID;
2226              price2[pre(inPointerFIFO2)] := portIn.monetaryValue;
2227              inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
2228              nElementsFIFO1 := pre(nElementsFIFO1) - 1;
2229              accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                        + 1;
2230              accumulatedProfits := pre(accumulatedProfits) + portIn.monetaryValue;
2231              currentBusinessValue := pre(currentBusinessValue) + portIn.monetaryValue -
                         price1[pre(outPointerFIFO1)] - price2[pre(outPointerFIFO2)];
2232              IDVector[pre(portIn.ID)] := 2;
2233            end if;
2234            if ((pre(nElementsFIFO1) < lengthFIFO1) and (pre(nElementsFIFO2) <
                    lengthFIFO2) and uniformRNG.outPort[1] >= 0.5) then
2235              FIFO1[pre(inPointerFIFO1)] := portIn.ID;
2236              price1[pre(inPointerFIFO1)] := portIn.monetaryValue;
```

```
2237            inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
2238            nElementsFIFO2 := pre(nElementsFIFO2) - 1;
2239            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 1;
2240            accumulatedProfits := pre(accumulatedProfits) + portIn.monetaryValue;
2241            currentBusinessValue := pre(currentBusinessValue) + portIn.monetaryValue -
                     price1[pre(outPointerFIFO1)] - price2[pre(outPointerFIFO2)];
2242            IDVector[pre(portIn.ID)] := 1;
2243          end if;
2244          if ((pre(nElementsFIFO2) < lengthFIFO2) and (pre(nElementsFIFO1) <
                 lengthFIFO1) and (not uniformRNG.outPort[1] >= 0.5)) then
2245            FIFO2[pre(inPointerFIFO2)] := portIn.ID;
2246            price2[pre(inPointerFIFO2)] := portIn.monetaryValue;
2247            inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
2248            nElementsFIFO1 := pre(nElementsFIFO1) - 1;
2249            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 1;
2250            accumulatedProfits := pre(accumulatedProfits) + portIn.monetaryValue;
2251            currentBusinessValue := pre(currentBusinessValue) + portIn.monetaryValue -
                     price1[pre(outPointerFIFO1)] - price2[pre(outPointerFIFO2)];
2252            IDVector[pre(portIn.ID)] := 2;
2253          end if;
2254          if ((not pre(nElementsFIFO1) < lengthFIFO1) and (not pre(nElementsFIFO2) <
                 lengthFIFO2)) then
2255            nElementsFIFO1 := pre(nElementsFIFO1) - 1;
2256            nElementsFIFO2 := pre(nElementsFIFO2) - 1;
2257            moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn.monetaryValue;
2258            accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
2259            currentBusinessValue := pre(currentBusinessValue) - price1[pre(
                    outPointerFIFO1)] - price2[pre(outPointerFIFO2)];
2260          end if;
2261        elseif (IDVector[pre(portIn.ID)] == 1) then
2262          if (pre(nElementsFIFO1) < lengthFIFO1) then
2263            FIFO1[pre(inPointerFIFO1)] := portIn.ID;
2264            price1[pre(inPointerFIFO1)] := portIn.monetaryValue;
2265            inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
2266            nElementsFIFO2 := pre(nElementsFIFO2) - 1;
2267            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 1;
2268            accumulatedProfits := pre(accumulatedProfits) + portIn.monetaryValue;
2269            currentBusinessValue := pre(currentBusinessValue) + portIn.monetaryValue -
                     price1[pre(outPointerFIFO1)] - price2[pre(outPointerFIFO2)];
2270          else
2271            nElementsFIFO1 := pre(nElementsFIFO1) - 1;
2272            nElementsFIFO2 := pre(nElementsFIFO2) - 1;
2273            moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn.monetaryValue;
2274            accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
2275            currentBusinessValue := pre(currentBusinessValue) - price1[pre(
                    outPointerFIFO1)] - price2[pre(outPointerFIFO2)];
2276          end if;
2277        elseif (IDVector[pre(portIn.ID)] == 2) then
2278          if (pre(nElementsFIFO2) < lengthFIFO2) then
2279            FIFO2[pre(inPointerFIFO2)] := portIn.ID;
2280            price2[pre(inPointerFIFO2)] := portIn.monetaryValue;
2281            inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
2282            nElementsFIFO1 := pre(nElementsFIFO1) - 1;
```

```
2283            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 1;
2284            accumulatedProfits := pre(accumulatedProfits) + portIn.monetaryValue;
2285            currentBusinessValue := pre(currentBusinessValue) + portIn.monetaryValue -
                     price1[pre(outPointerFIFO1)] - price2[pre(outPointerFIFO2)];
2286          else
2287            nElementsFIFO1 := pre(nElementsFIFO1) - 1;
2288            nElementsFIFO2 := pre(nElementsFIFO2) - 1;
2289            moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn.monetaryValue;
2290            accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
2291            currentBusinessValue := pre(currentBusinessValue) - price1[pre(
                    outPointerFIFO1)] - price2[pre(outPointerFIFO2)];
2292          end if;
2293        end if;
2294        /* Reading from FIFO1 */
2295        portOut1.ID := FIFO1[pre(outPointerFIFO1)];
2296        portOut1.monetaryValue := price1[pre(outPointerFIFO1)];
2297        outPointerFIFO1 := mod(pre(outPointerFIFO1), lengthFIFO1) + 1;
2298        /* Reading from FIFO2 */
2299        portOut2.ID := FIFO2[pre(outPointerFIFO2)];
2300        portOut2.monetaryValue := price2[pre(outPointerFIFO2)];
2301        outPointerFIFO2 := mod(pre(outPointerFIFO2), lengthFIFO2) + 1;
2302        /* Setting the flags on the ports */
2303        portIn.writeOkFlag := true;
2304        portOut1.writeFlag := true;
2305        portOut2.writeFlag := true;
2306        state := 7;
2307        controlTime := time;
2308      end when;
2309
2310      when (pre(state) == 1 and (not pre(portIn.writeFlag)) and time > controlTime)
             then
2311        portIn.writeOkFlag := false;
2312        state := 0;
2313        controlTime := time;
2314      end when;
2315
2316      when (pre(state) == 2 and (not pre(portOut1.writeOkFlag)) and (not pre(portOut2.
             writeOkFlag)) and time > controlTime) then
2317        portOut1.writeFlag := false;
2318        portOut2.writeFlag := false;
2319        state := 0;
2320        controlTime := time;
2321      end when;
2322
2323      when (pre(state) == 3 and (not pre(portIn.writeFlag)) and (not pre(portOut1.
             writeOkFlag)) and time > controlTime) then
2324        portIn.writeOkFlag := false;
2325        portOut1.writeFlag := false;
2326        state := 0;
2327        controlTime := time;
2328      end when;
2329
2330      when (pre(state) == 4 and (not pre(portIn.writeFlag)) and (not pre(portOut2.
             writeOkFlag)) and time > controlTime) then
2331        portIn.writeOkFlag := false;
2332        portOut2.writeFlag := false;
```

```
2333          state := 0;
2334          controlTime := time;
2335        end when;
2336
2337        when (pre(state) == 5 and (not pre(portOut1.writeOkFlag)) and time > controlTime
                ) then
2338          portOut1.writeFlag := false;
2339          state := 0;
2340          controlTime := time;
2341        end when;
2342
2343        when (pre(state) == 6 and (not pre(portOut2.writeOkFlag)) and time > controlTime
                ) then
2344          portOut2.writeFlag := false;
2345          state := 0;
2346          controlTime := time;
2347        end when;
2348
2349        when (pre(state) == 7 and (not pre(portIn.writeFlag)) and (not pre(portOut1.
                writeOkFlag)) and (not pre(portOut2.writeOkFlag)) and time > controlTime)
                then
2350          portIn.writeOkFlag := false;
2351          portOut1.writeFlag := false;
2352          portOut2.writeFlag := false;
2353          state := 0;
2354          controlTime := time;
2355        end when;
2356
2357      equation
2358        connect(portIn, portIn) annotation (Line(
2359            points={{-86,-2},{-86,13},{-86,13},{-86,-2}},
2360            color={0,0,0},
2361            pattern=LinePattern.Dash,
2362            thickness=0.5,
2363            smooth=Smooth.None));
2364        annotation (Icon(graphics={
2365              Ellipse(
2366                extent={{84,56},{-76,-64}},
2367                lineColor={0,0,0},
2368                lineThickness=0.5),
2369              Text(
2370                extent={{-32,-32},{44,-52}},
2371                lineColor={0,0,0},
2372                lineThickness=0.5,
2373                fillPattern=FillPattern.Solid,
2374               textString="circuit4"),
2375             Bitmap(
2376               extent={{-70,36},{76,-30}},
2377               imageSource=
2378                    "/*Put your image*/",
2379               fileName="modelica://RiskAnalysis/reoDymola.png")}),
2380                                          Diagram(graphics));
2381      end ReoCircuit4;
2382
2383      model ReoCircuit5
2384
2385        parameter Integer lengthFIFO1 = 100;
```

```
2386        parameter Integer lengthFIFO2 = 100;
2387        Integer nElementsFIFO1(start=0);
2388        Integer nElementsFIFO2(start=0);
2389        Integer inPointerFIFO1(start=1);
2390        Integer inPointerFIFO2(start=1);
2391        Integer outPointerFIFO1(start=1);
2392        Integer outPointerFIFO2(start=1);
2393        Integer moneyLossTotalFIFO(start=0);
2394        Integer state(start=0);
2395        Real controlTime(start=0);
2396
2397        Real freeCapacity(start=0);
2398        Real usedCapacity(start=0);
2399        Integer accumulatedLosses(start=0);
2400        Integer accumulatedNumberOfLostEvents(start=0);
2401        Integer accumulatedNumberOfSuccessEvents(start=0);
2402        Integer accumulatedProfits(start=0);
2403        Integer currentBusinessValue(start=0);
2404
2405      Modelon.Blocks.Sources.RandomNumbers.uniformRNG uniformRNG(nout=1,
2406          samplePeriod=0.1,
2407        firstSeed=Modelon.Math.RandomNumbers.Seed(
2408              x=201,
2409              y=178,
2410              z=99))
2411        annotation (Placement(transformation(extent={{-48,-54},{-28,-34}})));
2412
2413    public
2414      ServicePortOut portOut1 annotation (Placement(transformation(extent={{70,22},{
2415              90,42}}), iconTransformation(extent={{70,22},{90,42}})));
2416      ServicePortOut portOut2 annotation (Placement(transformation(extent={{68,-46},
2417              {88,-26}}), iconTransformation(extent={{68,-46},{88,-26}})));
2418      ServicePortIn portIn2 annotation (Placement(transformation(extent={{-90,-44},{
2419              -70,-24}}), iconTransformation(extent={{-90,-44},{-70,-24}})));
2420      ServicePortIn portIn1 annotation (Placement(transformation(extent={{-90,22},{-70,
2421              42}}), iconTransformation(extent={{-90,22},{-70,42}})));
2422
2423    protected
2424        Integer[lengthFIFO1] FIFO1;
2425        Integer[lengthFIFO1] price1;
2426        Integer[lengthFIFO2] FIFO2;
2427        Integer[lengthFIFO2] price2;
2428
2429    public
2430      Modelica.Blocks.Interfaces.RealOutput currentFIFOCapacity annotation (
2431          Placement(transformation(extent={{-6,-32},{14,-12}}),
2432            iconTransformation(
2433            extent={{-10,-10},{10,10}},
2434            rotation=-90,
2435            origin={-14,-68})));
2436      Modelica.Blocks.Interfaces.IntegerOutput outAccumulatedLosses annotation (
2437          Placement(transformation(extent={{-72,-54},{-52,-34}}),
2438            iconTransformation(
2439            extent={{-10,-10},{10,10}},
2440            rotation=-90,
2441            origin={20,-68})));
2442    equation
```

```
2443      usedCapacity =  ((nElementsFIFO1 + nElementsFIFO1) * 100) / (lengthFIFO1 +
              lengthFIFO2);
2444      freeCapacity =  100 - usedCapacity;
2445      accumulatedLosses = moneyLossTotalFIFO;
2446      currentFIFOCapacity = usedCapacity;
2447      outAccumulatedLosses = accumulatedLosses;
2448
2449    algorithm
2450       /* The circuit receives data from portIn1, while both portOut1 and portOut2 are
              not ready to read*/
2451       when (pre(state) == 0 and pre(portIn1.writeFlag) and (not pre(portIn2.writeFlag)
              ) and (not pre(portOut1.writeOkFlag) or (not pre(nElementsFIFO1) > 0)) and (
              not pre(portOut2.writeOkFlag) or (not pre(nElementsFIFO2) > 0)) and time >
              controlTime) then
2452         if ((pre(nElementsFIFO1) < lengthFIFO1) and (not pre(nElementsFIFO2) <
                lengthFIFO2)) then
2453           FIFO1[pre(inPointerFIFO1)] := portIn1.ID;
2454           price1[pre(inPointerFIFO1)] := portIn1.monetaryValue;
2455           inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
2456           nElementsFIFO1 := pre(nElementsFIFO1) + 1;
2457           accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                  1;
2458           accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
2459           currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue;
2460         end if;
2461         if (pre(nElementsFIFO2) < lengthFIFO2 and (not pre(nElementsFIFO1) <
                lengthFIFO1)) then
2462           FIFO2[pre(inPointerFIFO2)] := portIn1.ID;
2463           price2[pre(inPointerFIFO2)] := portIn1.monetaryValue;
2464           inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
2465           nElementsFIFO2 := pre(nElementsFIFO2) + 1;
2466           accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                  1;
2467           accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
2468           currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue;
2469         end if;
2470         if ((pre(nElementsFIFO1) < lengthFIFO1) and (pre(nElementsFIFO2) < lengthFIFO2
              ) and uniformRNG.outPort[1] >= 0.5) then
2471           FIFO1[pre(inPointerFIFO1)] := portIn1.ID;
2472           price1[pre(inPointerFIFO1)] := portIn1.monetaryValue;
2473           inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
2474           nElementsFIFO1 := pre(nElementsFIFO1) + 1;
2475           accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                  1;
2476           accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
2477           currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue;
2478         end if;
2479         if ((pre(nElementsFIFO2) < lengthFIFO2) and (pre(nElementsFIFO1) < lengthFIFO1
              ) and (not uniformRNG.outPort[1] >= 0.5)) then
2480           FIFO2[pre(inPointerFIFO2)] := portIn1.ID;
2481           price2[pre(inPointerFIFO2)] := portIn1.monetaryValue;
2482           inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
2483           nElementsFIFO2 := pre(nElementsFIFO2) + 1;
2484           accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                  1;
2485           accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
2486           currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue;
```

```
2487          end if;
2488          if ((not pre(nElementsFIFO1) < lengthFIFO1) and (not pre(nElementsFIFO2) <
                  lengthFIFO2)) then
2489            moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn1.monetaryValue;
2490            accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
2491          end if;
2492          portIn1.writeOkFlag := true;
2493          state := 1;
2494          controlTime := time;
2495        end when;
2496
2497        /* The circuit receives data from portIn2, while both portOut1 and portOut2 are
               not ready to read*/
2498        when (pre(state) == 0 and pre(portIn2.writeFlag) and (not pre(portIn1.writeFlag)
               ) and (not pre(portOut1.writeOkFlag) or (not pre(nElementsFIFO1) > 0)) and (
               not pre(portOut2.writeOkFlag) or (not pre(nElementsFIFO2) > 0)) and time >
               controlTime) then
2499          if ((pre(nElementsFIFO1) < lengthFIFO1) and (not pre(nElementsFIFO2) <
                  lengthFIFO2)) then
2500            FIFO1[pre(inPointerFIFO1)] := portIn2.ID;
2501            price1[pre(inPointerFIFO1)] := portIn2.monetaryValue;
2502            inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
2503            nElementsFIFO1 := pre(nElementsFIFO1) + 1;
2504            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                    1;
2505            accumulatedProfits := pre(accumulatedProfits) + portIn2.monetaryValue;
2506            currentBusinessValue := pre(currentBusinessValue) + portIn2.monetaryValue;
2507          end if;
2508          if (pre(nElementsFIFO2) < lengthFIFO2 and (not pre(nElementsFIFO1) <
                  lengthFIFO1)) then
2509            FIFO2[pre(inPointerFIFO2)] := portIn2.ID;
2510            price2[pre(inPointerFIFO2)] := portIn2.monetaryValue;
2511            inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
2512            nElementsFIFO2 := pre(nElementsFIFO2) + 1;
2513            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                    1;
2514            accumulatedProfits := pre(accumulatedProfits) + portIn2.monetaryValue;
2515            currentBusinessValue := pre(currentBusinessValue) + portIn2.monetaryValue;
2516          end if;
2517          if ((pre(nElementsFIFO1) < lengthFIFO1) and (pre(nElementsFIFO2) < lengthFIFO2
                  ) and uniformRNG.outPort[1] >= 0.5) then
2518            FIFO1[pre(inPointerFIFO1)] := portIn2.ID;
2519            price1[pre(inPointerFIFO1)] := portIn2.monetaryValue;
2520            inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
2521            nElementsFIFO1 := pre(nElementsFIFO1) + 1;
2522            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                    1;
2523            accumulatedProfits := pre(accumulatedProfits) + portIn2.monetaryValue;
2524            currentBusinessValue := pre(currentBusinessValue) + portIn2.monetaryValue;
2525          end if;
2526          if ((pre(nElementsFIFO2) < lengthFIFO2) and (pre(nElementsFIFO1) < lengthFIFO1
                  ) and (not uniformRNG.outPort[1] >= 0.5)) then
2527            FIFO2[pre(inPointerFIFO2)] := portIn2.ID;
2528            price2[pre(inPointerFIFO2)] := portIn2.monetaryValue;
2529            inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
2530            nElementsFIFO2 := pre(nElementsFIFO2) + 1;
```

```
2531            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                    1;
2532            accumulatedProfits := pre(accumulatedProfits) + portIn2.monetaryValue;
2533            currentBusinessValue := pre(currentBusinessValue) + portIn2.monetaryValue;
2534          end if;
2535          if ((not pre(nElementsFIFO1) < lengthFIFO1) and (not pre(nElementsFIFO2) <
                 lengthFIFO2)) then
2536            moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn2.monetaryValue;
2537            accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
2538          end if;
2539          portIn2.writeOkFlag := true;
2540          state := 8;
2541          controlTime := time;
2542        end when;
2543
2544        /* No input from portIn1 and portIn2 but portOut1 and portOut2 read at the same
               time */
2545        when (pre(state) == 0 and (not pre(portIn1.writeFlag)) and (not pre(portIn2.
               writeFlag)) and pre(portOut1.writeOkFlag) and (pre(nElementsFIFO1) > 0) and
               pre(portOut2.writeOkFlag) and (pre(nElementsFIFO2) > 0) and time >
               controlTime) then
2546          /* Reading from FIFO1 */
2547          portOut1.ID := FIFO1[pre(outPointerFIFO1)];
2548          portOut1.monetaryValue := price1[pre(outPointerFIFO1)];
2549          outPointerFIFO1 := mod(pre(outPointerFIFO1), lengthFIFO1) + 1;
2550          nElementsFIFO1 := pre(nElementsFIFO1) - 1;
2551          /* Reading from FIFO2 */
2552          portOut2.ID := FIFO2[pre(outPointerFIFO2)];
2553          portOut2.monetaryValue := price2[pre(outPointerFIFO2)];
2554          portOut2.monetaryValue := price2[pre(outPointerFIFO2)];
2555          outPointerFIFO2 := mod(pre(outPointerFIFO2), lengthFIFO2) + 1;
2556          nElementsFIFO2 := pre(nElementsFIFO2) - 1;
2557          currentBusinessValue := pre(currentBusinessValue) - price1[pre(outPointerFIFO1
                 )] - price2[pre(outPointerFIFO2)];
2558          /* Setting the flags on the ports */
2559          portOut1.writeFlag := true;
2560          portOut2.writeFlag := true;
2561          state := 2;
2562          controlTime := time;
2563        end when;
2564
2565      /* The circuit receives data in input from portIn1 and from portIn2 but no write
             to portOut1 or portOut2 */
2566      when (pre(state) == 0 and pre(portIn1.writeFlag) and pre(portIn2.writeFlag) and
             ((not pre(portOut1.writeOkFlag)) or (not pre(nElementsFIFO1) > 0)) and ((not
             pre(portOut2.writeOkFlag)) or (not pre(nElementsFIFO2) > 0)) and time >
             controlTime) then
2567          // If one of the two queues is full
2568          if ((pre(nElementsFIFO1) < lengthFIFO1) and (not pre(nElementsFIFO2) <
                 lengthFIFO2)) then
2569          // Adding port1.ID
2570          FIFO1[pre(inPointerFIFO1)] := portIn1.ID;
2571          price1[pre(inPointerFIFO1)] := portIn1.monetaryValue;
2572          // Adding port2.ID if there is still space
2573          if ((pre(nElementsFIFO1) + 1) < lengthFIFO1) then
2574            FIFO1[pre(inPointerFIFO1)+1] := portIn2.ID;
2575            price1[pre(inPointerFIFO1)+1] := portIn2.monetaryValue;
```

```
2576            //inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 2;
2577            if (pre(nElementsFIFO1) == (lengthFIFO1 -2)) then
2578              inPointerFIFO1 := 1;
2579            else
2580              inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 2;
2581            end if;
2582            nElementsFIFO1 := pre(nElementsFIFO1) + 2;
2583            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                   + 2;
2584            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue +
                   portIn2.monetaryValue;
2585            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                   + portIn2.monetaryValue;
2586          else
2587            inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
2588            nElementsFIFO1 := pre(nElementsFIFO1) + 1;
2589            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                   + 1;
2590            moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn2.monetaryValue;
2591            accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
2592            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
2593            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue;
2594          end if;
2595        end if;
2596        if (pre(nElementsFIFO2) < lengthFIFO2 and (not pre(nElementsFIFO1) <
               lengthFIFO1)) then
2597          // Adding port1.ID
2598          FIFO2[pre(inPointerFIFO2)] := portIn1.ID;
2599          price2[pre(inPointerFIFO2)] := portIn1.monetaryValue;
2600          // Adding port2.ID if there is still space
2601          if ((pre(nElementsFIFO2) + 1) < lengthFIFO2) then
2602            FIFO2[pre(inPointerFIFO2)+1] := portIn2.ID;
2603            price2[pre(inPointerFIFO2)+1] := portIn2.monetaryValue;
2604            //inPointerFIFO2 := mod(pre(inPointerFIFO2) + 2, lengthFIFO2);
2605            if (pre(nElementsFIFO2) == (lengthFIFO2 -2)) then
2606              inPointerFIFO2 := 1;
2607            else
2608              inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 2;
2609            end if;
2610            nElementsFIFO2 := pre(nElementsFIFO2) + 2;
2611            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                   + 2;
2612            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue +
                   portIn2.monetaryValue;
2613            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                   + portIn2.monetaryValue;
2614          else
2615            inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
2616            nElementsFIFO2 := pre(nElementsFIFO2) + 1;
2617            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                   + 1;
2618            moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn2.monetaryValue;
2619            accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
2620            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
2621            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue;
2622          end if;
2623        end if;
```

```
2624        // If the two queues are not full, nondeterministic choice
2625        if ((pre(nElementsFIFO1) < lengthFIFO1) and (pre(nElementsFIFO2) < lengthFIFO2
                ) and uniformRNG.outPort[1] >= 0.5) then
2626          FIFO1[pre(inPointerFIFO1)] := portIn1.ID;
2627          price1[pre(inPointerFIFO1)] := portIn1.monetaryValue;
2628          if ((uniformRNG.outPort[1] >= 0.75) and (pre(nElementsFIFO1) + 1 <
                  lengthFIFO1)) then
2629            FIFO1[pre(inPointerFIFO1)+1] := portIn2.ID;
2630            price1[pre(inPointerFIFO1)+1] := portIn2.monetaryValue;
2631            //inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 2;
2632            if (pre(nElementsFIFO1) == (lengthFIFO1 -2)) then
2633              inPointerFIFO1 := 1;
2634            else
2635              inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 2;
2636            end if;
2637            nElementsFIFO1 := pre(nElementsFIFO1) + 2;
2638            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 2;
2639            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue +
                    portIn2.monetaryValue;
2640            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                    + portIn2.monetaryValue;
2641          else
2642            FIFO2[pre(inPointerFIFO2)] := portIn2.ID;
2643            price2[pre(inPointerFIFO2)] := portIn2.monetaryValue;
2644            inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
2645            nElementsFIFO2 := pre(nElementsFIFO2) + 1;
2646            inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
2647            nElementsFIFO1 := pre(nElementsFIFO1) + 1;
2648            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 2;
2649            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue +
                    portIn2.monetaryValue;
2650            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                    + portIn2.monetaryValue;
2651          end if;
2652        end if;
2653        if ((pre(nElementsFIFO2) < lengthFIFO2) and (pre(nElementsFIFO1) < lengthFIFO1
                ) and (not uniformRNG.outPort[1] >= 0.5)) then
2654          FIFO2[pre(inPointerFIFO2)] := portIn1.ID;
2655          price2[pre(inPointerFIFO2)] := portIn1.monetaryValue;
2656          if ((uniformRNG.outPort[1] <= 0.25) and (pre(nElementsFIFO2) + 1 <
                  lengthFIFO2)) then
2657            FIFO2[pre(inPointerFIFO2)+1] := portIn2.ID;
2658            price2[pre(inPointerFIFO2)+1] := portIn2.monetaryValue;
2659            //inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 2;
2660            if (pre(nElementsFIFO2) == (lengthFIFO2 -2)) then
2661              inPointerFIFO2 := 1;
2662            else
2663              inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 2;
2664            end if;
2665            nElementsFIFO2 := pre(nElementsFIFO2) + 2;
2666            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 2;
2667            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue +
                    portIn2.monetaryValue;
```

```
2668            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                    + portIn2.monetaryValue;
2669          else
2670            FIFO1[pre(inPointerFIFO1)] := portIn2.ID;
2671            price1[pre(inPointerFIFO1)] := portIn2.monetaryValue;
2672            inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
2673            nElementsFIFO1 := pre(nElementsFIFO1) + 1;
2674            inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
2675            nElementsFIFO2 := pre(nElementsFIFO2) + 1;
2676            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 2;
2677            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue +
                     portIn2.monetaryValue;
2678            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                    + portIn2.monetaryValue;
2679          end if;
2680        end if;
2681        // Both FIFOs are full
2682        if ((not pre(nElementsFIFO1) < lengthFIFO1) and (not pre(nElementsFIFO2) <
                lengthFIFO2)) then
2683          moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn1.monetaryValue +
                   portIn2.monetaryValue;
2684          accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 2;
2685        end if;
2686        portIn1.writeOkFlag := true;
2687        portIn2.writeOkFlag := true;
2688        state := 12;
2689        controlTime := time;
2690      end when;
2691
2692       /* Reading from portIn1 ad writing to portOut1 at the same time */
2693       when (pre(state) == 0 and pre(portIn1.writeFlag) and (not pre(portIn2.writeFlag)
              ) and pre(portOut1.writeOkFlag) and (pre(nElementsFIFO1) > 0) and ((not pre(
             portOut2.writeOkFlag)) or (not pre(nElementsFIFO2) > 0)) and time >
             controlTime) then
2694        if ((pre(nElementsFIFO1) < lengthFIFO1) and (not pre(nElementsFIFO2) <
               lengthFIFO2)) then
2695          FIFO1[pre(inPointerFIFO1)] := portIn1.ID;
2696          price1[pre(inPointerFIFO1)] := portIn1.monetaryValue;
2697          inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
2698          accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                   1;
2699          accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
2700          currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue -
                   price1[pre(outPointerFIFO1)];
2701        end if;
2702        if (pre(nElementsFIFO2) < lengthFIFO2 and (not pre(nElementsFIFO1) <
               lengthFIFO1)) then
2703          FIFO2[pre(inPointerFIFO2)] := portIn1.ID;
2704          price2[pre(inPointerFIFO2)] := portIn1.monetaryValue;
2705          inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
2706          nElementsFIFO2 := pre(nElementsFIFO2) + 1;
2707          nElementsFIFO1 := pre(nElementsFIFO1) - 1;
2708          accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                   1;
2709          accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
```

```
2710            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue -
                    price1[pre(outPointerFIFO1)];
2711          end if;
2712          if ((pre(nElementsFIFO1) < lengthFIFO1) and (pre(nElementsFIFO2) < lengthFIFO2
                ) and uniformRNG.outPort[1] >= 0.5) then
2713            FIFO1[pre(inPointerFIFO1)] := portIn1.ID;
2714            price1[pre(inPointerFIFO1)] := portIn1.monetaryValue;
2715            inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
2716            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                    1;
2717            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
2718            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue -
                    price1[pre(outPointerFIFO1)];
2719          end if;
2720          if ((pre(nElementsFIFO2) < lengthFIFO2) and (pre(nElementsFIFO1) < lengthFIFO1
                ) and (not uniformRNG.outPort[1] >= 0.5)) then
2721            FIFO2[pre(inPointerFIFO2)] := portIn1.ID;
2722            price2[pre(inPointerFIFO2)] := portIn1.monetaryValue;
2723            inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
2724            nElementsFIFO2 := pre(nElementsFIFO2) + 1;
2725            nElementsFIFO1 := pre(nElementsFIFO1) - 1;
2726            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                    1;
2727            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
2728            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue -
                    price1[pre(outPointerFIFO1)];
2729          end if;
2730          if ((not pre(nElementsFIFO1) < lengthFIFO1) and (not pre(nElementsFIFO2) <
                lengthFIFO2)) then
2731            nElementsFIFO1 := pre(nElementsFIFO1) - 1;
2732            moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn1.monetaryValue;
2733            accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
2734            currentBusinessValue := pre(currentBusinessValue) - price1[pre(
                    outPointerFIFO1)];
2735          end if;
2736          /* Reading from FIFO1 */
2737          portOut1.ID := FIFO1[pre(outPointerFIFO1)];
2738          portOut1.monetaryValue := price1[pre(outPointerFIFO1)];
2739          outPointerFIFO1 := mod(pre(outPointerFIFO1), lengthFIFO1) + 1;
2740          /* Setting the flags on the ports */
2741          portIn1.writeOkFlag := true;
2742          portOut1.writeFlag := true;
2743          state := 3;
2744          controlTime := time;
2745        end when;
2746
2747        /* Reading from portIn1 and portIn2 ad writing to portOut1 at the same time */
2748        when (pre(state) == 0 and pre(portIn1.writeFlag) and pre(portIn2.writeFlag) and
                pre(portOut1.writeOkFlag) and (pre(nElementsFIFO1) > 0) and ((not pre(
                portOut2.writeOkFlag)) or (not pre(nElementsFIFO2) > 0)) and time >
                controlTime) then
2749          if ((pre(nElementsFIFO1) < lengthFIFO1) and (not pre(nElementsFIFO2) <
                  lengthFIFO2)) then
2750            // Adding port1.ID
2751            FIFO1[pre(inPointerFIFO1)] := portIn1.ID;
2752            price1[pre(inPointerFIFO1)] := portIn1.monetaryValue;
2753            // Adding port2.ID if there is still space
```

```
2754            if ((pre(nElementsFIFO1) + 1) < lengthFIFO1) then
2755              FIFO1[pre(inPointerFIFO1)+1] := portIn2.ID;
2756              price1[pre(inPointerFIFO1)+1] := portIn2.monetaryValue;
2757              //inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 2;
2758              if (pre(nElementsFIFO1) == (lengthFIFO1 -2)) then
2759               inPointerFIFO1 := 1;
2760              else
2761               inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 2;
2762              end if;
2763              accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 2;
2764              nElementsFIFO1 := pre(nElementsFIFO1) + 1;
2765              accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue +
                     portIn2.monetaryValue;
2766              currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                    + portIn2.monetaryValue - price1[pre(outPointerFIFO1)];
2767            else
2768              inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
2769              accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 1;
2770              moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn2.monetaryValue;
2771              accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
2772              accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
2773              currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                    - price1[pre(outPointerFIFO1)];
2774            end if;
2775          end if;
2776          if (pre(nElementsFIFO2) < lengthFIFO2 and (not pre(nElementsFIFO1) <
                lengthFIFO1)) then
2777            // Adding port1.ID
2778            FIFO2[pre(inPointerFIFO2)] := portIn1.ID;
2779            price2[pre(inPointerFIFO2)] := portIn1.monetaryValue;
2780            nElementsFIFO1 := pre(nElementsFIFO1) - 1;
2781            // Adding port2.ID if there is still space
2782            if ((pre(nElementsFIFO2) + 1) < lengthFIFO2) then
2783              FIFO2[pre(inPointerFIFO2)+1] := portIn2.ID;
2784              price2[pre(inPointerFIFO2)+1] := portIn2.monetaryValue;
2785              //inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 2;
2786              if (pre(nElementsFIFO2) == (lengthFIFO2 -2)) then
2787                inPointerFIFO2 := 1;
2788              else
2789                inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 2;
2790              end if;
2791              accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 2;
2792              nElementsFIFO2 := pre(nElementsFIFO2) + 2;
2793              accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue +
                     portIn2.monetaryValue;
2794              currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                    + portIn2.monetaryValue - price1[pre(outPointerFIFO1)];
2795            else
2796              inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
2797              nElementsFIFO2 := pre(nElementsFIFO2) + 1;
2798              accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 1;
2799              moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn2.monetaryValue;
2800              accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
```

```
2801            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
2802            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                    - price1[pre(outPointerFIFO1)];
2803          end if;
2804        end if;
2805        // If the two queues are not full, nondeterministic choice
2806        if ((pre(nElementsFIFO1) < lengthFIFO1) and (pre(nElementsFIFO2) < lengthFIFO2
                ) and uniformRNG.outPort[1] >= 0.5) then
2807          FIFO1[pre(inPointerFIFO1)] := portIn1.ID;
2808          price1[pre(inPointerFIFO1)] := portIn1.monetaryValue;
2809          if ((uniformRNG.outPort[1] >= 0.75) and (pre(nElementsFIFO1) + 1 <
                  lengthFIFO1)) then
2810            FIFO1[pre(inPointerFIFO1)+1] := portIn2.ID;
2811            price1[pre(inPointerFIFO1)+1] := portIn2.monetaryValue;
2812            //inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 2;
2813            if (pre(nElementsFIFO1) == (lengthFIFO1 -2)) then
2814              inPointerFIFO1 := 1;
2815            else
2816              inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 2;
2817            end if;
2818            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 2;
2819            nElementsFIFO1 := pre(nElementsFIFO1) + 1;
2820            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue +
                    portIn2.monetaryValue;
2821            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                    + portIn2.monetaryValue - price1[pre(outPointerFIFO1)];
2822          else
2823            FIFO2[pre(inPointerFIFO2)] := portIn2.ID;
2824            price2[pre(inPointerFIFO2)] := portIn2.monetaryValue;
2825            inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
2826            nElementsFIFO2 := pre(nElementsFIFO2) + 1;
2827            inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
2828            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 2;
2829            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue +
                    portIn2.monetaryValue;
2830            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                    + portIn2.monetaryValue - price1[pre(outPointerFIFO1)];
2831          end if;
2832        end if;
2833        if ((pre(nElementsFIFO2) < lengthFIFO2) and (pre(nElementsFIFO1) < lengthFIFO1
                ) and (not uniformRNG.outPort[1] >= 0.5)) then
2834          FIFO2[pre(inPointerFIFO2)] := portIn1.ID;
2835          price2[pre(inPointerFIFO2)] := portIn1.monetaryValue;
2836          if ((uniformRNG.outPort[1] <= 0.25) and (pre(nElementsFIFO2) + 1 <
                  lengthFIFO2)) then
2837            FIFO2[pre(inPointerFIFO2)+1] := portIn2.ID;
2838            price2[pre(inPointerFIFO2)+1] := portIn2.monetaryValue;
2839            //inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 2;
2840            if (pre(nElementsFIFO2) == (lengthFIFO2 -2)) then
2841              inPointerFIFO2 := 1;
2842            else
2843              inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 2;
2844            end if;
2845            nElementsFIFO2 := pre(nElementsFIFO2) + 2;
2846            nElementsFIFO1 := pre(nElementsFIFO1) - 1;
```

```
2847            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 2;
2848            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue +
                    portIn2.monetaryValue;
2849            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                    + portIn2.monetaryValue - price1[pre(outPointerFIFO1)];
2850          else
2851            FIFO1[pre(inPointerFIFO1)] := portIn2.ID;
2852            price1[pre(inPointerFIFO1)] := portIn2.monetaryValue;
2853            inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
2854            inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
2855            nElementsFIFO2 := pre(nElementsFIFO2) + 1;
2856            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 2;
2857            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue +
                    portIn2.monetaryValue;
2858            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                    + portIn2.monetaryValue - price1[pre(outPointerFIFO1)];
2859          end if;
2860        end if;
2861        if (not (pre(nElementsFIFO1) < lengthFIFO1) and (not pre(nElementsFIFO2) <
              lengthFIFO2)) then
2862          nElementsFIFO1 := pre(nElementsFIFO1) - 1;
2863          moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn1.monetaryValue +
                portIn2.monetaryValue;
2864          accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 2;
2865          currentBusinessValue := pre(currentBusinessValue) - price1[pre(
                outPointerFIFO1)];
2866        end if;
2867        /* Reading from FIFO1 */
2868        portOut1.ID := FIFO1[pre(outPointerFIFO1)];
2869        portOut1.monetaryValue := price1[pre(outPointerFIFO1)];
2870        outPointerFIFO1 := mod(pre(outPointerFIFO1), lengthFIFO1) + 1;
2871        /* Setting the flags on the ports */
2872        portIn1.writeOkFlag := true;
2873        portIn2.writeOkFlag := true;
2874        portOut1.writeFlag := true;
2875        state := 13;
2876        controlTime := time;
2877      end when;
2878
2879      /* Reading from portIn1 and portIn2 ad writing to portOut2 at the same time */
2880      when (pre(state) == 0 and pre(portIn1.writeFlag) and pre(portIn2.writeFlag) and
            pre(portOut2.writeOkFlag) and (pre(nElementsFIFO2) > 0) and ((not pre(
            portOut1.writeOkFlag)) or (not pre(nElementsFIFO1) > 0)) and time >
            controlTime) then
2881        if ((pre(nElementsFIFO1) < lengthFIFO1) and (not pre(nElementsFIFO2) <
              lengthFIFO2)) then
2882        // Adding port1.ID
2883        FIFO1[pre(inPointerFIFO1)] := portIn1.ID;
2884        price1[pre(inPointerFIFO1)] := portIn1.monetaryValue;
2885        nElementsFIFO2 := pre(nElementsFIFO2) - 1;
2886        // Adding port2.ID if there is still space
2887        if ((pre(nElementsFIFO1) + 1) < lengthFIFO1) then
2888          FIFO1[pre(inPointerFIFO1)+1] := portIn2.ID;
2889          price1[pre(inPointerFIFO1)+1] := portIn2.monetaryValue;
2890          //inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 2;
```

```
2891            if (pre(nElementsFIFO1) == (lengthFIFO1 -2)) then
2892              inPointerFIFO1 := 1;
2893            else
2894              inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 2;
2895            end if;
2896            nElementsFIFO1 := pre(nElementsFIFO1) + 2;
2897            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                   + 2;
2898            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue +
                   portIn2.monetaryValue;
2899            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                   + portIn2.monetaryValue - price2[pre(outPointerFIFO2)];
2900          else
2901            inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
2902            nElementsFIFO1 := pre(nElementsFIFO1) + 1;
2903            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                   + 1;
2904            moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn2.monetaryValue;
2905            accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
2906            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
2907            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                   - price2[pre(outPointerFIFO2)];
2908          end if;
2909        end if;
2910        if (pre(nElementsFIFO2) < lengthFIFO2 and (not pre(nElementsFIFO1) <
               lengthFIFO1)) then
2911          // Adding port1.ID
2912          FIFO2[pre(inPointerFIFO2)] := portIn1.ID;
2913          price2[pre(inPointerFIFO2)] := portIn1.monetaryValue;
2914          // Adding port2.ID if there is still space
2915          if ((pre(nElementsFIFO2) + 1) < lengthFIFO2) then
2916            FIFO2[pre(inPointerFIFO2)+1] := portIn2.ID;
2917            price2[pre(inPointerFIFO2)+1] := portIn2.monetaryValue;
2918            //inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 2;
2919            if (pre(nElementsFIFO2) == (lengthFIFO2 -2)) then
2920              inPointerFIFO2 := 1;
2921            else
2922              inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 2;
2923            end if;
2924            nElementsFIFO2 := pre(nElementsFIFO2) + 1;
2925            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                   + 2;
2926            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue +
                   portIn2.monetaryValue;
2927            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                   + portIn2.monetaryValue - price2[pre(outPointerFIFO2)];
2928          else
2929            inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
2930            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                   + 1;
2931            moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn2.monetaryValue;
2932            accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
2933            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
2934            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                   - price2[pre(outPointerFIFO2)];
2935          end if;
2936        end if;
```

```
2937        // If the two queues are not full, nondeterministic choice
2938        if ((pre(nElementsFIFO1) < lengthFIFO1) and (pre(nElementsFIFO2) < lengthFIFO2
                ) and uniformRNG.outPort[1] >= 0.5) then
2939          FIFO1[pre(inPointerFIFO1)] := portIn1.ID;
2940          price1[pre(inPointerFIFO1)] := portIn1.monetaryValue;
2941          if ((uniformRNG.outPort[1] >= 0.75) and (pre(nElementsFIFO1) + 1 <
                  lengthFIFO1)) then
2942            FIFO1[pre(inPointerFIFO1)+1] := portIn2.ID;
2943            price1[pre(inPointerFIFO1)+1] := portIn2.monetaryValue;
2944            //inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 2;
2945            if (pre(nElementsFIFO1) == (lengthFIFO1 -2)) then
2946              inPointerFIFO1 := 1;
2947            else
2948              inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 2;
2949            end if;
2950            nElementsFIFO1 := pre(nElementsFIFO1) + 2;
2951            nElementsFIFO2 := pre(nElementsFIFO2) - 1;
2952            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 2;
2953            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue +
                    portIn2.monetaryValue;
2954            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                    + portIn2.monetaryValue - price2[pre(outPointerFIFO2)];
2955          else
2956            FIFO2[pre(inPointerFIFO2)] := portIn2.ID;
2957            price2[pre(inPointerFIFO2)] := portIn2.monetaryValue;
2958            inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
2959            inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
2960            nElementsFIFO1 := pre(nElementsFIFO1) + 1;
2961            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 2;
2962            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue +
                    portIn2.monetaryValue;
2963            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                    + portIn2.monetaryValue - price2[pre(outPointerFIFO2)];
2964          end if;
2965        end if;
2966        if ((pre(nElementsFIFO2) < lengthFIFO2) and (pre(nElementsFIFO1) < lengthFIFO1
                ) and (not uniformRNG.outPort[1] >= 0.5)) then
2967          FIFO2[pre(inPointerFIFO2)] := portIn1.ID;
2968          price2[pre(inPointerFIFO2)] := portIn1.monetaryValue;
2969          if ((uniformRNG.outPort[1] <= 0.25) and (pre(nElementsFIFO2) + 1 <
                  lengthFIFO2)) then
2970            FIFO2[pre(inPointerFIFO2)+1] := portIn2.ID;
2971            price2[pre(inPointerFIFO2)+1] := portIn2.monetaryValue;
2972            //inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 2;
2973            if (pre(nElementsFIFO2) == (lengthFIFO2 -2)) then
2974              inPointerFIFO2 := 1;
2975            else
2976              inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 2;
2977            end if;
2978            nElementsFIFO2 := pre(nElementsFIFO2) + 1;
2979            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 2;
2980            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue +
                    portIn2.monetaryValue;
```

```
2981            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                    + portIn2.monetaryValue - price2[pre(outPointerFIFO2)];
2982          else
2983            FIFO1[pre(inPointerFIFO1)] := portIn2.ID;
2984            price1[pre(inPointerFIFO1)] := portIn2.monetaryValue;
2985            inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
2986            nElementsFIFO1 := pre(nElementsFIFO1) + 1;
2987            inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
2988            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 2;
2989            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue +
                     portIn2.monetaryValue;
2990            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                    + portIn2.monetaryValue - price2[pre(outPointerFIFO2)];
2991          end if;
2992        end if;
2993        if (not (pre(nElementsFIFO1) < lengthFIFO1) and (not pre(nElementsFIFO2) <
               lengthFIFO2)) then
2994          nElementsFIFO2 := pre(nElementsFIFO2) - 1;
2995          moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn1.monetaryValue +
                  portIn2.monetaryValue;
2996          accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 2;
2997          currentBusinessValue := pre(currentBusinessValue)  - price2[pre(
                  outPointerFIFO2)];
2998        end if;
2999        /* Reading from FIFO2 */
3000        portOut2.ID := FIFO2[pre(outPointerFIFO2)];
3001        portOut2.monetaryValue := price2[pre(outPointerFIFO2)];
3002        outPointerFIFO2 := mod(pre(outPointerFIFO2), lengthFIFO2) + 1;
3003        /* Setting the flags on the ports */
3004        portIn1.writeOkFlag := true;
3005        portIn2.writeOkFlag := true;
3006        portOut2.writeFlag := true;
3007        state := 14;
3008        controlTime := time;
3009      end when;
3010
3011      /* Reading from portIn2 ad writing to portOut1 at the same time */
3012      when (pre(state) == 0 and pre(portIn2.writeFlag) and (not pre(portIn1.writeFlag)
             ) and pre(portOut1.writeOkFlag) and (pre(nElementsFIFO1) > 0) and ((not pre(
             portOut2.writeOkFlag)) or (not pre(nElementsFIFO2) > 0)) and time >
             controlTime) then
3013        if ((pre(nElementsFIFO1) < lengthFIFO1) and (not pre(nElementsFIFO2) <
               lengthFIFO2)) then
3014          FIFO1[pre(inPointerFIFO1)] := portIn2.ID;
3015          price1[pre(inPointerFIFO1)] := portIn2.monetaryValue;
3016          inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
3017          accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                  1;
3018          accumulatedProfits := pre(accumulatedProfits) + portIn2.monetaryValue;
3019          currentBusinessValue := pre(currentBusinessValue) + portIn2.monetaryValue -
                  price1[pre(outPointerFIFO1)];
3020        end if;
3021        if (pre(nElementsFIFO2) < lengthFIFO2 and (not pre(nElementsFIFO1) <
               lengthFIFO1)) then
3022          FIFO2[pre(inPointerFIFO2)] := portIn2.ID;
3023          price2[pre(inPointerFIFO2)] := portIn2.monetaryValue;
```

```
3024            inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
3025            nElementsFIFO2 := pre(nElementsFIFO2) + 1;
3026            nElementsFIFO1 := pre(nElementsFIFO1) - 1;
3027            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                    1;
3028            accumulatedProfits := pre(accumulatedProfits) + portIn2.monetaryValue;
3029            currentBusinessValue := pre(currentBusinessValue) + portIn2.monetaryValue -
                    price1[pre(outPointerFIFO1)];
3030          end if;
3031          if ((pre(nElementsFIFO1) < lengthFIFO1) and (pre(nElementsFIFO2) < lengthFIFO2
                ) and uniformRNG.outPort[1] >= 0.5) then
3032            FIFO1[pre(inPointerFIFO1)] := portIn2.ID;
3033            price1[pre(inPointerFIFO1)] := portIn2.monetaryValue;
3034            inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
3035            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                    1;
3036            accumulatedProfits := pre(accumulatedProfits) + portIn2.monetaryValue;
3037            currentBusinessValue := pre(currentBusinessValue) + portIn2.monetaryValue -
                    price1[pre(outPointerFIFO1)];
3038          end if;
3039          if ((pre(nElementsFIFO2) < lengthFIFO2) and (pre(nElementsFIFO1) < lengthFIFO1
                ) and (not uniformRNG.outPort[1] >= 0.5)) then
3040            FIFO2[pre(inPointerFIFO2)] := portIn2.ID;
3041            price2[pre(inPointerFIFO2)] := portIn2.monetaryValue;
3042            inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
3043            nElementsFIFO2 := pre(nElementsFIFO2) + 1;
3044            nElementsFIFO1 := pre(nElementsFIFO1) - 1;
3045            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                    1;
3046            accumulatedProfits := pre(accumulatedProfits) + portIn2.monetaryValue;
3047            currentBusinessValue := pre(currentBusinessValue) + portIn2.monetaryValue -
                    price1[pre(outPointerFIFO1)];
3048          end if;
3049          if ((not pre(nElementsFIFO1) < lengthFIFO1) and (not pre(nElementsFIFO2) <
                lengthFIFO2)) then
3050            nElementsFIFO1 := pre(nElementsFIFO1) - 1;
3051            moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn2.monetaryValue;
3052            accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
3053            currentBusinessValue := pre(currentBusinessValue) - price1[pre(
                    outPointerFIFO1)];
3054          end if;
3055          /* Reading from FIFO1 */
3056          portOut1.ID := FIFO1[pre(outPointerFIFO1)];
3057          portOut1.monetaryValue := price1[pre(outPointerFIFO1)];
3058          outPointerFIFO1 := mod(pre(outPointerFIFO1), lengthFIFO1) + 1;
3059          /* Setting the flags on the ports */
3060          portIn2.writeOkFlag := true;
3061          portOut1.writeFlag := true;
3062          state := 9;
3063          controlTime := time;
3064        end when;
3065
3066        /* Reading from portIn1 ad writing to portOut2 at the same time */
3067        when (pre(state) == 0 and pre(portIn1.writeFlag) and (not pre(portIn2.writeFlag)
                ) and ((not pre(portOut1.writeOkFlag)) or (not pre(nElementsFIFO1) > 0)) and
                (pre(portOut2.writeOkFlag)) and (pre(nElementsFIFO2) > 0) and time >
                controlTime) then
```

```
3068        if ((pre(nElementsFIFO1) < lengthFIFO1) and (not pre(nElementsFIFO2) <
               lengthFIFO2)) then
3069          FIFO1[pre(inPointerFIFO1)] := portIn1.ID;
3070          price1[pre(inPointerFIFO1)] := portIn1.monetaryValue;
3071          inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
3072          nElementsFIFO1 := pre(nElementsFIFO1) + 1;
3073          nElementsFIFO2 := pre(nElementsFIFO2) - 1;
3074          accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                 1;
3075          accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
3076          currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue -
                 price2[pre(outPointerFIFO2)];
3077        end if;
3078        if (pre(nElementsFIFO2) < lengthFIFO2 and (not pre(nElementsFIFO1) <
               lengthFIFO1)) then
3079          FIFO2[pre(inPointerFIFO2)] := portIn1.ID;
3080          price2[pre(inPointerFIFO2)] := portIn1.monetaryValue;
3081          inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
3082          accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                 1;
3083          accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
3084          currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue -
                 price2[pre(outPointerFIFO2)];
3085        end if;
3086        if ((pre(nElementsFIFO1) < lengthFIFO1) and (pre(nElementsFIFO2) < lengthFIFO2
               ) and uniformRNG.outPort[1] >= 0.5) then
3087          FIFO1[pre(inPointerFIFO1)] := portIn1.ID;
3088          price1[pre(inPointerFIFO1)] := portIn1.monetaryValue;
3089          inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
3090          nElementsFIFO1 := pre(nElementsFIFO1) + 1;
3091          nElementsFIFO2 := pre(nElementsFIFO2) - 1;
3092          accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                 1;
3093          accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
3094          currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue -
                 price2[pre(outPointerFIFO2)];
3095        end if;
3096        if ((pre(nElementsFIFO2) < lengthFIFO2) and (pre(nElementsFIFO1) < lengthFIFO1
               ) and (not uniformRNG.outPort[1] >= 0.5)) then
3097          FIFO2[pre(inPointerFIFO2)] := portIn1.ID;
3098          price2[pre(inPointerFIFO2)] := portIn1.monetaryValue;
3099          inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
3100          accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                 1;
3101          accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
3102          currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue -
                 price2[pre(outPointerFIFO2)];
3103        end if;
3104        if ((not pre(nElementsFIFO1) < lengthFIFO1) and (not pre(nElementsFIFO2) <
               lengthFIFO2)) then
3105          nElementsFIFO2 := pre(nElementsFIFO2) - 1;
3106          moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn1.monetaryValue;
3107          accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
3108          currentBusinessValue := pre(currentBusinessValue) - price2[pre(
                 outPointerFIFO2)];
3109        end if;
3110        /* Reading from FIFO2 */
```

```
3111          portOut2.ID := FIFO2[pre(outPointerFIFO2)];
3112          portOut2.monetaryValue := price2[pre(outPointerFIFO2)];
3113          outPointerFIFO2 := mod(pre(outPointerFIFO2), lengthFIFO2) + 1;
3114          /* Setting the flags on the ports */
3115          portIn1.writeOkFlag := true;
3116          portOut2.writeFlag := true;
3117          state := 4;
3118          controlTime := time;
3119        end when;
3120
3121        /* Reading from portIn2 ad writing to portOut2 at the same time */
3122        when (pre(state) == 0 and pre(portIn2.writeFlag) and (not pre(portIn1.writeFlag)
               ) and ((not pre(portOut1.writeOkFlag)) or (not pre(nElementsFIFO1) > 0)) and
              (pre(portOut2.writeOkFlag)) and (pre(nElementsFIFO2) > 0) and time >
              controlTime) then
3123          if ((pre(nElementsFIFO1) < lengthFIFO1) and (not pre(nElementsFIFO2) <
                 lengthFIFO2)) then
3124            FIFO1[pre(inPointerFIFO1)] := portIn2.ID;
3125            price1[pre(inPointerFIFO1)] := portIn2.monetaryValue;
3126            inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
3127            nElementsFIFO1 := pre(nElementsFIFO1) + 1;
3128            nElementsFIFO2 := pre(nElementsFIFO2) - 1;
3129            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                   1;
3130            accumulatedProfits := pre(accumulatedProfits) + portIn2.monetaryValue;
3131            currentBusinessValue := pre(currentBusinessValue) + portIn2.monetaryValue -
                   price2[pre(outPointerFIFO2)];
3132          end if;
3133          if (pre(nElementsFIFO2) < lengthFIFO2 and (not pre(nElementsFIFO1) <
                 lengthFIFO1)) then
3134            FIFO2[pre(inPointerFIFO2)] := portIn2.ID;
3135            price2[pre(inPointerFIFO2)] := portIn2.monetaryValue;
3136            inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
3137            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                   1;
3138            accumulatedProfits := pre(accumulatedProfits) + portIn2.monetaryValue;
3139            currentBusinessValue := pre(currentBusinessValue) + portIn2.monetaryValue -
                   price2[pre(outPointerFIFO2)];
3140          end if;
3141          if ((pre(nElementsFIFO1) < lengthFIFO1) and (pre(nElementsFIFO2) < lengthFIFO2
                 ) and uniformRNG.outPort[1] >= 0.5) then
3142            FIFO1[pre(inPointerFIFO1)] := portIn2.ID;
3143            price1[pre(inPointerFIFO1)] := portIn2.monetaryValue;
3144            inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
3145            nElementsFIFO1 := pre(nElementsFIFO1) + 1;
3146            nElementsFIFO2 := pre(nElementsFIFO2) - 1;
3147            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                   1;
3148            accumulatedProfits := pre(accumulatedProfits) + portIn2.monetaryValue;
3149            currentBusinessValue := pre(currentBusinessValue) + portIn2.monetaryValue -
                   price2[pre(outPointerFIFO2)];
3150          end if;
3151          if ((pre(nElementsFIFO2) < lengthFIFO2) and (pre(nElementsFIFO1) < lengthFIFO1
                 ) and (not uniformRNG.outPort[1] >= 0.5)) then
3152            FIFO2[pre(inPointerFIFO2)] := portIn2.ID;
3153            price2[pre(inPointerFIFO2)] := portIn2.monetaryValue;
3154            inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
```

```
3155            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                    1;
3156            accumulatedProfits := pre(accumulatedProfits) + portIn2.monetaryValue;
3157            currentBusinessValue := pre(currentBusinessValue) + portIn2.monetaryValue -
                    price2[pre(outPointerFIFO2)];
3158          end if;
3159          if ((not pre(nElementsFIFO1) < lengthFIFO1) and (not pre(nElementsFIFO2) <
                 lengthFIFO2)) then
3160            nElementsFIFO2 := pre(nElementsFIFO2) - 1;
3161            moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn2.monetaryValue;
3162            accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
3163            currentBusinessValue := pre(currentBusinessValue) - price2[pre(
                    outPointerFIFO2)];
3164          end if;
3165          /* Reading from FIFO2 */
3166          portOut2.ID := FIFO2[pre(outPointerFIFO2)];
3167          portOut2.monetaryValue := price2[pre(outPointerFIFO2)];
3168          outPointerFIFO2 := mod(pre(outPointerFIFO2), lengthFIFO2) + 1;
3169          /* Setting the flags on the ports */
3170          portIn2.writeOkFlag := true;
3171          portOut2.writeFlag := true;
3172          state := 10;
3173          controlTime := time;
3174        end when;
3175
3176        /* Writing to portOut1 */
3177        when (pre(state) == 0 and (not pre(portIn1.writeFlag)) and (not pre(portIn2.
                writeFlag)) and pre(portOut1.writeOkFlag) and (pre(nElementsFIFO1) > 0) and
                ((not pre(portOut2.writeOkFlag)) or (not pre(nElementsFIFO2) > 0)) and time >
                 controlTime) then
3178          /* Reading from FIFO1 */
3179          portOut1.ID := FIFO1[pre(outPointerFIFO1)];
3180          portOut1.monetaryValue := price1[pre(outPointerFIFO1)];
3181          outPointerFIFO1 := mod(pre(outPointerFIFO1), lengthFIFO1) + 1;
3182          nElementsFIFO1 := pre(nElementsFIFO1) - 1;
3183          currentBusinessValue := pre(currentBusinessValue) - price1[pre(outPointerFIFO1
                    )];
3184          /* Setting the flags on the ports */
3185          portOut1.writeFlag := true;
3186          state := 5;
3187          controlTime := time;
3188        end when;
3189
3190        /* Writing to portOut2 */
3191        when (pre(state) == 0 and (not pre(portIn1.writeFlag)) and (not pre(portIn2.
                writeFlag)) and ((not pre(portOut1.writeOkFlag)) or (not pre(nElementsFIFO1)
                > 0)) and (pre(portOut2.writeOkFlag)) and (pre(nElementsFIFO2) > 0) and time
                > controlTime) then
3192          /* Reading from FIFO1 */
3193          portOut2.ID := FIFO2[pre(outPointerFIFO2)];
3194          portOut2.monetaryValue := price2[pre(outPointerFIFO2)];
3195          outPointerFIFO2 := mod(pre(outPointerFIFO2), lengthFIFO2) + 1;
3196          nElementsFIFO2 := pre(nElementsFIFO2) - 1;
3197          currentBusinessValue := pre(currentBusinessValue) - price2[pre(outPointerFIFO2
                    )];
3198          /* Setting the flags on the ports */
3199          portOut2.writeFlag := true;
```

```
3200          state := 6;
3201          controlTime := time;
3202        end when;
3203
3204        /* Reading from portIn1 ad writing to portOut1 and portOut2 at the same time */
3205        when (pre(state) == 0 and pre(portIn1.writeFlag) and (not pre(portIn2.writeFlag)
                ) and pre(portOut1.writeOkFlag) and (pre(nElementsFIFO1) > 0) and pre(
                portOut2.writeOkFlag) and pre(nElementsFIFO2) > 0 and time > controlTime)
                then
3206          if ((pre(nElementsFIFO1) < lengthFIFO1) and (not pre(nElementsFIFO2) <
                  lengthFIFO2)) then
3207            FIFO1[pre(inPointerFIFO1)] := portIn1.ID;
3208            price1[pre(inPointerFIFO1)] := portIn1.monetaryValue;
3209            inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
3210            nElementsFIFO2 := pre(nElementsFIFO2) - 1;
3211            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                    1;
3212            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
3213            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue -
                    price1[pre(outPointerFIFO1)] - price2[pre(outPointerFIFO2)];
3214          end if;
3215          if (pre(nElementsFIFO2) < lengthFIFO2 and (not pre(nElementsFIFO1) <
                  lengthFIFO1)) then
3216            FIFO2[pre(inPointerFIFO2)] := portIn1.ID;
3217            price2[pre(inPointerFIFO2)] := portIn1.monetaryValue;
3218            inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
3219            nElementsFIFO1 := pre(nElementsFIFO1) - 1;
3220            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                    1;
3221            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
3222            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue -
                    price1[pre(outPointerFIFO1)] - price2[pre(outPointerFIFO2)];
3223          end if;
3224          if ((pre(nElementsFIFO1) < lengthFIFO1) and (pre(nElementsFIFO2) < lengthFIFO2
                ) and uniformRNG.outPort[1] >= 0.5) then
3225            FIFO1[pre(inPointerFIFO1)] := portIn1.ID;
3226            price1[pre(inPointerFIFO1)] := portIn1.monetaryValue;
3227            inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
3228            nElementsFIFO2 := pre(nElementsFIFO2) - 1;
3229            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                    1;
3230            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
3231            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue -
                    price1[pre(outPointerFIFO1)] - price2[pre(outPointerFIFO2)];
3232          end if;
3233          if ((pre(nElementsFIFO2) < lengthFIFO2) and (pre(nElementsFIFO1) < lengthFIFO1
                ) and (not uniformRNG.outPort[1] >= 0.5)) then
3234            FIFO2[pre(inPointerFIFO2)] := portIn1.ID;
3235            price2[pre(inPointerFIFO2)] := portIn1.monetaryValue;
3236            inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
3237            nElementsFIFO1 := pre(nElementsFIFO1) - 1;
3238            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                    1;
3239            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
3240            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue -
                    price1[pre(outPointerFIFO1)] - price2[pre(outPointerFIFO2)];
3241          end if;
```

```
3242          if ((not pre(nElementsFIFO1) < lengthFIFO1) and (not pre(nElementsFIFO2) <
                  lengthFIFO2)) then
3243            nElementsFIFO1 := pre(nElementsFIFO1) - 1;
3244            nElementsFIFO2 := pre(nElementsFIFO2) - 1;
3245            moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn1.monetaryValue;
3246            accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
3247            currentBusinessValue := pre(currentBusinessValue) - price1[pre(
                  outPointerFIFO1)] - price2[pre(outPointerFIFO2)];
3248          end if;
3249          /* Reading from FIFO1 */
3250          portOut1.ID := FIFO1[pre(outPointerFIFO1)];
3251          portOut1.monetaryValue := price1[pre(outPointerFIFO1)];
3252          outPointerFIFO1 := mod(pre(outPointerFIFO1), lengthFIFO1) + 1;
3253          /* Reading from FIFO2 */
3254          portOut2.ID := FIFO2[pre(outPointerFIFO2)];
3255          portOut2.monetaryValue := price2[pre(outPointerFIFO2)];
3256          outPointerFIFO2 := mod(pre(outPointerFIFO2), lengthFIFO2) + 1;
3257          /* Setting the flags on the ports */
3258          portIn1.writeOkFlag := true;
3259          portOut1.writeFlag := true;
3260          portOut2.writeFlag := true;
3261          state := 7;
3262          controlTime := time;
3263        end when;
3264
3265        /* Reading from portIn2 ad writing to portOut1 and portOut2 at the same time */
3266        when (pre(state) == 0 and pre(portIn2.writeFlag) and (not pre(portIn1.writeFlag)
              ) and pre(portOut1.writeOkFlag) and (pre(nElementsFIFO1) > 0) and pre(
              portOut2.writeOkFlag) and pre(nElementsFIFO2) > 0 and time > controlTime)
              then
3267          if ((pre(nElementsFIFO1) < lengthFIFO1) and (not pre(nElementsFIFO2) <
                  lengthFIFO2)) then
3268            FIFO1[pre(inPointerFIFO1)] := portIn2.ID;
3269            price1[pre(inPointerFIFO1)] := portIn2.monetaryValue;
3270            inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
3271            nElementsFIFO2 := pre(nElementsFIFO2) - 1;
3272            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                    1;
3273            accumulatedProfits := pre(accumulatedProfits) + portIn2.monetaryValue;
3274            currentBusinessValue := pre(currentBusinessValue) + portIn2.monetaryValue -
                    price1[pre(outPointerFIFO1)] - price2[pre(outPointerFIFO2)];
3275          end if;
3276          if (pre(nElementsFIFO2) < lengthFIFO2 and (not pre(nElementsFIFO1) <
                  lengthFIFO1)) then
3277            FIFO2[pre(inPointerFIFO2)] := portIn2.ID;
3278            price2[pre(inPointerFIFO2)] := portIn2.monetaryValue;
3279            inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
3280            nElementsFIFO1 := pre(nElementsFIFO1) - 1;
3281            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                    1;
3282            accumulatedProfits := pre(accumulatedProfits) + portIn2.monetaryValue;
3283            currentBusinessValue := pre(currentBusinessValue) + portIn2.monetaryValue -
                    price1[pre(outPointerFIFO1)] - price2[pre(outPointerFIFO2)];
3284          end if;
3285          if ((pre(nElementsFIFO1) < lengthFIFO1) and (pre(nElementsFIFO2) < lengthFIFO2
                ) and uniformRNG.outPort[1] >= 0.5) then
3286            FIFO1[pre(inPointerFIFO1)] := portIn2.ID;
```

```
3287            price1[pre(inPointerFIFO1)] := portIn2.monetaryValue;
3288            inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
3289            nElementsFIFO2 := pre(nElementsFIFO2) - 1;
3290            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                    1;
3291            accumulatedProfits := pre(accumulatedProfits) + portIn2.monetaryValue;
3292            currentBusinessValue := pre(currentBusinessValue) + portIn2.monetaryValue -
                    price1[pre(outPointerFIFO1)] - price2[pre(outPointerFIFO2)];
3293          end if;
3294          if ((pre(nElementsFIFO2) < lengthFIFO2) and (pre(nElementsFIFO1) < lengthFIFO1
                ) and (not uniformRNG.outPort[1] >= 0.5)) then
3295            FIFO2[pre(inPointerFIFO2)] := portIn2.ID;
3296            price2[pre(inPointerFIFO2)] := portIn2.monetaryValue;
3297            inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
3298            nElementsFIFO1 := pre(nElementsFIFO1) - 1;
3299            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                    1;
3300            accumulatedProfits := pre(accumulatedProfits) + portIn2.monetaryValue;
3301            currentBusinessValue := pre(currentBusinessValue) + portIn2.monetaryValue -
                    price1[pre(outPointerFIFO1)] - price2[pre(outPointerFIFO2)];
3302          end if;
3303          if ((not pre(nElementsFIFO1) < lengthFIFO1) and (not pre(nElementsFIFO2) <
                lengthFIFO2)) then
3304            nElementsFIFO1 := pre(nElementsFIFO1) - 1;
3305            nElementsFIFO2 := pre(nElementsFIFO2) - 1;
3306            moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn2.monetaryValue;
3307            accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
3308            currentBusinessValue := pre(currentBusinessValue) - price1[pre(
                    outPointerFIFO1)] - price2[pre(outPointerFIFO2)];
3309          end if;
3310          /* Reading from FIFO1 */
3311          portOut1.ID := FIFO1[pre(outPointerFIFO1)];
3312          portOut1.monetaryValue := price1[pre(outPointerFIFO1)];
3313          outPointerFIFO1 := mod(pre(outPointerFIFO1), lengthFIFO1) + 1;
3314          /* Reading from FIFO2 */
3315          portOut2.ID := FIFO2[pre(outPointerFIFO2)];
3316          portOut2.monetaryValue := price2[pre(outPointerFIFO2)];
3317          outPointerFIFO2 := mod(pre(outPointerFIFO2), lengthFIFO2) + 1;
3318          /* Setting the flags on the ports */
3319          portIn2.writeOkFlag := true;
3320          portOut1.writeFlag := true;
3321          portOut2.writeFlag := true;
3322          state := 11;
3323          controlTime := time;
3324        end when;
3325
3326        /* Reading from port1 and portIn2 ad writing to portOut1 and portOut2 at the
               same time */
3327        when (pre(state) == 0 and pre(portIn2.writeFlag) and pre(portIn1.writeFlag) and
               pre(portOut1.writeOkFlag) and (pre(nElementsFIFO1) > 0) and pre(portOut2.
               writeOkFlag) and pre(nElementsFIFO2) > 0 and time > controlTime) then
3328        if ((pre(nElementsFIFO1) < lengthFIFO1) and (not pre(nElementsFIFO2) <
               lengthFIFO2)) then
3329            // Adding port1.ID
3330            FIFO1[pre(inPointerFIFO1)] := portIn1.ID;
3331            price1[pre(inPointerFIFO1)] := portIn1.monetaryValue;
3332            nElementsFIFO2 := pre(nElementsFIFO2) - 1;
```

```
3333            // Adding port2.ID if there is still space
3334            if ((pre(nElementsFIFO1) + 1) < lengthFIFO1) then
3335              FIFO1[pre(inPointerFIFO1)+1] := portIn2.ID;
3336              price1[pre(inPointerFIFO1)+1] := portIn2.monetaryValue;
3337              //inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 2;
3338              if (pre(nElementsFIFO1) == (lengthFIFO1 -2)) then
3339                inPointerFIFO1 := 1;
3340              else
3341                inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 2;
3342              end if;
3343              nElementsFIFO1 := pre(nElementsFIFO1) + 1;
3344              accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                     + 2;
3345              accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue +
                     portIn2.monetaryValue;
3346              currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                     + portIn2.monetaryValue - price1[pre(outPointerFIFO1)] - price2[pre(
                     outPointerFIFO2)];
3347            else
3348              inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
3349              accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                     + 1;
3350              moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn2.monetaryValue;
3351              accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
3352              accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
3353              currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                     - price1[pre(outPointerFIFO1)] - price2[pre(outPointerFIFO2)];
3354            end if;
3355          end if;
3356          if (pre(nElementsFIFO2) < lengthFIFO2 and (not pre(nElementsFIFO1) <
                 lengthFIFO1)) then
3357            // Adding port1.ID
3358            FIFO2[pre(inPointerFIFO2)] := portIn1.ID;
3359            price2[pre(inPointerFIFO2)] := portIn1.monetaryValue;
3360            nElementsFIFO1 := pre(nElementsFIFO1) - 1;
3361            // Adding port2.ID if there is still space
3362            if ((pre(nElementsFIFO2) + 1) < lengthFIFO2) then
3363              FIFO2[pre(inPointerFIFO2)+1] := portIn2.ID;
3364              price2[pre(inPointerFIFO2)+1] := portIn2.monetaryValue;
3365              //inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 2;
3366              if (pre(nElementsFIFO2) == (lengthFIFO2 -2)) then
3367                inPointerFIFO2 := 1;
3368              else
3369                inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 2;
3370              end if;
3371              nElementsFIFO2 := pre(nElementsFIFO2) + 1;
3372              accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                     + 2;
3373              accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue +
                     portIn2.monetaryValue;
3374              currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                     + portIn2.monetaryValue - price1[pre(outPointerFIFO1)] - price2[pre(
                     outPointerFIFO2)];
3375            else
3376              inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
3377              accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                     + 1;
```

```
3378            moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn2.monetaryValue;
3379            accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
3380            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
3381            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                    - price1[pre(outPointerFIFO1)] - price2[pre(outPointerFIFO2)];
3382          end if;
3383        end if;
3384        // If the two queues are not full, nondeterministic choice
3385        if ((pre(nElementsFIFO1) < lengthFIFO1) and (pre(nElementsFIFO2) < lengthFIFO2
               ) and uniformRNG.outPort[1] >= 0.5) then
3386          FIFO1[pre(inPointerFIFO1)] := portIn1.ID;
3387          price1[pre(inPointerFIFO1)] := portIn1.monetaryValue;
3388          if ((uniformRNG.outPort[1] >= 0.75) and (pre(nElementsFIFO1) + 1 <
                   lengthFIFO1)) then
3389            FIFO1[pre(inPointerFIFO1)+1] := portIn2.ID;
3390            price1[pre(inPointerFIFO1)+1] := portIn2.monetaryValue;
3391            //inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 2;
3392            if (pre(nElementsFIFO1) == (lengthFIFO1 -2)) then
3393              inPointerFIFO1 := 1;
3394            else
3395              inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 2;
3396            end if;
3397            nElementsFIFO1 := pre(nElementsFIFO1) + 1;
3398            nElementsFIFO2 := pre(nElementsFIFO2) - 1;
3399            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                       + 2;
3400            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue +
                       portIn2.monetaryValue;
3401            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                       + portIn2.monetaryValue - price1[pre(outPointerFIFO1)] - price2[pre(
                       outPointerFIFO2)];
3402          else
3403            FIFO2[pre(inPointerFIFO2)] := portIn2.ID;
3404            price2[pre(inPointerFIFO2)] := portIn2.monetaryValue;
3405            inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
3406            inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
3407            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                       + 2;
3408            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue +
                       portIn2.monetaryValue;
3409            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                       + portIn2.monetaryValue - price1[pre(outPointerFIFO1)] - price2[pre(
                       outPointerFIFO2)];
3410          end if;
3411        end if;
3412        if ((pre(nElementsFIFO2) < lengthFIFO2) and (pre(nElementsFIFO1) < lengthFIFO1
               ) and (not uniformRNG.outPort[1] >= 0.5)) then
3413          FIFO2[pre(inPointerFIFO2)] := portIn1.ID;
3414          price2[pre(inPointerFIFO2)] := portIn1.monetaryValue;
3415          if ((uniformRNG.outPort[1] <= 0.25) and (pre(nElementsFIFO2) + 1 <
                   lengthFIFO2)) then
3416            FIFO2[pre(inPointerFIFO2)+1] := portIn2.ID;
3417            price2[pre(inPointerFIFO2)+1] := portIn2.monetaryValue;
3418            //inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 2;
3419            if (pre(nElementsFIFO2) == (lengthFIFO2 -2)) then
3420              inPointerFIFO2 := 1;
3421            else
```

```
3422              inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 2;
3423            end if;
3424            nElementsFIFO2 := pre(nElementsFIFO2) + 1;
3425            nElementsFIFO1 := pre(nElementsFIFO1) - 1;
3426            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 2;
3427            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue +
                    portIn2.monetaryValue;
3428            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                    + portIn2.monetaryValue - price1[pre(outPointerFIFO1)] - price2[pre(
                    outPointerFIFO2)];
3429          else
3430            FIFO1[pre(inPointerFIFO1)] := portIn2.ID;
3431            price1[pre(inPointerFIFO1)] := portIn2.monetaryValue;
3432            inPointerFIFO1 := mod(pre(inPointerFIFO1), lengthFIFO1) + 1;
3433            inPointerFIFO2 := mod(pre(inPointerFIFO2), lengthFIFO2) + 1;
3434            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents)
                    + 2;
3435            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue +
                    portIn2.monetaryValue;
3436            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue
                    + portIn2.monetaryValue - price1[pre(outPointerFIFO1)] - price2[pre(
                    outPointerFIFO2)];
3437          end if;
3438        end if;
3439        if (not (pre(nElementsFIFO1) < lengthFIFO1) and (not pre(nElementsFIFO2) <
                lengthFIFO2)) then
3440          nElementsFIFO1 := pre(nElementsFIFO1) - 1;
3441          nElementsFIFO2 := pre(nElementsFIFO2) - 1;
3442          moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn1.monetaryValue +
                  portIn2.monetaryValue;
3443          accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 2;
3444          currentBusinessValue := pre(currentBusinessValue) - price1[pre(
                  outPointerFIFO1)] - price2[pre(outPointerFIFO2)];
3445        end if;
3446        /* Reading from FIFO1 */
3447        portOut1.ID := FIFO1[pre(outPointerFIFO1)];
3448        portOut1.monetaryValue := price1[pre(outPointerFIFO1)];
3449        outPointerFIFO1 := mod(pre(outPointerFIFO1), lengthFIFO1) + 1;
3450        /* Reading from FIFO2 */
3451        portOut2.ID := FIFO2[pre(outPointerFIFO2)];
3452        portOut2.monetaryValue := price2[pre(outPointerFIFO2)];
3453        outPointerFIFO2 := mod(pre(outPointerFIFO2), lengthFIFO2) + 1;
3454        /* Setting the flags on the ports */
3455        portIn2.writeOkFlag := true;
3456        portOut1.writeFlag := true;
3457        portOut2.writeFlag := true;
3458        state := 15;
3459        controlTime := time;
3460      end when;
3461
3462      when (pre(state) == 1 and (not pre(portIn1.writeFlag)) and time > controlTime)
            then
3463        portIn1.writeOkFlag := false;
3464        state := 0;
3465        controlTime := time;
3466      end when;
```

```
3467
3468        when (pre(state) == 2 and (not pre(portOut1.writeOkFlag)) and (not pre(portOut2.
                writeOkFlag)) and time > controlTime) then
3469          portOut1.writeFlag := false;
3470          portOut2.writeFlag := false;
3471          state := 0;
3472          controlTime := time;
3473        end when;
3474
3475        when (pre(state) == 3 and (not pre(portIn1.writeFlag)) and (not pre(portOut1.
                writeOkFlag)) and time > controlTime) then
3476          portIn1.writeOkFlag := false;
3477          portOut1.writeFlag := false;
3478          state := 0;
3479          controlTime := time;
3480        end when;
3481
3482        when (pre(state) == 4 and (not pre(portIn1.writeFlag)) and (not pre(portOut2.
                writeOkFlag)) and time > controlTime) then
3483          portIn1.writeOkFlag := false;
3484          portOut2.writeFlag := false;
3485          state := 0;
3486          controlTime := time;
3487        end when;
3488
3489        when (pre(state) == 5 and (not pre(portOut1.writeOkFlag)) and time > controlTime
                ) then
3490          portOut1.writeFlag := false;
3491          state := 0;
3492          controlTime := time;
3493        end when;
3494
3495        when (pre(state) == 6 and (not pre(portOut2.writeOkFlag)) and time > controlTime
                ) then
3496          portOut2.writeFlag := false;
3497          state := 0;
3498          controlTime := time;
3499        end when;
3500
3501        when (pre(state) == 7 and (not pre(portIn1.writeFlag)) and (not pre(portOut1.
                writeOkFlag)) and (not pre(portOut2.writeOkFlag)) and time > controlTime)
                then
3502          portIn1.writeOkFlag := false;
3503          portOut1.writeFlag := false;
3504          portOut2.writeFlag := false;
3505          state := 0;
3506          controlTime := time;
3507        end when;
3508
3509        when (pre(state) == 8 and (not pre(portIn2.writeFlag)) and time > controlTime)
                then
3510          portIn2.writeOkFlag := false;
3511          state := 0;
3512          controlTime := time;
3513           end when;
3514
```

```
3515        when (pre(state) == 9 and (not pre(portIn2.writeFlag)) and (not pre(portOut1.
                writeOkFlag)) and time > controlTime) then
3516          portIn2.writeOkFlag := false;
3517          portOut1.writeFlag := false;
3518          state := 0;
3519          controlTime := time;
3520        end when;
3521
3522        when (pre(state) == 10 and (not pre(portIn2.writeFlag)) and (not pre(portOut2.
                writeOkFlag)) and time > controlTime) then
3523          portIn2.writeOkFlag := false;
3524          portOut2.writeFlag := false;
3525          state := 0;
3526          controlTime := time;
3527        end when;
3528
3529        when (pre(state) == 11 and (not pre(portIn2.writeFlag)) and (not pre(portOut1.
                writeOkFlag)) and (not pre(portOut2.writeOkFlag)) and time > controlTime)
                then
3530          portIn2.writeOkFlag := false;
3531          portOut1.writeFlag := false;
3532          portOut2.writeFlag := false;
3533          state := 0;
3534          controlTime := time;
3535        end when;
3536
3537        when (pre(state) == 12 and (not pre(portIn1.writeFlag)) and (not pre(portIn2.
                writeFlag)) and time > controlTime) then
3538          portIn1.writeOkFlag := false;
3539          portIn2.writeOkFlag := false;
3540          state := 0;
3541          controlTime := time;
3542        end when;
3543
3544        when (pre(state) == 13 and (not pre(portIn1.writeFlag)) and (not pre(portIn2.
                writeFlag)) and (not pre(portOut1.writeOkFlag)) and time > controlTime) then
3545          portIn1.writeOkFlag := false;
3546          portIn2.writeOkFlag := false;
3547          portOut1.writeFlag := false;
3548          state := 0;
3549          controlTime := time;
3550        end when;
3551
3552        when (pre(state) == 14 and (not pre(portIn1.writeFlag)) and (not pre(portIn2.
                writeFlag)) and (not pre(portOut2.writeOkFlag)) and time > controlTime) then
3553          portIn1.writeOkFlag := false;
3554          portIn2.writeOkFlag := false;
3555          portOut2.writeFlag := false;
3556          state := 0;
3557          controlTime := time;
3558           end when;
3559
3560        when (pre(state) == 15 and (not pre(portIn1.writeFlag)) and (not pre(portIn2.
                writeFlag)) and (not pre(portOut1.writeOkFlag)) and (not pre(portOut2.
                writeOkFlag)) and time > controlTime) then
3561          portIn1.writeOkFlag := false;
3562          portIn2.writeOkFlag := false;
```

```
3563        portOut1.writeFlag := false;
3564        portOut2.writeFlag := false;
3565        state := 0;
3566        controlTime := time;
3567      end when;
3568    annotation (Icon(graphics={
3569            Ellipse(
3570              extent={{80,60},{-80,-60}},
3571              lineColor={0,0,0},
3572              lineThickness=0.5),
3573            Text(
3574              extent={{-36,-28},{40,-48}},
3575              lineColor={0,0,0},
3576              lineThickness=0.5,
3577              fillPattern=FillPattern.Solid,
3578             textString="circuit5"),
3579           Bitmap(
3580             extent={{-74,38},{72,-28}},
3581             imageSource=
3582                 "/*Put your image*/",
3583             fileName="modelica://RiskAnalysis/reoDymola.png")}),
3584                                         Diagram(graphics));
3585  end ReoCircuit5;
3586
3587  model ReoCircuit6
3588
3589    parameter Integer lengthFIFO = 100;
3590    Integer inPointerFIFO(start=1);
3591    Integer outPointerFIFO(start=1);
3592    Integer nElementsFIFO(start=0);
3593    Integer moneyLossTotalFIFO(start=0);
3594    Integer state(start=0);
3595    Real controlTime(start=0);
3596
3597    Real freeCapacity(start=0);
3598    Real usedCapacity(start=0);
3599    Integer accumulatedLosses(start=0);
3600    Integer accumulatedNumberOfLostEvents(start=0);
3601    Integer accumulatedNumberOfSuccessEvents(start=0);
3602    Integer accumulatedProfits(start=0);
3603    Integer currentBusinessValue(start=0);
3604
3605  public
3606    ServicePortIn portIn2 annotation (Placement(transformation(extent={{-92,-42},{
3607            -72,-22}}), iconTransformation(extent={{-92,-42},{-72,-22}})));
3608    ServicePortIn portIn1 annotation (Placement(transformation(extent={{-92,28},{-72,
3609            48}}), iconTransformation(extent={{-92,28},{-72,48}})));
3610    ServicePortOut portOut annotation (Placement(transformation(extent={{76,-6},{96,
3611            14}}), iconTransformation(extent={{76,-6},{96,14}})));
3612    Modelica.Blocks.Interfaces.RealOutput currentFIFOCapacity annotation (
3613        Placement(transformation(extent={{-16,-42},{4,-22}}),
3614          iconTransformation(
3615          extent={{-10,-10},{10,10}},
3616          rotation=-90,
3617          origin={-22,-64})));
3618    Modelica.Blocks.Interfaces.IntegerOutput outAccumulatedLosses annotation (
3619        Placement(transformation(extent={{4,-60},{24,-40}}),
```

```
3620            iconTransformation(
3621            extent={{-10,-10},{10,10}},
3622            rotation=-90,
3623            origin={18,-64})));
3624    equation
3625      usedCapacity =  (nElementsFIFO * 100) / (lengthFIFO);
3626      freeCapacity =  100 - usedCapacity;
3627      accumulatedLosses = moneyLossTotalFIFO;
3628      currentFIFOCapacity = usedCapacity;
3629      outAccumulatedLosses = accumulatedLosses;
3630
3631    protected
3632       Integer[lengthFIFO] FIFO;
3633       Integer[lengthFIFO] price;
3634
3635    algorithm
3636      /* The circuit receives data in input from portIn1 */
3637      when (pre(state) == 0 and pre(portIn1.writeFlag) and (not pre(portIn2.writeFlag))
           and ((not pre(portOut.writeOkFlag)) or (not pre(nElementsFIFO) > 0)) and time
           > controlTime) then
3638        if (pre(nElementsFIFO) < lengthFIFO) then
3639          FIFO[pre(inPointerFIFO)] := portIn1.ID;
3640          price[pre(inPointerFIFO)] := portIn1.monetaryValue;
3641          inPointerFIFO := mod(pre(inPointerFIFO), lengthFIFO) + 1;
3642          nElementsFIFO := pre(nElementsFIFO) + 1;
3643          accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
               1;
3644          accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
3645          currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue;
3646        else
3647          moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn1.monetaryValue;
3648          accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
3649        end if;
3650        portIn1.writeOkFlag := true;
3651        state := 1;
3652        controlTime := time;
3653      end when;
3654
3655      /* The circuit receives data in input from portIn2 */
3656      when (pre(state) == 0 and (not pre(portIn1.writeFlag)) and pre(portIn2.writeFlag)
           and ((not pre(portOut.writeOkFlag)) or (not pre(nElementsFIFO) > 0)) and time
           > controlTime) then
3657        if (pre(nElementsFIFO) < lengthFIFO) then
3658          FIFO[pre(inPointerFIFO)] := portIn2.ID;
3659          price[pre(inPointerFIFO)] := portIn2.monetaryValue;
3660          inPointerFIFO := mod(pre(inPointerFIFO), lengthFIFO) + 1;
3661          nElementsFIFO := pre(nElementsFIFO) + 1;
3662          accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
               1;
3663          accumulatedProfits := pre(accumulatedProfits) + portIn2.monetaryValue;
3664          currentBusinessValue := pre(currentBusinessValue) + portIn2.monetaryValue;
3665        else
3666          moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn2.monetaryValue;
3667          accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
3668        end if;
3669        portIn2.writeOkFlag := true;
3670        state := 2;
```

```
3671        controlTime := time;
3672      end when;
3673
3674      /* The circuit receives data in input from portIn1 and from portIn2 */
3675      when (pre(state) == 0 and pre(portIn1.writeFlag) and pre(portIn2.writeFlag) and
              ((not pre(portOut.writeOkFlag)) or (not pre(nElementsFIFO) > 0)) and time >
             controlTime) then
3676        if (pre(nElementsFIFO) < lengthFIFO) then
3677          FIFO[pre(inPointerFIFO)] := portIn1.ID;
3678          price[pre(inPointerFIFO)] := portIn1.monetaryValue;
3679          if ((pre(nElementsFIFO) + 1) < lengthFIFO) then
3680            FIFO[pre(inPointerFIFO)+1] := portIn2.ID;
3681            price[pre(inPointerFIFO)+1] := portIn2.monetaryValue;
3682            if (pre(nElementsFIFO) == (lengthFIFO -2)) then
3683              inPointerFIFO := 1;
3684            else
3685              inPointerFIFO := mod(pre(inPointerFIFO), lengthFIFO) + 2;
3686            end if;
3687            nElementsFIFO := pre(nElementsFIFO) + 2;
3688            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                   2;
3689            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue +
                  portIn2.monetaryValue;
3690            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue +
                   portIn1.monetaryValue;
3691          else
3692            inPointerFIFO := mod(pre(inPointerFIFO), lengthFIFO) + 1;
3693            nElementsFIFO := pre(nElementsFIFO) + 1;
3694            moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn2.monetaryValue;
3695            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                   1;
3696            accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
3697            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
3698            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue;
3699          end if;
3700        else
3701          moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn1.monetaryValue +
                portIn2.monetaryValue;
3702          accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 2;
3703        end if;
3704        portIn1.writeOkFlag := true;
3705        portIn2.writeOkFlag := true;
3706        state := 3;
3707        controlTime := time;
3708      end when;
3709
3710      /* The circuit receives data in input from portIn1 and writes to portOut */
3711      when (pre(state) == 0 and pre(portIn1.writeFlag) and (not pre(portIn2.writeFlag))
             and pre(portOut.writeOkFlag) and pre(nElementsFIFO) > 0 and time >
            controlTime) then
3712        if (pre(nElementsFIFO) < lengthFIFO) then
3713          FIFO[pre(inPointerFIFO)] := portIn1.ID;
3714          price[pre(inPointerFIFO)] := portIn1.monetaryValue;
3715          inPointerFIFO := mod(pre(inPointerFIFO), lengthFIFO) + 1;
3716          accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                 1;
3717          accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
```

```
3718          currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue -
                  price[pre(outPointerFIFO)];
3719        else
3720          nElementsFIFO := pre(nElementsFIFO) - 1;
3721          moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn1.monetaryValue;
3722          accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
3723          currentBusinessValue := pre(currentBusinessValue) - price[pre(outPointerFIFO)
                  ];
3724        end if;
3725        portOut.ID := FIFO[pre(outPointerFIFO)];
3726        portOut.monetaryValue := price[pre(outPointerFIFO)];
3727        outPointerFIFO := mod(pre(outPointerFIFO), lengthFIFO) + 1;
3728        portIn1.writeOkFlag := true;
3729        portOut.writeFlag:= true;
3730        state := 4;
3731        controlTime := time;
3732      end when;
3733
3734      /* The circuit receives data in input from portIn2 and writes to portOut */
3735      when (pre(state) == 0 and (not pre(portIn1.writeFlag)) and pre(portIn2.writeFlag)
             and pre(portOut.writeOkFlag) and pre(nElementsFIFO) > 0 and time >
            controlTime) then
3736        if (pre(nElementsFIFO) < lengthFIFO) then
3737          FIFO[pre(inPointerFIFO)] := portIn2.ID;
3738          price[pre(inPointerFIFO)] := portIn2.monetaryValue;
3739          inPointerFIFO := mod(pre(inPointerFIFO), lengthFIFO) + 1;
3740          accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                  1;
3741          accumulatedProfits := pre(accumulatedProfits) + portIn2.monetaryValue;
3742          currentBusinessValue := pre(currentBusinessValue) + portIn2.monetaryValue -
                  price[pre(outPointerFIFO)];
3743        else
3744          nElementsFIFO := pre(nElementsFIFO) - 1;
3745          moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn2.monetaryValue;
3746          accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
3747          currentBusinessValue := pre(currentBusinessValue) - price[pre(outPointerFIFO)
                  ];
3748        end if;
3749        portOut.ID := FIFO[pre(outPointerFIFO)];
3750        portOut.monetaryValue := price[pre(outPointerFIFO)];
3751        outPointerFIFO := mod(pre(outPointerFIFO), lengthFIFO) + 1;
3752        portIn2.writeOkFlag := true;
3753        portOut.writeFlag:= true;
3754        state := 5;
3755        controlTime := time;
3756      end when;
3757
3758      /* The circuit  writes to portOut */
3759      when (pre(state) == 0 and (not pre(portIn1.writeFlag)) and (not pre(portIn2.
             writeFlag)) and pre(portOut.writeOkFlag) and pre(nElementsFIFO) > 0 and time >
            controlTime) then
3760        portOut.ID := FIFO[pre(outPointerFIFO)];
3761        portOut.monetaryValue := price[pre(outPointerFIFO)];
3762        outPointerFIFO := mod(pre(outPointerFIFO), lengthFIFO) + 1;
3763        nElementsFIFO := pre(nElementsFIFO) - 1;
3764        currentBusinessValue := pre(currentBusinessValue) - price[pre(outPointerFIFO)];
3765        portOut.writeFlag:= true;
```

```
3766        state := 6;
3767        controlTime := time;
3768      end when;
3769
3770      /* The circuit receives data in input from portIn1 and from portIn2 and writes to
               portOut */
3771      when (pre(state) == 0 and pre(portIn1.writeFlag) and pre(portIn2.writeFlag) and
             pre(portOut.writeOkFlag) and pre(nElementsFIFO) > 0 and time > controlTime)
             then
3772        if (pre(nElementsFIFO) < lengthFIFO) then
3773          FIFO[pre(inPointerFIFO)] := portIn1.ID;
3774          price[pre(inPointerFIFO)] := portIn1.monetaryValue;
3775          if ((pre(nElementsFIFO) + 1) < lengthFIFO) then
3776            FIFO[pre(inPointerFIFO)+1] := portIn2.ID;
3777            price[pre(inPointerFIFO)+1] := portIn2.monetaryValue;
3778            if (pre(nElementsFIFO) == (lengthFIFO -2)) then
3779              inPointerFIFO := 1;
3780            else
3781              inPointerFIFO := mod(pre(inPointerFIFO), lengthFIFO) + 2;
3782            end if;
3783            nElementsFIFO := pre(nElementsFIFO) + 1;
3784            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                   2;
3785            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue +
                 portIn2.monetaryValue;
3786            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue +
                   portIn2.monetaryValue - price[pre(outPointerFIFO)];
3787          else
3788            inPointerFIFO := mod(pre(inPointerFIFO), lengthFIFO) + 1;
3789            moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn2.monetaryValue;
3790            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                   1;
3791            accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
3792            accumulatedProfits := pre(accumulatedProfits) + portIn1.monetaryValue;
3793            currentBusinessValue := pre(currentBusinessValue) + portIn1.monetaryValue -
                   price[pre(outPointerFIFO)];
3794          end if;
3795        else
3796          nElementsFIFO := pre(nElementsFIFO) - 1;
3797          moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn1.monetaryValue +
                 portIn2.monetaryValue;
3798          accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 2;
3799          currentBusinessValue := pre(currentBusinessValue) - price[pre(outPointerFIFO)
                 ];
3800        end if;
3801        portOut.ID := FIFO[pre(outPointerFIFO)];
3802        portOut.monetaryValue := price[pre(outPointerFIFO)];
3803        outPointerFIFO := mod(pre(outPointerFIFO), lengthFIFO) + 1;
3804        portOut.writeFlag:= true;
3805        portIn1.writeOkFlag := true;
3806        portIn2.writeOkFlag := true;
3807        state := 7;
3808        controlTime := time;
3809      end when;
3810
3811      when (pre(state) == 1 and (not pre(portIn1.writeFlag)) and time > controlTime)
             then
```

```
3812          portIn1.writeOkFlag := false;
3813          state := 0;
3814          controlTime := time;
3815        end when;
3816
3817        when (pre(state) == 2 and (not pre(portIn2.writeFlag)) and time > controlTime)
                then
3818          portIn2.writeOkFlag := false;
3819          state := 0;
3820          controlTime := time;
3821            end when;
3822
3823         when (pre(state) == 3 and (not pre(portIn1.writeFlag)) and (not pre(portIn2.
               writeFlag)) and time > controlTime) then
3824          portIn1.writeOkFlag := false;
3825          portIn2.writeOkFlag := false;
3826          state := 0;
3827          controlTime := time;
3828         end when;
3829
3830         when (pre(state) == 4 and (not pre(portIn1.writeFlag)) and (not pre(portOut.
               writeOkFlag)) and time > controlTime) then
3831          portIn1.writeOkFlag := false;
3832          portOut.writeFlag := false;
3833          state := 0;
3834          controlTime := time;
3835         end when;
3836
3837         when (pre(state) == 5 and (not pre(portIn2.writeFlag)) and (not pre(portOut.
               writeOkFlag)) and time > controlTime) then
3838          portIn2.writeOkFlag := false;
3839          portOut.writeFlag := false;
3840          state := 0;
3841          controlTime := time;
3842         end when;
3843
3844         when (pre(state) == 6 and (not pre(portOut.writeOkFlag)) and time > controlTime)
               then
3845          portOut.writeFlag := false;
3846          state := 0;
3847          controlTime := time;
3848         end when;
3849
3850         when (pre(state) == 7 and (not pre(portIn1.writeFlag)) and (not pre(portIn2.
               writeFlag)) and (not pre(portOut.writeOkFlag)) and time > controlTime) then
3851          portIn1.writeOkFlag := false;
3852          portIn2.writeOkFlag := false;
3853          portOut.writeFlag := false;
3854          state := 0;
3855          controlTime := time;
3856         end when;
3857
3858        annotation (Icon(graphics={
3859              Ellipse(
3860                extent={{76,64},{-84,-56}},
3861                lineColor={0,0,0},
3862                lineThickness=0.5),
```

```
3863            Text(
3864              extent={{-40,-24},{36,-44}},
3865              lineColor={0,0,0},
3866              lineThickness=0.5,
3867              fillPattern=FillPattern.Solid,
3868              textString="circuit6"),
3869            Bitmap(
3870              extent={{-80,44},{66,-22}},
3871              imageSource=
3872                  "/*Put your image*/",
3873              fileName="modelica://RiskAnalysis/reoDymola.png")}), Diagram(graphics));
3874    end ReoCircuit6;
3875
3876    model ReoCircuit7
3877
3878      parameter Integer lengthFIFO = 100;
3879      Integer nElementsFIFO(start=0);
3880      Integer inPointerFIFO(start=1);
3881      Integer outPointerFIFO(start=1);
3882      Integer moneyLossTotalFIFO(start=0);
3883      Integer state(start=0);
3884      Real controlTime(start=0);
3885
3886      Real freeCapacity(start=0);
3887      Real usedCapacity(start=0);
3888      Integer accumulatedLosses(start=0);
3889      Integer accumulatedNumberOfLostEvents(start=0);
3890      Integer accumulatedNumberOfSuccessEvents(start=0);
3891      Integer accumulatedProfits(start=0);
3892      Integer currentBusinessValue(start=0);
3893
3894      ServicePortIn portIn annotation (Placement(transformation(extent={{
3895                -100,-10},{-80,10}}), iconTransformation(extent={{-96,-12},{-76,
3896               8}})));
3897      ServicePortOut portOut1 annotation (Placement(transformation(extent={
3898                {60,24},{80,44}}), iconTransformation(extent={{72,22},{92,42}})));
3899      ServicePortOut portOut2 annotation (Placement(transformation(extent={{74,-42},
3900               {94,-22}}),            iconTransformation(extent={{70,-52},{90,
3901               -32}})));
3902
3903      Modelica.Blocks.Interfaces.RealOutput currentFIFOCapacity annotation (
3904          Placement(transformation(extent={{-12,54},{8,74}}),
3905            iconTransformation(
3906            extent={{-10,-10},{10,10}},
3907            rotation=90,
3908            origin={2,66})));
3909
3910    protected
3911      Integer[lengthFIFO] FIFO;
3912      Integer[lengthFIFO] price;
3913
3914    public
3915      Modelica.Blocks.Interfaces.IntegerInput open annotation (Placement(
3916            transformation(extent={{-44,-44},{-4,-4}}), iconTransformation(
3917            extent={{-12,-12},{12,12}},
3918            rotation=90,
3919            origin={-16,-74})));
```

```
3920      Modelica.Blocks.Interfaces.IntegerInput case1or2 annotation (Placement(
3921            transformation(extent={{-6,-60},{34,-20}}), iconTransformation(
3922            extent={{-12,-12},{12,12}},
3923            rotation=90,
3924            origin={24,-74})));
3925    equation
3926      usedCapacity =  (nElementsFIFO * 100) / (lengthFIFO);
3927      freeCapacity =  100 - usedCapacity;
3928      accumulatedLosses = moneyLossTotalFIFO;
3929      currentFIFOCapacity = usedCapacity;
3930
3931    algorithm
3932      /* The circuit receives data from Client, while portService1, which should
             receive (case1or2), is not ready */
3933      when (pre(state) == 0 and pre(portIn.writeFlag) and (pre(open) <> 0) and (pre(
             case1or2) <> 0) and (not pre(portOut1.writeOkFlag) or (not pre(nElementsFIFO)
             > 0)) and time > controlTime) then
3934        if ((pre(nElementsFIFO) < lengthFIFO)) then
3935          FIFO[pre(inPointerFIFO)] := portIn.ID;
3936          price[pre(inPointerFIFO)] := portIn.monetaryValue;
3937          inPointerFIFO := mod(pre(inPointerFIFO), lengthFIFO) + 1;
3938          nElementsFIFO := pre(nElementsFIFO) + 1;
3939          accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                 1;
3940          accumulatedProfits := pre(accumulatedProfits) + portIn.monetaryValue;
3941          currentBusinessValue := pre(currentBusinessValue) + portIn.monetaryValue;
3942        else
3943          moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn.monetaryValue;
3944          accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
3945        end if;
3946        portIn.writeOkFlag := true;
3947        state := 1;
3948        controlTime := time;
3949      end when;
3950
3951      /* The circuit receives data from Client, while portService2, which should
             receive (not case1or2), is not ready */
3952      when (pre(state) == 0 and pre(portIn.writeFlag) and (pre(open) <> 0) and (pre(
             case1or2) == 0) and (not pre(portOut2.writeOkFlag) or (not pre(nElementsFIFO)
             > 0)) and time > controlTime) then
3953        if ((pre(nElementsFIFO) < lengthFIFO)) then
3954          FIFO[pre(inPointerFIFO)] := portIn.ID;
3955          price[pre(inPointerFIFO)] := portIn.monetaryValue;
3956          inPointerFIFO := mod(pre(inPointerFIFO), lengthFIFO) + 1;
3957          nElementsFIFO := pre(nElementsFIFO) + 1;
3958          accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                 1;
3959          accumulatedProfits := pre(accumulatedProfits) + portIn.monetaryValue;
3960          currentBusinessValue := pre(currentBusinessValue) + portIn.monetaryValue;
3961        else
3962          moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn.monetaryValue;
3963          accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
3964        end if;
3965        portIn.writeOkFlag := true;
3966        state := 1;
3967        controlTime := time;
3968      end when;
```

```
3969
3970        /* The circuit receives data from Client, but the "tap" is closed (not open) */
3971        when (pre(state) == 0 and pre(portIn.writeFlag) and (pre(open) == 0) and time >
                controlTime) then
3972          if ((pre(nElementsFIFO) < lengthFIFO)) then
3973            FIFO[pre(inPointerFIFO)] := portIn.ID;
3974            price[pre(inPointerFIFO)] := portIn.monetaryValue;
3975            inPointerFIFO := mod(pre(inPointerFIFO), lengthFIFO) + 1;
3976            nElementsFIFO := pre(nElementsFIFO) + 1;
3977            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                    1;
3978            accumulatedProfits := pre(accumulatedProfits) + portIn.monetaryValue;
3979            currentBusinessValue := pre(currentBusinessValue) + portIn.monetaryValue;
3980          else
3981            moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn.monetaryValue;
3982            accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
3983          end if;
3984          portIn.writeOkFlag := true;
3985          state := 1;
3986          controlTime := time;
3987        end when;
3988
3989        /* The circuit receives data from Client and sends to portService1 (case1or2 is
                true) */
3990        when (pre(state) == 0 and pre(portIn.writeFlag) and (pre(open) <> 0) and (pre(
                case1or2) <> 0) and pre(portOut1.writeOkFlag) and pre(nElementsFIFO) > 0 and
                time > controlTime) then
3991          if ((pre(nElementsFIFO) < lengthFIFO)) then
3992            FIFO[pre(inPointerFIFO)] := portIn.ID;
3993            price[pre(inPointerFIFO)] := portIn.monetaryValue;
3994            inPointerFIFO := mod(pre(inPointerFIFO), lengthFIFO) + 1;
3995            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                    1;
3996            accumulatedProfits := pre(accumulatedProfits) + portIn.monetaryValue;
3997            currentBusinessValue := pre(currentBusinessValue) + portIn.monetaryValue -
                    price[pre(outPointerFIFO)];
3998          else
3999            moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn.monetaryValue;
4000            accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
4001            currentBusinessValue := pre(currentBusinessValue) - price[pre(outPointerFIFO
                    )];
4002            nElementsFIFO := pre(nElementsFIFO) - 1;
4003          end if;
4004          /* Reading from FIFO */
4005          portOut1.ID := FIFO[pre(outPointerFIFO)];
4006          portOut1.monetaryValue := price[pre(outPointerFIFO)];
4007          outPointerFIFO := mod(pre(outPointerFIFO), lengthFIFO) + 1;
4008          /* Setting the flags on the ports */
4009          portIn.writeOkFlag := true;
4010          portOut1.writeFlag := true;
4011          state := 2;
4012          controlTime := time;
4013        end when;
4014
4015        /* The circuit receives data from Client and sends to portService2 (case1or2 is
                false) */
```

```
4016        when (pre(state) == 0 and pre(portIn.writeFlag) and (pre(open) <> 0) and (pre(
                case1or2) == 0) and pre(portOut2.writeOkFlag) and pre(nElementsFIFO) > 0 and
                time > controlTime) then
4017          if ((pre(nElementsFIFO) < lengthFIFO)) then
4018            FIFO[pre(inPointerFIFO)] := portIn.ID;
4019            price[pre(inPointerFIFO)] := portIn.monetaryValue;
4020            inPointerFIFO := mod(pre(inPointerFIFO), lengthFIFO) + 1;
4021            accumulatedNumberOfSuccessEvents := pre(accumulatedNumberOfSuccessEvents) +
                    1;
4022            accumulatedProfits := pre(accumulatedProfits) + portIn.monetaryValue;
4023            currentBusinessValue := pre(currentBusinessValue) + portIn.monetaryValue -
                    price[pre(outPointerFIFO)];
4024          else
4025            moneyLossTotalFIFO := pre(moneyLossTotalFIFO) + portIn.monetaryValue;
4026            accumulatedNumberOfLostEvents := pre(accumulatedNumberOfLostEvents) + 1;
4027            currentBusinessValue := pre(currentBusinessValue) - price[pre(outPointerFIFO
                    )];
4028            nElementsFIFO := pre(nElementsFIFO) - 1;
4029          end if;
4030          /* Reading from FIFO */
4031          portOut2.ID := FIFO[pre(outPointerFIFO)];
4032          portOut2.monetaryValue := price[pre(outPointerFIFO)];
4033          outPointerFIFO := mod(pre(outPointerFIFO), lengthFIFO) + 1;
4034          /* Setting the flags on the ports */
4035          portIn.writeOkFlag := true;
4036          portOut2.writeFlag := true;
4037          state := 3;
4038          controlTime := time;
4039        end when;
4040
4041        /* The circuit sends to portService1 (case1or2 is true) */
4042        when (pre(state) == 0 and (not pre(portIn.writeFlag)) and (pre(open) <> 0) and (
                pre(case1or2) <> 0) and pre(portOut1.writeOkFlag) and pre(nElementsFIFO) > 0
                and time > controlTime) then
4043          /* Reading from FIFO */
4044          portOut1.ID := FIFO[pre(outPointerFIFO)];
4045          portOut1.monetaryValue := price[pre(outPointerFIFO)];
4046          outPointerFIFO := mod(pre(outPointerFIFO), lengthFIFO) + 1;
4047          currentBusinessValue := pre(currentBusinessValue) - price[pre(outPointerFIFO)
                  ];
4048          nElementsFIFO := pre(nElementsFIFO) - 1;
4049          /* Setting the flags on the ports */
4050          //portIn.writeOkFlag := true;
4051          portOut1.writeFlag := true;
4052          state := 4;
4053          controlTime := time;
4054        end when;
4055
4056        /* The circuit sends to portService2 (case1or2 is false) */
4057        when (pre(state) == 0 and (not pre(portIn.writeFlag)) and (pre(open) <> 0) and (
                pre(case1or2) == 0) and pre(portOut2.writeOkFlag) and pre(nElementsFIFO) > 0
                and time > controlTime) then
4058          /* Reading from FIFO */
4059          portOut2.ID := FIFO[pre(outPointerFIFO)];
4060          portOut2.monetaryValue := price[pre(outPointerFIFO)];
4061          outPointerFIFO := mod(pre(outPointerFIFO), lengthFIFO) + 1;
```

```
4062          currentBusinessValue := pre(currentBusinessValue) - price[pre(outPointerFIFO)
                  ];
4063          nElementsFIFO := pre(nElementsFIFO) - 1;
4064          /* Setting the flags on the ports */
4065          //portIn.writeOkFlag := true;
4066          portOut2.writeFlag := true;
4067          state := 5;
4068          controlTime := time;
4069        end when;
4070
4071        when (pre(state) == 1 and (not pre(portIn.writeFlag)) and time > controlTime)
              then
4072          portIn.writeOkFlag := false;
4073          state := 0;
4074          controlTime := time;
4075        end when;
4076
4077        when (pre(state) == 2 and (not pre(portIn.writeFlag)) and (not pre(portOut1.
              writeOkFlag)) and time > controlTime) then
4078          portIn.writeOkFlag := false;
4079          portOut1.writeFlag := false;
4080          state := 0;
4081          controlTime := time;
4082        end when;
4083
4084        when (pre(state) == 3 and (not pre(portIn.writeFlag)) and (not pre(portOut2.
              writeOkFlag)) and time > controlTime) then
4085          portIn.writeOkFlag := false;
4086          portOut2.writeFlag := false;
4087          state := 0;
4088          controlTime := time;
4089        end when;
4090
4091        when (pre(state) == 4 and (not pre(portOut1.writeOkFlag)) and time > controlTime
              ) then
4092          portOut1.writeFlag := false;
4093          state := 0;
4094          controlTime := time;
4095        end when;
4096
4097        when (pre(state) == 5 and (not pre(portOut2.writeOkFlag)) and time > controlTime
              ) then
4098          portOut2.writeFlag := false;
4099          state := 0;
4100          controlTime := time;
4101        end when;
4102
4103      annotation (Icon(graphics={
4104            Ellipse(
4105              extent={{84,56},{-76,-64}},
4106              lineColor={0,0,0},
4107              lineThickness=0.5),
4108            Text(
4109              extent={{-34,-32},{42,-52}},
4110              lineColor={0,0,0},
4111              lineThickness=0.5,
4112              fillPattern=FillPattern.Solid,
```

```
4113              textString="circuit7"),
4114            Bitmap(
4115              extent={{-70,36},{76,-30}},
4116              imageSource=
4117                  "/*Put your image*/",
4118              fileName="modelica://RiskAnalysis/reoDymola.png")}),
4119                                          Diagram(graphics));
4120    end ReoCircuit7;
4121
4122    model C1
4123
4124      BasketService basketService(
4125        mX=2,
4126        mY=3,
4127        mZ=1)
4128        annotation (Placement(transformation(extent={{-46,38},{-20,64}})));
4129      PaymentService paymentService(
4130        mX=3,
4131        mY=2,
4132        mZ=1)
4133        annotation (Placement(transformation(extent={{-6,38},{22,64}})));
4134      CollectOrderService collectOrderService
4135        annotation (Placement(transformation(extent={{32,38},{56,64}})));
4136      SendMailService sendMailService(
4137        mX=4,
4138        mY=2,
4139        mZ=1)
4140        annotation (Placement(transformation(extent={{74,-52},{100,-26}})));
4141      ReoCircuit1 reoCircuit1_1
4142        annotation (Placement(transformation(extent={{-86,-18},{-50,12}})));
4143      CollectOrderService collectOrderService1
4144        annotation (Placement(transformation(extent={{30,-64},{56,-38}})));
4145      ReoCircuit2 reoCircuit2_1
4146        annotation (Placement(transformation(extent={{-32,4},{4,32}})));
4147      ReoCircuit2 reoCircuit2_2
4148        annotation (Placement(transformation(extent={{-32,-36},{4,-8}})));
4149      ReoCircuit2 reoCircuit2_3
4150        annotation (Placement(transformation(extent={{10,-36},{46,-8}})));
4151      ReoCircuit2 reoCircuit2_4
4152        annotation (Placement(transformation(extent={{10,4},{46,32}})));
4153      ReoCircuit3 reoCircuit3_1
4154        annotation (Placement(transformation(extent={{54,-20},{90,10}})));
4155      Modelica.Blocks.Sources.BooleanTable flushTable(table={36,71,107,142,178,
4156            214,249,285,320,356})
4157        annotation (Placement(transformation(extent={{62,80},{82,100}})));
4158      Modelica.Blocks.Sources.BooleanTable flushTable1(table={36,71,107,142,178,
4159            214,249,285,320,356})
4160        annotation (Placement(transformation(extent={{30,-100},{50,-80}})));
4161      ServicePortIn servicePortIn annotation (Placement(transformation(extent={{-110,
4162                0},{-90,20}}), iconTransformation(extent={{-124,-28},{-84,10}})));
4163      Modelica.Blocks.Interfaces.IntegerOutput outSendMailService annotation (
4164          Placement(transformation(extent={{74,-70},{94,-50}}),
4165            iconTransformation(
4166            extent={{-15,-15},{15,15}},
4167            rotation=90,
4168            origin={65,77})));
4169      Modelica.Blocks.Math.Max max2
```

```
4170        annotation (Placement(transformation(extent={{-26,-8},{-14,4}})));
4171      Modelica.Blocks.Math.Max max3
4172        annotation (Placement(transformation(extent={{2,-8},{14,4}})));
4173      Modelica.Blocks.Math.Max max1
4174        annotation (Placement(transformation(extent={{-48,-26},{-36,-14}})));
4175      Modelica.Blocks.Math.Max max4
4176        annotation (Placement(transformation(extent={{30,-8},{42,4}})));
4177      Modelica.Blocks.Math.Max max5
4178        annotation (Placement(transformation(extent={{68,16},{80,28}})));
4179      Modelica.Blocks.Interfaces.RealOutput outMaxFIFOCapacity annotation (
4180          Placement(transformation(extent={{76,40},{96,60}}), iconTransformation(
4181            extent={{-15,-15},{15,15}},
4182            rotation=90,
4183            origin={23,77})));
4184      ForwardService forwardService
4185        annotation (Placement(transformation(extent={{-90,20},{-66,44}})));
4186      Modelica.Blocks.MathInteger.Sum sum(nu=6) annotation (Placement(
4187            transformation(
4188            extent={{-6,-6},{6,6}},
4189            rotation=90,
4190            origin={-64,78})));
4191      Modelica.Blocks.Interfaces.IntegerOutput outAccumulatedLosses annotation (
4192          Placement(transformation(
4193            extent={{-10,-10},{10,10}},
4194            rotation=90,
4195            origin={-90,96}), iconTransformation(
4196            extent={{-15,-15},{15,15}},
4197            rotation=90,
4198            origin={-65,77})));
4199      Modelica.Blocks.MathInteger.Sum sum1(
4200                                          nu=4) annotation (Placement(
4201            transformation(
4202            extent={{-6,-6},{6,6}},
4203            rotation=90,
4204            origin={-40,82})));
4205      Modelica.Blocks.Interfaces.IntegerOutput outAccumulatedLossesServices
4206                                                                  annotation (
4207          Placement(transformation(
4208            extent={{-10,-10},{10,10}},
4209            rotation=90,
4210            origin={-46,102}),iconTransformation(
4211            extent={{-15,-15},{15,15}},
4212            rotation=90,
4213            origin={-23,77})));
4214      PaymentService1 paymentService1_1(
4215        mX=2,
4216        mY=3,
4217        mZ=4) annotation (Placement(transformation(extent={{-6,-64},{22,-38}})));
4218      BasketService1 basketService1_1(
4219        mX=1,
4220        mY=2,
4221        mZ=3) annotation (Placement(transformation(extent={{-46,-64},{-20,-38}})));
4222    equation
4223      connect(flushTable.y, collectOrderService.emptyEvent)    annotation (Line(
4224          points={{83,90},{90,90},{90,64},{43.88,64},{43.88,60.49}},
4225          color={255,0,255},
4226          smooth=Smooth.None));
```

```
4227      connect(flushTable1.y, collectOrderService1.emptyEvent)    annotation (Line(
4228          points={{51,-90},{60,-90},{60,-41.51},{42.87,-41.51}},
4229          color={255,0,255},
4230          smooth=Smooth.None));
4231      connect(reoCircuit3_1.portOut,sendMailService. mailServiceIn) annotation (
4232          Line(
4233          points={{86.76,-3.5},{86.76,-21.75},{75.3,-21.75},{75.3,-39.26}},
4234          color={255,128,0},
4235          smooth=Smooth.None));
4236      connect(paymentService.basketIn,reoCircuit2_1. portOut) annotation (Line(
4237          points={{-7.4,50.74},{-7.4,34.37},{2.2,34.37},{2.2,18.84}},
4238          color={255,128,0},
4239          smooth=Smooth.None));
4240      connect(paymentService.basketOut,reoCircuit2_4. portIn) annotation (Line(
4241          points={{20.6,50.74},{20.6,35.37},{11.8,35.37},{11.8,18.56}},
4242          color={255,128,0},
4243          smooth=Smooth.None));
4244      connect(collectOrderService.accountStatementOut,reoCircuit3_1. portIn2)
4245        annotation (Line(
4246          points={{55.04,50.74},{55.04,25.37},{59.4,25.37},{59.4,1.3}},
4247          color={255,128,0},
4248          smooth=Smooth.None));
4249      connect(collectOrderService1.accountStatementOut,reoCircuit3_1. portIn1)
4250        annotation (Line(
4251          points={{54.96,-51.26},{54.96,-29.63},{59.4,-29.63},{59.4,-8.9}},
4252          color={255,128,0},
4253          smooth=Smooth.None));
4254      connect(collectOrderService.basketIn,reoCircuit2_4. portOut) annotation (
4255          Line(
4256          points={{33.44,51},{33.44,34.37},{44.2,34.37},{44.2,18.84}},
4257          color={255,128,0},
4258          smooth=Smooth.None));
4259      connect(collectOrderService1.basketIn,reoCircuit2_3. portOut) annotation (
4260          Line(
4261          points={{31.56,-51},{31.56,-36.63},{44.2,-36.63},{44.2,-21.16}},
4262          color={255,128,0},
4263          smooth=Smooth.None));
4264      connect(sendMailService.outputCT, outSendMailService) annotation (Line(
4265          points={{88.43,-48.49},{88.43,-54.245},{84,-54.245},{84,-60}},
4266          color={255,127,0},
4267          smooth=Smooth.None));
4268      connect(max2.y, max3.u2) annotation (Line(
4269          points={{-13.4,-2},{-8,-2},{-8,-5.6},{0.8,-5.6}},
4270          color={0,0,127},
4271          smooth=Smooth.None));
4272      connect(reoCircuit2_4.currentFIFOCapacity, max3.u1) annotation (Line(
4273          points={{24.4,9.32},{-3.8,9.32},{-3.8,1.6},{0.8,1.6}},
4274          color={0,0,127},
4275          smooth=Smooth.None));
4276      connect(reoCircuit1_1.currentFIFOCapacity, max1.u1) annotation (Line(
4277          points={{-71.6,-13.2},{-71.6,-18.6},{-49.2,-18.6},{-49.2,-16.4}},
4278          color={0,0,127},
4279          smooth=Smooth.None));
4280      connect(reoCircuit2_2.currentFIFOCapacity, max1.u2) annotation (Line(
4281          points={{-17.6,-30.68},{-58,-30.68},{-58,-24},{-52,-24},{-49.2,-23.6}},
4282          color={0,0,127},
4283          smooth=Smooth.None));
```

```
4284
4285        connect(max1.y, max2.u2) annotation (Line(
4286            points={{-35.4,-20},{-34,-20},{-34,-5.6},{-27.2,-5.6}},
4287            color={0,0,127},
4288            smooth=Smooth.None));
4289        connect(reoCircuit2_1.currentFIFOCapacity, max2.u1) annotation (Line(
4290            points={{-17.6,9.32},{-32.8,9.32},{-32.8,1.6},{-27.2,1.6}},
4291            color={0,0,127},
4292            smooth=Smooth.None));
4293        connect(max3.y, max4.u1) annotation (Line(
4294            points={{14.6,-2},{22,-2},{22,1.6},{28.8,1.6}},
4295            color={0,0,127},
4296            smooth=Smooth.None));
4297        connect(reoCircuit2_3.currentFIFOCapacity, max4.u2) annotation (Line(
4298            points={{24.4,-30.68},{24,-30},{8,-30},{8,-12},{26,-6},{28.8,-5.6}},
4299            color={0,0,127},
4300            smooth=Smooth.None));
4301        connect(max4.y, max5.u1) annotation (Line(
4302            points={{42.6,-2},{54,-2},{54,25.6},{66.8,25.6}},
4303            color={0,0,127},
4304            smooth=Smooth.None));
4305        connect(reoCircuit3_1.currentFIFOCapacity, max5.u2) annotation (Line(
4306            points={{69.12,-13.7},{48,-14},{48,8},{62,8},{62,18},{66.8,18},{66.8,
4307                18.4}},
4308            color={0,0,127},
4309            smooth=Smooth.None));
4310        connect(max5.y, outMaxFIFOCapacity) annotation (Line(
4311            points={{80.6,22},{80,22},{80,50},{86,50}},
4312            color={0,0,127},
4313            smooth=Smooth.None));
4314        connect(reoCircuit1_1.portOut1, basketService.singleOrderIn) annotation (
4315            Line(
4316            points={{-53.96,2.1},{-53.96,26.05},{-44.7,26.05},{-44.7,51}},
4317            color={255,128,0},
4318            smooth=Smooth.None));
4319        connect(basketService.basketServiceOut, reoCircuit2_1.portIn) annotation (
4320            Line(
4321            points={{-21.04,51},{-21.04,35.5},{-30.2,35.5},{-30.2,18.56}},
4322            color={255,128,0},
4323            smooth=Smooth.None));
4324        connect(servicePortIn, forwardService.forwardServiceIn) annotation (Line(
4325            points={{-100,10},{-94,10},{-94,32.96},{-90.48,32.96}},
4326            color={255,128,0},
4327            smooth=Smooth.None));
4328        connect(forwardService.forwardServiceOut, reoCircuit1_1.portIn) annotation (
4329           Line(
4330            points={{-66.48,32.96},{-66.48,20.48},{-84.2,20.48},{-84.2,-3.3}},
4331            color={255,128,0},
4332            smooth=Smooth.None));
4333        connect(reoCircuit3_1.outAccumulatedLosses, sum.u[1]) annotation (Line(
4334            points={{75.96,-13.7},{-4.02,-13.7},{-4.02,72},{-67.5,72}},
4335            color={255,127,0},
4336            smooth=Smooth.None));
4337        connect(sum.y, outAccumulatedLosses) annotation (Line(
4338            points={{-64,84.9},{-64,96},{-90,96}},
4339            color={255,127,0},
4340            smooth=Smooth.None));
```

```
4341      connect(outMaxFIFOCapacity, outMaxFIFOCapacity) annotation (Line(
4342          points={{86,50},{86,50},{86,50}},
4343          color={0,0,127},
4344          smooth=Smooth.None));
4345      connect(outAccumulatedLosses, outAccumulatedLosses) annotation (Line(
4346          points={{-90,96},{-90,96}},
4347          color={255,127,0},
4348          smooth=Smooth.None));
4349      connect(reoCircuit2_4.outAccumulatedLosses, sum.u[2]) annotation (Line(
4350          points={{31.96,9.32},{-13.02,9.32},{-13.02,72},{-66.1,72}},
4351          color={255,127,0},
4352          smooth=Smooth.None));
4353      connect(reoCircuit2_3.outAccumulatedLosses, sum.u[3]) annotation (Line(
4354          points={{31.96,-30.68},{31.96,20.66},{-64.7,20.66},{-64.7,72}},
4355          color={255,127,0},
4356          smooth=Smooth.None));
4357      connect(reoCircuit2_1.outAccumulatedLosses, sum.u[4]) annotation (Line(
4358          points={{-10.04,9.32},{-10.04,40.66},{-63.3,40.66},{-63.3,72}},
4359          color={255,127,0},
4360          smooth=Smooth.None));
4361      connect(reoCircuit2_2.outAccumulatedLosses, sum.u[5]) annotation (Line(
4362          points={{-10.04,-30.68},{-10.04,20.66},{-61.9,20.66},{-61.9,72}},
4363          color={255,127,0},
4364          smooth=Smooth.None));
4365      connect(reoCircuit1_1.outAccumulatedLosses, sum.u[6]) annotation (Line(
4366          points={{-64.4,-13.2},{-64.4,29.4},{-60.5,29.4},{-60.5,72}},
4367          color={255,127,0},
4368          smooth=Smooth.None));
4369      connect(basketService.outAccumulatedLosses, sum1.u[1]) annotation (Line(
4370          points={{-33.13,60.49},{-33.13,70.245},{-43.15,70.245},{-43.15,76}},
4371          color={255,127,0},
4372          smooth=Smooth.None));
4373      connect(sum1.y, outAccumulatedLossesServices) annotation (Line(
4374          points={{-40,88.9},{-42,88.9},{-42,102},{-46,102}},
4375          color={255,127,0},
4376          smooth=Smooth.None));
4377      connect(outSendMailService, outSendMailService) annotation (Line(
4378          points={{84,-60},{84,-60}},
4379          color={255,127,0},
4380          smooth=Smooth.None));
4381      connect(paymentService.outAccumulatedLosses, sum1.u[2]) annotation (Line(
4382          points={{6.74,60.49},{-12.11,60.49},{-12.11,76},{-41.05,76}},
4383          color={255,127,0},
4384          smooth=Smooth.None));
4385      connect(paymentService1_1.basketOut, reoCircuit2_3.portIn) annotation (Line(
4386          points={{20.6,-51.26},{20.6,-36.63},{11.8,-36.63},{11.8,-21.44}},
4387          color={255,128,0},
4388          smooth=Smooth.None));
4389      connect(paymentService1_1.basketIn, reoCircuit2_2.portOut) annotation (Line(
4390          points={{-7.4,-51.26},{-7.4,-36.63},{2.2,-36.63},{2.2,-21.16}},
4391          color={255,128,0},
4392          smooth=Smooth.None));
4393      connect(paymentService1_1.outAccumulatedLosses, sum1.u[3]) annotation (Line(
4394          points={{6.74,-41.51},{6.74,17.245},{-38.95,17.245},{-38.95,76}},
4395          color={255,127,0},
4396          smooth=Smooth.None));
4397      connect(basketService1_1.outAccumulatedLosses, sum1.u[4]) annotation (Line(
```

```
4398          points={{-33.13,-41.51},{-33.13,17.245},{-36.85,17.245},{-36.85,76}},
4399          color={255,127,0},
4400          smooth=Smooth.None));
4401      connect(reoCircuit1_1.portOut2, basketService1_1.singleOrderIn) annotation (
4402        Line(
4403          points={{-53.96,-8.7},{-53.96,-30.35},{-44.7,-30.35},{-44.7,-51}},
4404          color={255,128,0},
4405          smooth=Smooth.None));
4406      connect(basketService1_1.basketServiceOut, reoCircuit2_2.portIn)
4407        annotation (Line(
4408          points={{-21.04,-51},{-21.04,-36.5},{-30.2,-36.5},{-30.2,-21.44}},
4409          color={255,128,0},
4410          smooth=Smooth.None));
4411      annotation (Icon(graphics={Rectangle(
4412              extent={{-100,62},{100,-78}},
4413              lineColor={0,0,255},
4414              lineThickness=0.5), Text(
4415              extent={{-60,8},{60,-28}},
4416              lineColor={0,0,0},
4417              textString="Case1")}),
4418                                    Diagram(graphics));
4419    end C1;
4420
4421    model C2
4422
4423      BasketService basketService(
4424        mX=2,
4425        mY=3,
4426        mZ=1)
4427        annotation (Placement(transformation(extent={{-46,38},{-20,64}})));
4428      PaymentService paymentService(
4429        mX=3,
4430        mY=2,
4431        mZ=1)
4432        annotation (Placement(transformation(extent={{-8,38},{20,64}})));
4433      CollectOrderService collectOrderService
4434        annotation (Placement(transformation(extent={{30,38},{54,64}})));
4435      SendMailService sendMailService(
4436        mX=4,
4437        mY=2,
4438        mZ=1)
4439        annotation (Placement(transformation(extent={{72,-52},{98,-26}})));
4440      PaymentService1 paymentService1_1(
4441        mX=2,
4442        mY=3,
4443        mZ=4) annotation (Placement(transformation(extent={{-8,-64},{20,-38}})));
4444      CollectOrderService collectOrderService1
4445        annotation (Placement(transformation(extent={{28,-64},{54,-38}})));
4446      Modelica.Blocks.Sources.BooleanTable flushTable(table={36,71,107,142,178,
4447            214,249,285,320,356})
4448        annotation (Placement(transformation(extent={{60,80},{80,100}})));
4449      Modelica.Blocks.Sources.BooleanTable flushTable1(table={36,71,107,142,178,
4450            214,249,285,320,356})
4451        annotation (Placement(transformation(extent={{30,-100},{50,-80}})));
4452      ReoCircuit4  recoCircuit4_1
4453        annotation (Placement(transformation(extent={{-72,-14},{-36,14}})));
4454      ReoCircuit5  recoCircuit5_1
```

```
4455        annotation (Placement(transformation(extent={{-30,-14},{6,14}})));
4456      ReoCircuit5  reoCircuit5_2
4457        annotation (Placement(transformation(extent={{8,-14},{46,14}})));
4458      ReoCircuit6 reoCircuit6_1
4459        annotation (Placement(transformation(extent={{52,-16},{88,14}})));
4460      ServicePortIn servicePortIn annotation (Placement(transformation(extent={{-110,
4461              0},{-90,20}}), iconTransformation(extent={{-122,-26},{-82,12}})));
4462      Modelica.Blocks.Interfaces.IntegerOutput outSendMailService annotation (
4463          Placement(transformation(extent={{76,-74},{96,-54}}),
4464            iconTransformation(
4465            extent={{-15,-15},{15,15}},
4466            rotation=90,
4467            origin={65,79})));
4468      Modelica.Blocks.Math.Max max1
4469        annotation (Placement(transformation(extent={{-18,-28},{-6,-16}})));
4470      Modelica.Blocks.Math.Max max2
4471        annotation (Placement(transformation(extent={{16,-28},{28,-16}})));
4472      Modelica.Blocks.Math.Max max3
4473        annotation (Placement(transformation(extent={{70,20},{82,32}})));
4474      Modelica.Blocks.Interfaces.RealOutput outMaxFIFOCapacity annotation (
4475          Placement(transformation(extent={{88,48},{108,68}}), iconTransformation(
4476            extent={{-15,-15},{15,15}},
4477            rotation=90,
4478            origin={23,79})));
4479      ForwardService forwardService
4480        annotation (Placement(transformation(extent={{-86,34},{-62,58}})));
4481      Modelica.Blocks.MathInteger.Sum sum(nu=4) annotation (Placement(
4482            transformation(
4483            extent={{-6,-6},{6,6}},
4484            rotation=90,
4485            origin={-82,-20})));
4486      Modelica.Blocks.Interfaces.IntegerOutput outAccumulatedLosses annotation (
4487          Placement(transformation(
4488            extent={{-10,-10},{10,10}},
4489            rotation=90,
4490            origin={-82,10}), iconTransformation(
4491            extent={{-15,-15},{15,15}},
4492            rotation=90,
4493            origin={-65,79})));
4494      Modelica.Blocks.MathInteger.Sum sum1(
4495                                          nu=4) annotation (Placement(
4496            transformation(
4497            extent={{-6,-6},{6,6}},
4498            rotation=90,
4499            origin={-52,72})));
4500      Modelica.Blocks.Interfaces.IntegerOutput outAccumulatedLossesServices
4501                                                                  annotation (
4502          Placement(transformation(
4503            extent={{-10,-10},{10,10}},
4504            rotation=90,
4505            origin={-26,84}), iconTransformation(
4506            extent={{-15,-15},{15,15}},
4507            rotation=90,
4508            origin={-21,79})));
4509
4510      BasketService1 basketService1_1(
4511        mX=1,
```

```
4512        mY=2,
4513        mZ=3) annotation (Placement(transformation(extent={{-46,-64},{-20,-38}})));
4514    equation
4515      connect(flushTable.y, collectOrderService.emptyEvent)   annotation (Line(
4516          points={{81,90},{88,90},{88,64},{41.88,64},{41.88,60.49}},
4517          color={255,0,255},
4518          smooth=Smooth.None));
4519      connect(flushTable1.y, collectOrderService1.emptyEvent)   annotation (Line(
4520          points={{51,-90},{58,-90},{58,-41.51},{40.87,-41.51}},
4521          color={255,0,255},
4522          smooth=Smooth.None));
4523      connect(recoCircuit5_1.portOut1,paymentService. basketIn) annotation (Line(
4524          points={{2.4,4.48},{2.4,27.52},{-9.4,27.52},{-9.4,50.74}},
4525          color={255,128,0},
4526          smooth=Smooth.None));
4527      connect(recoCircuit5_2.portOut1,collectOrderService. basketIn) annotation (
4528          Line(
4529          points={{42.2,4.48},{42.2,27.52},{31.44,27.52},{31.44,51}},
4530          color={255,128,0},
4531          smooth=Smooth.None));
4532      connect(recoCircuit5_2.portOut2,collectOrderService1. basketIn) annotation (
4533         Line(
4534          points={{41.82,-5.04},{41.82,-28.66},{29.56,-28.66},{29.56,-51}},
4535          color={255,128,0},
4536          smooth=Smooth.None));
4537      connect(collectOrderService.accountStatementOut,reoCircuit6_1. portIn2)
4538        annotation (Line(
4539          points={{53.04,50.74},{53.04,27.37},{55.24,27.37},{55.24,-5.8}},
4540          color={255,128,0},
4541          smooth=Smooth.None));
4542      connect(collectOrderService1.accountStatementOut,reoCircuit6_1. portIn1)
4543        annotation (Line(
4544          points={{52.96,-51.26},{52.96,-28.63},{55.24,-28.63},{55.24,4.7}},
4545          color={255,128,0},
4546          smooth=Smooth.None));
4547      connect(reoCircuit6_1.portOut,sendMailService. mailServiceIn) annotation (
4548          Line(
4549          points={{85.48,-0.4},{85.48,-20.2},{73.3,-20.2},{73.3,-39.26}},
4550          color={255,128,0},
4551          smooth=Smooth.None));
4552      connect(sendMailService.outputCT, outSendMailService) annotation (Line(
4553          points={{86.43,-48.49},{86.43,-55.245},{86,-55.245},{86,-64}},
4554          color={255,127,0},
4555          smooth=Smooth.None));
4556      connect(basketService.basketServiceOut, recoCircuit5_1.portIn1) annotation (
4557         Line(
4558          points={{-21.04,51},{-21.04,27.5},{-26.4,27.5},{-26.4,4.48}},
4559          color={255,128,0},
4560          smooth=Smooth.None));
4561      connect(basketService.singleOrderIn, recoCircuit4_1.portOut1) annotation (
4562          Line(
4563          points={{-44.7,51},{-44.7,27.5},{-39.24,27.5},{-39.24,4.48}},
4564          color={255,128,0},
4565          smooth=Smooth.None));
4566      connect(recoCircuit5_1.currentFIFOCapacity, max1.u1) annotation (Line(
4567          points={{-14.52,-9.52},{-14.52,-13.76},{-19.2,-13.76},{-19.2,-18.4}},
4568          color={0,0,127},
```

```
4569            smooth=Smooth.None));
4570        connect(recoCircuit4_1.currentFIFOCapacity, max1.u2) annotation (Line(
4571            points={{-56.52,-9.8},{-38.26,-9.8},{-38.26,-25.6},{-19.2,-25.6}},
4572            color={0,0,127},
4573            smooth=Smooth.None));
4574        connect(recoCircuit5_2.currentFIFOCapacity, max2.u1) annotation (Line(
4575            points={{24.34,-9.52},{9.17,-9.52},{9.17,-18.4},{14.8,-18.4}},
4576            color={0,0,127},
4577            smooth=Smooth.None));
4578        connect(max1.y, max2.u2) annotation (Line(
4579            points={{-5.4,-22},{4,-22},{4,-25.6},{14.8,-25.6}},
4580            color={0,0,127},
4581            smooth=Smooth.None));
4582        connect(max2.y, max3.u1) annotation (Line(
4583            points={{28.6,-22},{48,-22},{48,29.6},{68.8,29.6}},
4584            color={0,0,127},
4585            smooth=Smooth.None));
4586        connect(reoCircuit6_1.currentFIFOCapacity, max3.u2) annotation (Line(
4587            points={{66.04,-10.6},{66,-10},{66,-10},{50,-10},{50,-10},{50,22},{50,
4588                22},{68.8,22.4}},
4589            color={0,0,127},
4590            smooth=Smooth.None));
4591        connect(max3.y, outMaxFIFOCapacity) annotation (Line(
4592            points={{82.6,26},{86,26},{86,58},{98,58}},
4593            color={0,0,127},
4594            smooth=Smooth.None));
4595        connect(servicePortIn, forwardService.forwardServiceIn) annotation (Line(
4596            points={{-100,10},{-100,40},{-86.48,40},{-86.48,46.96}},
4597            color={255,128,0},
4598            smooth=Smooth.None));
4599        connect(forwardService.forwardServiceOut, recoCircuit4_1.portIn)
4600          annotation (Line(
4601            points={{-62.48,46.96},{-62.48,23.48},{-69.48,23.48},{-69.48,-0.28}},
4602            color={255,128,0},
4603            smooth=Smooth.None));
4604        connect(recoCircuit4_1.outAccumulatedLosses, sum.u[1]) annotation (Line(
4605            points={{-50.04,-9.8},{-65.02,-9.8},{-65.02,-26},{-85.15,-26}},
4606            color={255,127,0},
4607            smooth=Smooth.None));
4608        connect(recoCircuit5_1.outAccumulatedLosses, sum.u[2]) annotation (Line(
4609            points={{-8.4,-9.52},{-44.2,-9.52},{-44.2,-26},{-83.05,-26}},
4610            color={255,127,0},
4611            smooth=Smooth.None));
4612        connect(recoCircuit5_2.outAccumulatedLosses, sum.u[3]) annotation (Line(
4613            points={{30.8,-9.52},{-25.6,-9.52},{-25.6,-26},{-80.95,-26}},
4614            color={255,127,0},
4615            smooth=Smooth.None));
4616        connect(reoCircuit6_1.outAccumulatedLosses, sum.u[4]) annotation (Line(
4617            points={{73.24,-10.6},{-4.38,-10.6},{-4.38,-26},{-78.85,-26}},
4618            color={255,127,0},
4619            smooth=Smooth.None));
4620        connect(sum.y, outAccumulatedLosses) annotation (Line(
4621            points={{-82,-13.1},{-82,10}},
4622            color={255,127,0},
4623            smooth=Smooth.None));
4624        connect(basketService.outAccumulatedLosses, sum1.u[1]) annotation (Line(
4625            points={{-33.13,60.49},{-45.565,60.49},{-45.565,66},{-55.15,66}},
```

```
4626            color={255,127,0},
4627            smooth=Smooth.None));
4628      connect(sum1.y, outAccumulatedLossesServices) annotation (Line(
4629            points={{-52,78.9},{-62,78.9},{-62,84},{-26,84}},
4630            color={255,127,0},
4631            smooth=Smooth.None));
4632      connect(outMaxFIFOCapacity, outMaxFIFOCapacity) annotation (Line(
4633            points={{98,58},{98,58}},
4634            color={0,0,127},
4635            smooth=Smooth.None));
4636      connect(paymentService.outAccumulatedLosses, sum1.u[2]) annotation (Line(
4637            points={{4.74,60.49},{-20.11,60.49},{-20.11,66},{-53.05,66}},
4638            color={255,127,0},
4639            smooth=Smooth.None));
4640      connect(paymentService1_1.basketOut, recoCircuit5_2.portIn2) annotation (
4641            Line(
4642            points={{18.6,-51.26},{18.6,-28.63},{11.8,-28.63},{11.8,-4.76}},
4643            color={255,128,0},
4644            smooth=Smooth.None));
4645      connect(paymentService1_1.basketIn, recoCircuit5_1.portOut2) annotation (
4646            Line(
4647            points={{-9.4,-51.26},{-9.4,-28.63},{2.04,-28.63},{2.04,-5.04}},
4648            color={255,128,0},
4649            smooth=Smooth.None));
4650      connect(paymentService1_1.outAccumulatedLosses, sum1.u[3]) annotation (Line(
4651            points={{4.74,-41.51},{4.74,11.245},{-50.95,11.245},{-50.95,66}},
4652            color={255,127,0},
4653            smooth=Smooth.None));
4654      connect(recoCircuit5_2.portIn1, paymentService.basketOut) annotation (Line(
4655            points={{11.8,4.48},{11.8,27.24},{18.6,27.24},{18.6,50.74}},
4656            color={255,128,0},
4657            smooth=Smooth.None));
4658      connect(recoCircuit4_1.portOut2, basketService1_1.singleOrderIn)
4659        annotation (Line(
4660            points={{-39.24,-5.32},{-39.24,-28.66},{-44.7,-28.66},{-44.7,-51}},
4661            color={255,128,0},
4662            smooth=Smooth.None));
4663      connect(basketService1_1.basketServiceOut, recoCircuit5_1.portIn2)
4664        annotation (Line(
4665            points={{-21.04,-51},{-21.04,-28.5},{-26.4,-28.5},{-26.4,-4.76}},
4666            color={255,128,0},
4667            smooth=Smooth.None));
4668      connect(basketService1_1.outAccumulatedLosses, sum1.u[4]) annotation (Line(
4669            points={{-33.13,-41.51},{-33.13,11.245},{-48.85,11.245},{-48.85,66}},
4670            color={255,127,0},
4671            smooth=Smooth.None));
4672      annotation (Icon(graphics={Rectangle(
4673              extent={{-100,64},{100,-76}},
4674              lineColor={0,0,255},
4675              lineThickness=0.5), Text(
4676              extent={{-60,10},{60,-26}},
4677              lineColor={0,0,0},
4678              textString="Case2")}),
4679                                      Diagram(graphics),
4680        DymolaStoredErrors);
4681    end C2;
4682
```

```
4683  model Case1
4684
4685    Client client
4686      "Receives the basket ID, finds the cost and starts the businness process flow"
4687      annotation (Placement(transformation(extent={{-100,4},{-76,32}})));
4688    BasketService basketService(
4689      mX=2,
4690      mY=3,
4691      mZ=1)
4692      annotation (Placement(transformation(extent={{-48,38},{-22,64}})));
4693    PaymentService paymentService(
4694      mX=3,
4695      mY=2,
4696      mZ=1)
4697      annotation (Placement(transformation(extent={{-8,38},{20,64}})));
4698    CollectOrderService collectOrderService
4699      annotation (Placement(transformation(extent={{30,38},{54,64}})));
4700    SendMailService sendMailService(
4701      mX=4,
4702      mY=2,
4703      mZ=1)
4704      annotation (Placement(transformation(extent={{72,-52},{98,-26}})));
4705    ReoCircuit1 reoCircuit1_1
4706      annotation (Placement(transformation(extent={{-80,-18},{-44,12}})));
4707    CollectOrderService collectOrderService1
4708      annotation (Placement(transformation(extent={{28,-64},{54,-38}})));
4709    ReoCircuit2 reoCircuit2_1
4710      annotation (Placement(transformation(extent={{-34,4},{2,32}})));
4711    ReoCircuit2 reoCircuit2_2
4712      annotation (Placement(transformation(extent={{-34,-32},{2,-4}})));
4713    ReoCircuit2 reoCircuit2_3
4714      annotation (Placement(transformation(extent={{8,-32},{44,-4}})));
4715    ReoCircuit2 reoCircuit2_4
4716      annotation (Placement(transformation(extent={{8,4},{44,32}})));
4717    ReoCircuit3 reoCircuit3_1
4718      annotation (Placement(transformation(extent={{52,-20},{88,10}})));
4719    Modelica.Blocks.Sources.IntegerTable idTable(table=[6,27; 8,13; 13,2; 15,1;
4720          16,7; 34,17; 37,19; 47,5; 48,4; 51,20; 52,22; 54,27; 55,3; 60,2; 61,8;
4721          62,21; 65,22; 72,3; 73,4; 77,13; 80,21; 84,25; 89,17; 93,1; 94,21; 99,
4722          16; 103,1; 107,9; 108,12; 110,6; 116,23; 117,20; 120,8; 128,4; 134,22;
4723          136,18; 138,15; 139,25; 143,20; 145,9; 147,11; 155,15; 160,14; 169,28;
4724          170,24; 178,16; 184,12; 185,5; 189,13; 194,15; 197,22; 204,18; 208,11;
4725          211,24; 212,10; 218,5; 220,9; 222,19; 223,14; 224,17; 231,26; 232,25;
4726          237,13; 243,27; 245,21; 251,7; 252,3; 253,16; 257,27; 260,17; 263,20;
4727          264,19; 265,18; 269,8; 280,11; 282,13; 283,3; 289,22; 294,7; 295,11;
4728          296,5; 297,4; 302,19; 308,9; 311,3; 312,4; 313,2; 314,11; 318,6; 319,
4729          16; 320,1; 322,18; 323,10; 326,15; 328,1; 343,23; 345,7; 351,8; 352,
4730          26; 356,20])
4731      annotation (Placement(transformation(extent={{-100,54},{-80,74}})));
4732    Modelica.Blocks.Sources.BooleanTable flushTable(table={36,71,107,142,178,
4733          214,249,285,319,356,420})
4734      annotation (Placement(transformation(extent={{60,80},{80,100}})));
4735    Modelica.Blocks.Sources.BooleanTable flushTable1(table={36,71,107,142,178,
4736          214,249,285,319,356,420})
4737      annotation (Placement(transformation(extent={{28,-100},{48,-80}})));
4738    Modelica.Blocks.Sources.IntegerTable priceTable(table=[6,5067; 8,8649; 13,
4739          5310; 15,7193; 16,3134; 34,8184; 37,4880; 47,3561; 48,4117; 51,3534;
```

```
4740            52,2637; 54,5026; 55,3777; 60,4094; 61,5703; 62,4837; 65,5612; 72,
4741            5340; 73,5367; 77,4716; 80,6284; 84,6421; 89,6115; 93,4608; 94,5256;
4742            99,5751; 103,3032; 107,8680; 108,5300; 110,5008; 116,9586; 117,3009;
4743            120,8268; 128,6240; 134,9754; 136,6316; 138,6993; 139,6836; 143,4227;
4744            145,4527; 147,5474; 155,5791; 160,4625; 169,0; 170,7791; 178,4276;
4745            184,0; 185,5091; 189,4371; 194,6086; 197,6159; 204,4849; 208,5798;
4746            211,0; 212,6244; 218,4639; 220,4242; 222,5856; 223,0; 224,2389; 231,
4747            5751; 232,0; 237,6311; 243,3972; 245,0; 251,6850; 252,4529; 253,2659;
4748            257,0; 260,0; 263,4286; 264,3086; 265,3506; 269,3376; 280,3766; 282,0;
4749            283,5867; 289,0; 294,4265; 295,5686; 296,0; 297,1662; 302,0; 308,0;
4750            311,0; 312,0; 313,0; 314,0; 318,0; 319,0; 320,3105; 322,0; 323,0; 326,
4751            0; 328,0; 343,0; 345,0; 351,0; 352,0; 356,0])
4752          annotation (Placement(transformation(extent={{-100,80},{-80,100}})));
4753        Modelica.Blocks.Interaction.Show.IntegerValue integerValue
4754          annotation (Placement(transformation(extent={{-88,-26},{-76,-14}})));
4755        Modelica.Blocks.Interaction.Show.IntegerValue integerValue1
4756          annotation (Placement(transformation(extent={{76,-68},{88,-56}})));
4757        Modelica.Blocks.Interaction.Show.RealValue realValue
4758          annotation (Placement(transformation(extent={{-20,-4},{-10,8}})));
4759        Modelica.Blocks.Interaction.Show.RealValue realValue1
4760          annotation (Placement(transformation(extent={{22,-4},{32,8}})));
4761        Modelica.Blocks.Interaction.Show.RealValue realValue2
4762          annotation (Placement(transformation(extent={{-22,-44},{-12,-32}})));
4763        Modelica.Blocks.Interaction.Show.RealValue realValue3
4764          annotation (Placement(transformation(extent={{24,-44},{34,-32}})));
4765        Modelica.Blocks.Interaction.Show.RealValue realValue4
4766          annotation (Placement(transformation(extent={{62,-30},{72,-18}})));
4767        Modelica.Blocks.Interaction.Show.RealValue realValue5
4768          annotation (Placement(transformation(extent={{-64,-30},{-54,-18}})));
4769        Modelica.Blocks.Interaction.Show.IntegerValue integerValue2
4770          annotation (Placement(transformation(extent={{-4,-4},{8,8}})));
4771        Modelica.Blocks.Interaction.Show.IntegerValue integerValue3
4772          annotation (Placement(transformation(extent={{38,-4},{50,8}})));
4773        Modelica.Blocks.Interaction.Show.IntegerValue integerValue4
4774          annotation (Placement(transformation(extent={{-50,-24},{-38,-12}})));
4775        Modelica.Blocks.Interaction.Show.IntegerValue integerValue5
4776          annotation (Placement(transformation(extent={{40,-38},{52,-26}})));
4777        Modelica.Blocks.Interaction.Show.IntegerValue integerValue6
4778          annotation (Placement(transformation(extent={{-2,-36},{10,-24}})));
4779        Modelica.Blocks.Interaction.Show.IntegerValue integerValue7
4780          annotation (Placement(transformation(extent={{86,-24},{98,-12}})));
4781        Modelica.Blocks.Interaction.Show.IntegerValue integerValue8
4782          annotation (Placement(transformation(extent={{-48,72},{-36,84}})));
4783        Modelica.Blocks.Interaction.Show.IntegerValue integerValue9
4784          annotation (Placement(transformation(extent={{-66,-42},{-54,-30}})));
4785        Modelica.Blocks.Interaction.Show.IntegerValue integerValue10
4786          annotation (Placement(transformation(extent={{20,70},{32,82}})));
4787        Modelica.Blocks.Interaction.Show.IntegerValue integerValue11
4788          annotation (Placement(transformation(extent={{22,-76},{34,-64}})));
4789        PaymentService1 paymentService1_1(
4790          mX=2,
4791          mY=3,
4792          mZ=4) annotation (Placement(transformation(extent={{-8,-64},{20,-38}})));
4793        BasketService1 basketService1_1(
4794          mX=1,
4795          mY=2,
4796          mZ=3) annotation (Placement(transformation(extent={{-48,-64},{-22,-38}})));
```

```
4797    equation
4798
4799      connect(flushTable.y, collectOrderService.emptyEvent)    annotation (Line(
4800          points={{81,90},{88,90},{88,64},{41.88,64},{41.88,60.49}},
4801          color={255,0,255},
4802          smooth=Smooth.None));
4803      connect(flushTable1.y, collectOrderService1.emptyEvent)    annotation (Line(
4804          points={{49,-90},{58,-90},{58,-41.51},{40.87,-41.51}},
4805          color={255,0,255},
4806          smooth=Smooth.None));
4807      connect(client.singleOrderOut, reoCircuit1_1.portIn) annotation (Line(
4808          points={{-77.2,17.72},{-77.2,12.86},{-78.2,12.86},{-78.2,-3.3}},
4809          color={255,128,0},
4810          smooth=Smooth.None));
4811      connect(reoCircuit3_1.portOut, sendMailService.mailServiceIn) annotation (
4812          Line(
4813          points={{84.76,-3.5},{84.76,-21.75},{73.3,-21.75},{73.3,-39.26}},
4814          color={255,128,0},
4815          smooth=Smooth.None));
4816      connect(paymentService.basketIn, reoCircuit2_1.portOut) annotation (Line(
4817          points={{-9.4,50.74},{-9.4,34.37},{0.2,34.37},{0.2,18.84}},
4818          color={255,128,0},
4819          smooth=Smooth.None));
4820      connect(paymentService.basketOut, reoCircuit2_4.portIn) annotation (Line(
4821          points={{18.6,50.74},{18.6,35.37},{9.8,35.37},{9.8,18.56}},
4822          color={255,128,0},
4823          smooth=Smooth.None));
4824      connect(collectOrderService.accountStatementOut, reoCircuit3_1.portIn2)
4825        annotation (Line(
4826          points={{53.04,50.74},{53.04,25.37},{57.4,25.37},{57.4,1.3}},
4827          color={255,128,0},
4828          smooth=Smooth.None));
4829      connect(collectOrderService1.accountStatementOut, reoCircuit3_1.portIn1)
4830        annotation (Line(
4831          points={{52.96,-51.26},{52.96,-29.63},{57.4,-29.63},{57.4,-8.9}},
4832          color={255,128,0},
4833          smooth=Smooth.None));
4834      connect(collectOrderService.basketIn, reoCircuit2_4.portOut) annotation (
4835          Line(
4836          points={{31.44,51},{31.44,34.37},{42.2,34.37},{42.2,18.84}},
4837          color={255,128,0},
4838          smooth=Smooth.None));
4839      connect(collectOrderService1.basketIn, reoCircuit2_3.portOut) annotation (
4840          Line(
4841          points={{29.56,-51},{29.56,-36.63},{42.2,-36.63},{42.2,-17.16}},
4842          color={255,128,0},
4843          smooth=Smooth.None));
4844      connect(priceTable.y, client.singleOrderPriceIn1) annotation (Line(
4845          points={{-79,90},{-68,90},{-68,46},{-83.56,46},{-83.56,28.22}},
4846          color={255,127,0},
4847          smooth=Smooth.None));
4848      connect(idTable.y, client.singleOrderIDIn) annotation (Line(
4849          points={{-79,64},{-76,64},{-76,52},{-94.84,52},{-94.84,28.22}},
4850          color={255,127,0},
4851          smooth=Smooth.None));
4852
4853      connect(reoCircuit1_1.portOut1, basketService.singleOrderIn) annotation (
```

```
4854            Line(
4855            points={{-47.96,2.1},{-47.96,26.05},{-46.7,26.05},{-46.7,51}},
4856            color={255,128,0},
4857            smooth=Smooth.None));
4858        connect(basketService.basketServiceOut, reoCircuit2_1.portIn) annotation (
4859            Line(
4860            points={{-23.04,51},{-23.04,33.5},{-32.2,33.5},{-32.2,18.56}},
4861            color={255,128,0},
4862            smooth=Smooth.None));
4863        connect(client.outputCT, integerValue.numberPort) annotation (Line(
4864            points={{-89.32,7.78},{-89.32,-7.11},{-88.9,-7.11},{-88.9,-20}},
4865            color={255,127,0},
4866            smooth=Smooth.None));
4867        connect(sendMailService.outputCT, integerValue1.numberPort) annotation (
4868            Line(
4869            points={{86.43,-48.49},{86.43,-55.245},{75.1,-55.245},{75.1,-62}},
4870            color={255,127,0},
4871            smooth=Smooth.None));
4872        connect(reoCircuit2_1.currentFIFOCapacity, realValue.numberPort)
4873          annotation (Line(
4874            points={{-19.6,9.32},{-19.6,4.8},{-20.75,4.8},{-20.75,2}},
4875            color={0,0,127},
4876            smooth=Smooth.None));
4877        connect(reoCircuit2_4.currentFIFOCapacity, realValue1.numberPort)
4878          annotation (Line(
4879            points={{22.4,9.32},{22.4,4.8},{21.25,4.8},{21.25,2}},
4880            color={0,0,127},
4881            smooth=Smooth.None));
4882        connect(reoCircuit2_2.currentFIFOCapacity, realValue2.numberPort)
4883          annotation (Line(
4884            points={{-19.6,-26.68},{-19.6,-34.2},{-22.75,-34.2},{-22.75,-38}},
4885            color={0,0,127},
4886            smooth=Smooth.None));
4887        connect(reoCircuit2_3.currentFIFOCapacity, realValue3.numberPort)
4888          annotation (Line(
4889            points={{22.4,-26.68},{22.4,-35.2},{23.25,-35.2},{23.25,-38}},
4890            color={0,0,127},
4891            smooth=Smooth.None));
4892        connect(reoCircuit3_1.currentFIFOCapacity, realValue4.numberPort)
4893          annotation (Line(
4894            points={{67.12,-13.7},{67.12,-17.85},{61.25,-17.85},{61.25,-24}},
4895            color={0,0,127},
4896            smooth=Smooth.None));
4897        connect(reoCircuit1_1.currentFIFOCapacity, realValue5.numberPort)
4898          annotation (Line(
4899            points={{-65.6,-13.2},{-65.6,-18.6},{-64.75,-18.6},{-64.75,-24}},
4900            color={0,0,127},
4901            smooth=Smooth.None));
4902        connect(reoCircuit2_1.outAccumulatedLosses, integerValue2.numberPort)
4903          annotation (Line(
4904            points={{-12.04,9.32},{-7.02,9.32},{-7.02,2},{-4.9,2}},
4905            color={255,127,0},
4906            smooth=Smooth.None));
4907        connect(reoCircuit2_4.outAccumulatedLosses, integerValue3.numberPort)
4908          annotation (Line(
4909            points={{29.96,9.32},{33.98,9.32},{33.98,2},{37.1,2}},
4910            color={255,127,0},
```

```
4911            smooth=Smooth.None));
4912        connect(reoCircuit1_1.outAccumulatedLosses, integerValue4.numberPort)
4913          annotation (Line(
4914            points={{-58.4,-13.2},{-54.2,-13.2},{-54.2,-18},{-50.9,-18}},
4915            color={255,127,0},
4916            smooth=Smooth.None));
4917        connect(reoCircuit2_3.outAccumulatedLosses, integerValue5.numberPort)
4918          annotation (Line(
4919            points={{29.96,-26.68},{34.98,-26.68},{34.98,-32},{39.1,-32}},
4920            color={255,127,0},
4921            smooth=Smooth.None));
4922        connect(reoCircuit2_2.outAccumulatedLosses, integerValue6.numberPort)
4923          annotation (Line(
4924            points={{-12.04,-26.68},{-8.02,-26.68},{-8.02,-30},{-2.9,-30}},
4925            color={255,127,0},
4926            smooth=Smooth.None));
4927        connect(reoCircuit3_1.outAccumulatedLosses, integerValue7.numberPort)
4928          annotation (Line(
4929            points={{73.96,-13.7},{78.98,-13.7},{78.98,-18},{85.1,-18}},
4930            color={255,127,0},
4931            smooth=Smooth.None));
4932        connect(basketService.outAccumulatedLosses, integerValue8.numberPort)
4933          annotation (Line(
4934            points={{-35.13,60.49},{-35.13,68.245},{-48.9,68.245},{-48.9,78}},
4935            color={255,127,0},
4936            smooth=Smooth.None));
4937        connect(paymentService.outAccumulatedLosses, integerValue10.numberPort)
4938          annotation (Line(
4939            points={{4.74,60.49},{4.74,68.245},{19.1,68.245},{19.1,76}},
4940            color={255,127,0},
4941            smooth=Smooth.None));
4942        connect(paymentService1_1.outAccumulatedLosses, integerValue11.numberPort)
4943          annotation (Line(
4944            points={{4.74,-41.51},{4.74,-56.755},{21.1,-56.755},{21.1,-70}},
4945            color={255,127,0},
4946            smooth=Smooth.None));
4947        connect(paymentService1_1.basketOut, reoCircuit2_3.portIn) annotation (Line(
4948            points={{18.6,-51.26},{18.6,-34.63},{9.8,-34.63},{9.8,-17.44}},
4949            color={255,128,0},
4950            smooth=Smooth.None));
4951        connect(paymentService1_1.basketIn, reoCircuit2_2.portOut) annotation (Line(
4952            points={{-9.4,-51.26},{-9.4,-34.63},{0.2,-34.63},{0.2,-17.16}},
4953            color={255,128,0},
4954            smooth=Smooth.None));
4955        connect(reoCircuit1_1.portOut2, basketService1_1.singleOrderIn) annotation (
4956           Line(
4957            points={{-47.96,-8.7},{-47.96,-30.35},{-46.7,-30.35},{-46.7,-51}},
4958            color={255,128,0},
4959            smooth=Smooth.None));
4960        connect(basketService1_1.outAccumulatedLosses, integerValue9.numberPort)
4961          annotation (Line(
4962            points={{-35.13,-41.51},{-50.565,-41.51},{-50.565,-36},{-66.9,-36}},
4963            color={255,127,0},
4964            smooth=Smooth.None));
4965        connect(basketService1_1.basketServiceOut, reoCircuit2_2.portIn)
4966          annotation (Line(
4967            points={{-23.04,-51},{-23.04,-34.5},{-32.2,-34.5},{-32.2,-17.44}},
```

```
4968          color={255,128,0},
4969          smooth=Smooth.None));
4970      annotation (Diagram(graphics),
4971        experiment(StopTime=450),
4972        __Dymola_experimentSetupOutput(derivatives=false, inputs=false));
4973    end Case1;
4974
4975    model Case2
4976
4977      Client client
4978        "Receives the basket ID, finds the cost and starts the businness process flow"
4979        annotation (Placement(transformation(extent={{-100,16},{-76,44}})));
4980      BasketService basketService(
4981        mX=2,
4982        mY=3,
4983        mZ=1)
4984        annotation (Placement(transformation(extent={{-48,38},{-22,64}})));
4985      PaymentService paymentService(
4986        mX=3,
4987        mY=2,
4988        mZ=1)
4989        annotation (Placement(transformation(extent={{-8,38},{20,64}})));
4990      CollectOrderService collectOrderService
4991        annotation (Placement(transformation(extent={{30,38},{54,64}})));
4992      SendMailService sendMailService(
4993        mX=4,
4994        mY=2,
4995        mZ=1)
4996        annotation (Placement(transformation(extent={{72,-52},{98,-26}})));
4997      CollectOrderService collectOrderService1
4998        annotation (Placement(transformation(extent={{28,-64},{54,-38}})));
4999      Modelica.Blocks.Sources.IntegerTable idTable(table=[6,27; 8,13; 13,2; 15,1;
5000            16,7; 34,17; 37,19; 47,5; 48,4; 51,20; 52,22; 54,27; 55,3; 60,2; 61,8;
5001            62,21; 65,22; 72,3; 73,4; 77,13; 80,21; 84,25; 89,17; 93,1; 94,21; 99,
5002            16; 103,1; 107,9; 108,12; 110,6; 116,23; 117,20; 120,8; 128,4; 134,22;
5003            136,18; 138,15; 139,25; 143,20; 145,9; 147,11; 155,15; 160,14; 169,28;
5004            170,24; 178,16; 184,12; 185,5; 189,13; 194,15; 197,22; 204,18; 208,11;
5005            211,24; 212,10; 218,5; 220,9; 222,19; 223,14; 224,17; 231,26; 232,25;
5006            237,13; 243,27; 245,21; 251,7; 252,3; 253,16; 257,27; 260,17; 263,20;
5007            264,19; 265,18; 269,8; 280,11; 282,13; 283,3; 289,22; 294,7; 295,11;
5008            296,5; 297,4; 302,19; 308,9; 311,3; 312,4; 313,2; 314,11; 318,6; 319,
5009            16; 320,1; 322,18; 323,10; 326,15; 328,1; 343,23; 345,7; 351,8; 352,
5010            26; 356,20])
5011        annotation (Placement(transformation(extent={{-100,54},{-80,74}})));
5012      Modelica.Blocks.Sources.BooleanTable flushTable(table={36,71,107,142,178,
5013            214,249,285,320,356,420})
5014        annotation (Placement(transformation(extent={{60,80},{80,100}})));
5015      Modelica.Blocks.Sources.BooleanTable flushTable1(table={36,71,107,142,178,
5016            214,249,285,320,356,420})
5017        annotation (Placement(transformation(extent={{28,-100},{48,-80}})));
5018      RiskAnalysis.ReoCircuit4
5019                   recoCircuit4_1
5020        annotation (Placement(transformation(extent={{-72,-14},{-36,14}})));
5021      RiskAnalysis.ReoCircuit5
5022                   recoCircuit5_1
5023        annotation (Placement(transformation(extent={{-30,-14},{6,14}})));
5024      RiskAnalysis.ReoCircuit5
```

```
5025                 recoCircuit5_2
5026       annotation (Placement(transformation(extent={{8,-14},{46,14}})));
5027     ReoCircuit6 reoCircuit6_1
5028       annotation (Placement(transformation(extent={{52,-16},{88,14}})));
5029     Modelica.Blocks.Sources.IntegerTable priceTable(table=[6,5067; 8,8649; 13,
5030           5310; 15,7193; 16,3134; 34,8184; 37,4880; 47,3561; 48,4117; 51,3534;
5031           52,2637; 54,5026; 55,3777; 60,4094; 61,5703; 62,4837; 65,5612; 72,
5032           5340; 73,5367; 77,4716; 80,6284; 84,6421; 89,6115; 93,4608; 94,5256;
5033           99,5751; 103,3032; 107,8680; 108,5300; 110,5008; 116,9586; 117,3009;
5034           120,8268; 128,6240; 134,9754; 136,6316; 138,6993; 139,6836; 143,4227;
5035           145,4527; 147,5474; 155,5791; 160,4625; 169,0; 170,7791; 178,4276;
5036           184,0; 185,5091; 189,4371; 194,6086; 197,6159; 204,4849; 208,5798;
5037           211,0; 212,6244; 218,4639; 220,4242; 222,5856; 223,0; 224,2389; 231,
5038           5751; 232,0; 237,6311; 243,3972; 245,0; 251,6850; 252,4529; 253,2659;
5039           257,0; 260,0; 263,4286; 264,3086; 265,3506; 269,3376; 280,3766; 282,0;
5040           283,5867; 289,0; 294,4265; 295,5686; 296,0; 297,1662; 302,0; 308,0;
5041           311,0; 312,0; 313,0; 314,0; 318,0; 319,0; 320,3105; 322,0; 323,0; 326,
5042           0; 328,0; 343,0; 345,0; 351,0; 352,0; 356,0])
5043       annotation (Placement(transformation(extent={{-100,80},{-80,100}})));
5044     Modelica.Blocks.Interaction.Show.IntegerValue integerValue
5045       annotation (Placement(transformation(extent={{-94,-10},{-82,2}})));
5046     Modelica.Blocks.Interaction.Show.IntegerValue integerValue1
5047       annotation (Placement(transformation(extent={{74,-66},{86,-54}})));
5048     Modelica.Blocks.Interaction.Show.RealValue realValue5
5049       annotation (Placement(transformation(extent={{-68,-24},{-58,-12}})));
5050     Modelica.Blocks.Interaction.Show.RealValue realValue1
5051       annotation (Placement(transformation(extent={{-22,-24},{-12,-12}})));
5052     Modelica.Blocks.Interaction.Show.RealValue realValue2
5053       annotation (Placement(transformation(extent={{18,-24},{28,-12}})));
5054     Modelica.Blocks.Interaction.Show.RealValue realValue3
5055       annotation (Placement(transformation(extent={{60,-24},{70,-12}})));
5056     Modelica.Blocks.Interaction.Show.IntegerValue integerValue2
5057       annotation (Placement(transformation(extent={{-48,-24},{-36,-12}})));
5058     Modelica.Blocks.Interaction.Show.IntegerValue integerValue3
5059       annotation (Placement(transformation(extent={{-4,-24},{8,-12}})));
5060     Modelica.Blocks.Interaction.Show.IntegerValue integerValue4
5061       annotation (Placement(transformation(extent={{36,-24},{48,-12}})));
5062     Modelica.Blocks.Interaction.Show.IntegerValue integerValue5
5063       annotation (Placement(transformation(extent={{82,-20},{94,-8}})));
5064     Modelica.Blocks.Interaction.Show.IntegerValue integerValue6
5065       annotation (Placement(transformation(extent={{-44,70},{-32,82}})));
5066     Modelica.Blocks.Interaction.Show.IntegerValue integerValue7
5067       annotation (Placement(transformation(extent={{-58,-34},{-46,-22}})));
5068     Modelica.Blocks.Interaction.Show.IntegerValue integerValue8
5069       annotation (Placement(transformation(extent={{16,68},{28,80}})));
5070     Modelica.Blocks.Interaction.Show.IntegerValue integerValue9
5071       annotation (Placement(transformation(extent={{8,-40},{20,-28}})));
5072
5073     PaymentService1 paymentService1_1(
5074       mX=2,
5075       mY=3,
5076       mZ=4) annotation (Placement(transformation(extent={{-8,-64},{20,-38}})));
5077     BasketService1 basketService1_1(
5078       mX=1,
5079       mY=2,
5080       mZ=3) annotation (Placement(transformation(extent={{-48,-62},{-22,-36}})));
5081   equation
```

```
5082      connect(flushTable.y, collectOrderService.emptyEvent)    annotation (Line(
5083          points={{81,90},{88,90},{88,64},{41.88,64},{41.88,60.49}},
5084          color={255,0,255},
5085          smooth=Smooth.None));
5086      connect(flushTable1.y, collectOrderService1.emptyEvent)    annotation (Line(
5087          points={{49,-90},{58,-90},{58,-41.51},{40.87,-41.51}},
5088          color={255,0,255},
5089          smooth=Smooth.None));
5090      connect(client.singleOrderOut, recoCircuit4_1.portIn) annotation (Line(
5091          points={{-77.2,29.72},{-77.2,14.86},{-69.48,14.86},{-69.48,-0.28}},
5092          color={255,128,0},
5093          smooth=Smooth.None));
5094      connect(recoCircuit5_1.portOut1, paymentService.basketIn) annotation (Line(
5095          points={{2.4,4.48},{2.4,27.52},{-9.4,27.52},{-9.4,50.74}},
5096          color={255,128,0},
5097          smooth=Smooth.None));
5098      connect(recoCircuit5_2.portOut1, collectOrderService.basketIn) annotation (
5099          Line(
5100          points={{42.2,4.48},{42.2,27.52},{31.44,27.52},{31.44,51}},
5101          color={255,128,0},
5102          smooth=Smooth.None));
5103      connect(recoCircuit5_2.portOut2, collectOrderService1.basketIn) annotation (
5104         Line(
5105          points={{41.82,-5.04},{41.82,-28.66},{29.56,-28.66},{29.56,-51}},
5106          color={255,128,0},
5107          smooth=Smooth.None));
5108      connect(collectOrderService.accountStatementOut, reoCircuit6_1.portIn2)
5109        annotation (Line(
5110          points={{53.04,50.74},{53.04,27.37},{55.24,27.37},{55.24,-5.8}},
5111          color={255,128,0},
5112          smooth=Smooth.None));
5113      connect(collectOrderService1.accountStatementOut, reoCircuit6_1.portIn1)
5114        annotation (Line(
5115          points={{52.96,-51.26},{52.96,-28.63},{55.24,-28.63},{55.24,4.7}},
5116          color={255,128,0},
5117          smooth=Smooth.None));
5118      connect(reoCircuit6_1.portOut, sendMailService.mailServiceIn) annotation (
5119          Line(
5120          points={{85.48,-0.4},{85.48,-20.2},{73.3,-20.2},{73.3,-39.26}},
5121          color={255,128,0},
5122          smooth=Smooth.None));
5123      connect(priceTable.y, client.singleOrderPriceIn1) annotation (Line(
5124          points={{-79,90},{-68,90},{-68,46},{-83.56,46},{-83.56,40.22}},
5125          color={255,127,0},
5126          smooth=Smooth.None));
5127      connect(idTable.y, client.singleOrderIDIn) annotation (Line(
5128          points={{-79,64},{-74,64},{-74,52},{-94,52},{-94.84,40.22}},
5129          color={255,127,0},
5130          smooth=Smooth.None));
5131      connect(client.outputCT, integerValue.numberPort) annotation (Line(
5132          points={{-89.32,19.78},{-89.32,4.89},{-94.9,4.89},{-94.9,-4}},
5133          color={255,127,0},
5134          smooth=Smooth.None));
5135      connect(sendMailService.outputCT, integerValue1.numberPort) annotation (
5136          Line(
5137          points={{86.43,-48.49},{86.43,-54.245},{73.1,-54.245},{73.1,-60}},
5138          color={255,127,0},
```

```
5139            smooth=Smooth.None));
5140        connect(recoCircuit6_1.currentFIFOCapacity, realValue3.numberPort)
5141          annotation (Line(
5142            points={{66.04,-10.6},{66.04,-14.3},{59.25,-14.3},{59.25,-18}},
5143            color={0,0,127},
5144            smooth=Smooth.None));
5145        connect(recoCircuit5_2.currentFIFOCapacity, realValue2.numberPort)
5146          annotation (Line(
5147            points={{24.34,-9.52},{24.34,-14.76},{17.25,-14.76},{17.25,-18}},
5148            color={0,0,127},
5149            smooth=Smooth.None));
5150        connect(recoCircuit5_1.currentFIFOCapacity, realValue1.numberPort)
5151          annotation (Line(
5152            points={{-14.52,-9.52},{-14.52,-14.76},{-22.75,-14.76},{-22.75,-18}},
5153            color={0,0,127},
5154            smooth=Smooth.None));
5155        connect(recoCircuit4_1.currentFIFOCapacity, realValue5.numberPort)
5156          annotation (Line(
5157            points={{-56.52,-9.8},{-72.26,-9.8},{-72.26,-18},{-68.75,-18}},
5158            color={0,0,127},
5159            smooth=Smooth.None));
5160        connect(recoCircuit4_1.outAccumulatedLosses, integerValue2.numberPort)
5161          annotation (Line(
5162            points={{-50.04,-9.8},{-50.04,-13.9},{-48.9,-13.9},{-48.9,-18}},
5163            color={255,127,0},
5164            smooth=Smooth.None));
5165        connect(recoCircuit5_1.outAccumulatedLosses, integerValue3.numberPort)
5166          annotation (Line(
5167            points={{-8.4,-9.52},{-8.4,-13.76},{-4.9,-13.76},{-4.9,-18}},
5168            color={255,127,0},
5169            smooth=Smooth.None));
5170        connect(recoCircuit5_2.outAccumulatedLosses, integerValue4.numberPort)
5171          annotation (Line(
5172            points={{30.8,-9.52},{30.8,-13.76},{35.1,-13.76},{35.1,-18}},
5173            color={255,127,0},
5174            smooth=Smooth.None));
5175        connect(recoCircuit6_1.outAccumulatedLosses, integerValue5.numberPort)
5176          annotation (Line(
5177            points={{73.24,-10.6},{76.62,-10.6},{76.62,-14},{81.1,-14}},
5178            color={255,127,0},
5179            smooth=Smooth.None));
5180        connect(basketService.outAccumulatedLosses, integerValue6.numberPort)
5181          annotation (Line(
5182            points={{-35.13,60.49},{-35.13,67.245},{-44.9,67.245},{-44.9,76}},
5183            color={255,127,0},
5184            smooth=Smooth.None));
5185        connect(paymentService.outAccumulatedLosses, integerValue8.numberPort)
5186          annotation (Line(
5187            points={{4.74,60.49},{4.74,66.245},{15.1,66.245},{15.1,74}},
5188            color={255,127,0},
5189            smooth=Smooth.None));
5190
5191        connect(paymentService1_1.outAccumulatedLosses, integerValue9.numberPort)
5192          annotation (Line(
5193            points={{4.74,-41.51},{4.74,-37.755},{7.1,-37.755},{7.1,-34}},
5194            color={255,127,0},
5195            smooth=Smooth.None));
```

```
5196      connect(paymentService1_1.basketOut, recoCircuit5_2.portIn2) annotation (
5197          Line(
5198          points={{18.6,-51.26},{18.6,-28.63},{11.8,-28.63},{11.8,-4.76}},
5199          color={255,128,0},
5200          smooth=Smooth.None));
5201      connect(paymentService1_1.basketIn, recoCircuit5_1.portOut2) annotation (
5202          Line(
5203          points={{-9.4,-51.26},{-9.4,-28.63},{2.04,-28.63},{2.04,-5.04}},
5204          color={255,128,0},
5205          smooth=Smooth.None));
5206      connect(basketService.basketServiceOut, recoCircuit5_1.portIn1) annotation (
5207         Line(
5208          points={{-23.04,51},{-23.04,28.5},{-26.4,28.5},{-26.4,4.48}},
5209          color={255,128,0},
5210          smooth=Smooth.None));
5211      connect(recoCircuit4_1.portOut1, basketService.singleOrderIn) annotation (
5212          Line(
5213          points={{-39.24,4.48},{-39.24,27.24},{-46.7,27.24},{-46.7,51}},
5214          color={255,128,0},
5215          smooth=Smooth.None));
5216      connect(recoCircuit5_2.portIn1, paymentService.basketOut) annotation (Line(
5217          points={{11.8,4.48},{11.8,27.24},{18.6,27.24},{18.6,50.74}},
5218          color={255,128,0},
5219          smooth=Smooth.None));
5220      connect(basketService1_1.outAccumulatedLosses, integerValue7.numberPort)
5221        annotation (Line(
5222          points={{-35.13,-39.51},{-47.565,-39.51},{-47.565,-28},{-58.9,-28}},
5223          color={255,127,0},
5224          smooth=Smooth.None));
5225      connect(basketService1_1.singleOrderIn, recoCircuit4_1.portOut2)
5226        annotation (Line(
5227          points={{-46.7,-49},{-46.7,-28.5},{-39.24,-28.5},{-39.24,-5.32}},
5228          color={255,128,0},
5229          smooth=Smooth.None));
5230      connect(basketService1_1.basketServiceOut, recoCircuit5_1.portIn2)
5231        annotation (Line(
5232          points={{-23.04,-49},{-23.04,-27.5},{-26.4,-27.5},{-26.4,-4.76}},
5233          color={255,128,0},
5234          smooth=Smooth.None));
5235      annotation (Diagram(graphics),
5236        experiment(StopTime=400),
5237        __Dymola_experimentSetupOutput(inputs=false, outputs=false));
5238    end Case2;
5239
5240    model Case3
5241
5242      Client client
5243        "Receives the basket ID, finds the cost and starts the businness process flow"
5244        annotation (Placement(transformation(extent={{-92,-10},{-68,18}})));
5245      Modelica.Blocks.Sources.IntegerTable idTable(table=[6,27; 8,13; 13,2; 15,1;
5246            16,7; 34,17; 37,19; 47,5; 48,4; 51,20; 52,22; 54,27; 55,3; 60,2; 61,8;
5247            62,21; 65,22; 72,3; 73,4; 77,13; 80,21; 84,25; 89,17; 93,1; 94,21; 99,
5248            16; 103,1; 107,9; 108,12; 110,6; 116,23; 117,20; 120,8; 128,4; 134,22;
5249            136,18; 138,15; 139,25; 143,20; 145,9; 147,11; 155,15; 160,14; 169,28;
5250            170,24; 178,16; 184,12; 185,5; 189,13; 194,15; 197,22; 204,18; 208,11;
5251            211,24; 212,10; 218,5; 220,9; 222,19; 223,14; 224,17; 231,26; 232,25;
5252            237,13; 243,27; 245,21; 251,7; 252,3; 253,16; 257,27; 260,17; 263,20;
```

```
5253            264,19; 265,18; 269,8; 280,11; 282,13; 283,3; 289,22; 294,7; 295,11;
5254            296,5; 297,4; 302,19; 308,9; 311,3; 312,4; 313,2; 314,11; 318,6; 319,
5255            16; 320,1; 322,18; 323,10; 326,15; 328,1; 343,23; 345,7; 351,8; 352,
5256            26; 356,20])
5257          annotation (Placement(transformation(extent={{-92,34},{-72,54}})));
5258        Modelica.Blocks.Sources.IntegerTable priceTable(table=[6,5067; 8,8649; 13,
5259            5310; 15,7193; 16,3134; 34,8184; 37,4880; 47,3561; 48,4117; 51,3534;
5260            52,2637; 54,5026; 55,3777; 60,4094; 61,5703; 62,4837; 65,5612; 72,
5261            5340; 73,5367; 77,4716; 80,6284; 84,6421; 89,6115; 93,4608; 94,5256;
5262            99,5751; 103,3032; 107,8680; 108,5300; 110,5008; 116,9586; 117,3009;
5263            120,8268; 128,6240; 134,9754; 136,6316; 138,6993; 139,6836; 143,4227;
5264            145,4527; 147,5474; 155,5791; 160,4625; 169,0; 170,7791; 178,4276;
5265            184,0; 185,5091; 189,4371; 194,6086; 197,6159; 204,4849; 208,5798;
5266            211,0; 212,6244; 218,4639; 220,4242; 222,5856; 223,0; 224,2389; 231,
5267            5751; 232,0; 237,6311; 243,3972; 245,0; 251,6850; 252,4529; 253,2659;
5268            257,0; 260,0; 263,4286; 264,3086; 265,3506; 269,3376; 280,3766; 282,0;
5269            283,5867; 289,0; 294,4265; 295,5686; 296,0; 297,1662; 302,0; 308,0;
5270            311,0; 312,0; 313,0; 314,0; 318,0; 319,0; 320,3105; 322,0; 323,0; 326,
5271            0; 328,0; 343,0; 345,0; 351,0; 352,0; 356,0])
5272          annotation (Placement(transformation(extent={{-64,54},{-44,74}})));
5273        ReoCircuit7 reoCircuit7_1
5274          annotation (Placement(transformation(extent={{-42,-10},{-6,20}})));
5275        RiskAnalysis.C1
5276           m1_1 annotation (Placement(transformation(extent={{8,12},{36,38}})));
5277        RiskAnalysis.C2
5278           m2_1 annotation (Placement(transformation(extent={{8,-38},{36,-12}})));
5279        Modelica.Blocks.Interaction.Show.IntegerValue integerValue1
5280          annotation (Placement(transformation(extent={{44,50},{56,62}})));
5281        Modelica.Blocks.Interaction.Show.RealValue realValue
5282          annotation (Placement(transformation(extent={{24,50},{34,62}})));
5283        Modelica.Blocks.Interaction.Show.IntegerValue integerValue2
5284          annotation (Placement(transformation(extent={{42,-10},{54,2}})));
5285        Modelica.Blocks.Interaction.Show.IntegerValue integerValue3
5286          annotation (Placement(transformation(extent={{-82,-36},{-70,-24}})));
5287        Modelica.Blocks.Interaction.Show.RealValue realValue1
5288          annotation (Placement(transformation(extent={{24,-10},{34,2}})));
5289        Modelica.Blocks.Interaction.Show.IntegerValue integerValue5
5290          annotation (Placement(transformation(extent={{-8,46},{4,58}})));
5291        Modelica.Blocks.Interaction.Show.IntegerValue integerValue6
5292          annotation (Placement(transformation(extent={{0,-10},{12,2}})));
5293        Modelica.Blocks.Interaction.Show.RealValue realValue2
5294          annotation (Placement(transformation(extent={{-22,26},{-12,38}})));
5295        Modelica.Blocks.Sources.IntegerExpression integerExpression(y=1)
5296          annotation (Placement(transformation(extent={{-52,-28},{-40,-16}})));
5297        Modelica.Blocks.Sources.IntegerExpression integerExpression1(y=1)
5298          annotation (Placement(transformation(extent={{-38,-50},{-26,-38}})));
5299        Modelica.Blocks.Interaction.Show.IntegerValue integerValue4
5300          annotation (Placement(transformation(extent={{8,56},{20,68}})));
5301        Modelica.Blocks.Interaction.Show.IntegerValue integerValue7
5302          annotation (Placement(transformation(extent={{16,-2},{28,10}})));
5303      equation
5304        connect(priceTable.y,client. singleOrderPriceIn1) annotation (Line(
5305            points={{-43,64},{-42,64},{-42,20},{-75.56,20},{-75.56,14.22}},
5306            color={255,127,0},
5307            smooth=Smooth.None));
5308        connect(idTable.y,client. singleOrderIDIn) annotation (Line(
5309            points={{-71,44},{-66,44},{-66,26},{-86.84,26},{-86.84,14.22}},
```

```
5310            color={255,127,0},
5311            smooth=Smooth.None));
5312        connect(client.singleOrderOut, reoCircuit7_1.portIn) annotation (Line(
5313            points={{-69.2,3.72},{-54,4},{-39.48,4.7}},
5314            color={255,128,0},
5315            smooth=Smooth.None));
5316        connect(reoCircuit7_1.portOut1, m1_1.servicePortIn) annotation (Line(
5317            points={{-9.24,9.8},{-9.24,26},{2,26},{2,23.83},{7.44,23.83}},
5318            color={255,128,0},
5319            smooth=Smooth.None));
5320        connect(reoCircuit7_1.portOut2, m2_1.servicePortIn) annotation (Line(
5321            points={{-9.6,-1.3},{-9.6,-24},{2,-24},{2,-25.91},{7.72,-25.91}},
5322            color={255,128,0},
5323            smooth=Smooth.None));
5324        connect(m1_1.outSendMailService, integerValue1.numberPort) annotation (Line(
5325            points={{31.1,35.01},{31.1,44.505},{43.1,44.505},{43.1,56}},
5326            color={255,127,0},
5327            smooth=Smooth.None));
5328        connect(m1_1.outMaxFIFOCapacity, realValue.numberPort) annotation (Line(
5329            points={{25.22,35.01},{25.22,45.505},{23.25,45.505},{23.25,56}},
5330            color={0,0,127},
5331            smooth=Smooth.None));
5332        connect(m2_1.outSendMailService, integerValue2.numberPort) annotation (Line(
5333            points={{31.1,-14.73},{31.1,-9.365},{41.1,-9.365},{41.1,-4}},
5334            color={255,127,0},
5335            smooth=Smooth.None));
5336        connect(client.outputCT, integerValue3.numberPort) annotation (Line(
5337            points={{-81.32,-6.22},{-81.32,-18.11},{-82.9,-18.11},{-82.9,-30}},
5338            color={255,127,0},
5339            smooth=Smooth.None));
5340        connect(m2_1.outMaxFIFOCapacity, realValue1.numberPort) annotation (Line(
5341            points={{25.22,-14.73},{25.22,-7.365},{23.25,-7.365},{23.25,-4}},
5342            color={0,0,127},
5343            smooth=Smooth.None));
5344        connect(reoCircuit7_1.currentFIFOCapacity, realValue2.numberPort)
5345          annotation (Line(
5346            points={{-23.64,14.9},{-23.64,23.45},{-22.75,23.45},{-22.75,32}},
5347            color={0,0,127},
5348            smooth=Smooth.None));
5349        connect(m1_1.outAccumulatedLosses, integerValue5.numberPort) annotation (
5350            Line(
5351            points={{12.9,35.01},{12.9,45.505},{-8.9,45.505},{-8.9,52}},
5352            color={255,127,0},
5353            smooth=Smooth.None));
5354        connect(m2_1.outAccumulatedLosses, integerValue6.numberPort) annotation (
5355            Line(
5356            points={{12.9,-14.73},{12.9,-10.365},{-0.9,-10.365},{-0.9,-4}},
5357            color={255,127,0},
5358            smooth=Smooth.None));
5359        connect(integerExpression1.y, reoCircuit7_1.case1or2) annotation (Line(
5360            points={{-25.4,-44},{-20,-44},{-20,-6.1},{-19.68,-6.1}},
5361            color={255,127,0},
5362            smooth=Smooth.None));
5363        connect(integerExpression.y, reoCircuit7_1.open) annotation (Line(
5364            points={{-39.4,-22},{-38,-22},{-38,-6.1},{-26.88,-6.1}},
5365            color={255,127,0},
5366            smooth=Smooth.None));
```

```
5367      connect(m1_1.outAccumulatedLossesServices, integerValue4.numberPort)
5368        annotation (Line(
5369          points={{18.78,35.01},{18.78,47.505},{7.1,47.505},{7.1,62}},
5370          color={255,127,0},
5371          smooth=Smooth.None));
5372      connect(m2_1.outAccumulatedLossesServices, integerValue7.numberPort)
5373        annotation (Line(
5374          points={{19.06,-14.73},{19.06,-6.365},{15.1,-6.365},{15.1,4}},
5375          color={255,127,0},
5376          smooth=Smooth.None));
5377      annotation (Diagram(graphics),
5378        experiment(StopTime=450),
5379        __Dymola_experimentSetupOutput(derivatives=false, inputs=false));
5380    end Case3;
5381
5382    model C3Simulink
5383
5384      Client client
5385        "Receives the basket ID, finds the cost and starts the businness process flow"
5386        annotation (Placement(transformation(extent={{-74,-14},{-50,14}})));
5387      ReoCircuit7 reoCircuit7_1
5388        annotation (Placement(transformation(extent={{-24,-14},{12,16}})));
5389      C1 m1_1 annotation (Placement(transformation(extent={{26,8},{54,34}})));
5390      C2 m2_1 annotation (Placement(transformation(extent={{26,-42},{54,-16}})));
5391      Modelica.Blocks.Interfaces.IntegerOutput outputClient annotation (Placement(
5392            transformation(
5393            extent={{-10,-10},{10,10}},
5394            rotation=-90,
5395            origin={-54,-60}), iconTransformation(
5396            extent={{-10,-10},{10,10}},
5397            rotation=0,
5398            origin={110,-28})));
5399      Modelica.Blocks.Interfaces.IntegerInput inIdTable annotation (Placement(
5400            transformation(extent={{-102,40},{-62,80}}), iconTransformation(
5401              extent={{-122,30},{-100,52}})));
5402      Modelica.Blocks.Interfaces.IntegerInput inPriceTable annotation (Placement(
5403            transformation(extent={{-70,48},{-30,88}}), iconTransformation(extent
                ={{-122,6},
5404              {-100,28}})));
5405      Modelica.Blocks.Interfaces.IntegerOutput outAccumulatedLossesC1 annotation (
5406          Placement(transformation(
5407            extent={{-10,-10},{10,10}},
5408            rotation=-90,
5409            origin={16,78}), iconTransformation(
5410            extent={{-10,-10},{10,10}},
5411            rotation=90,
5412            origin={-50,70})));
5413      Modelica.Blocks.Interfaces.IntegerOutput outAccumulatedLossesC2 annotation (
5414          Placement(transformation(
5415            extent={{-10,-10},{10,10}},
5416            rotation=-90,
5417            origin={32,-64}), iconTransformation(
5418            extent={{-10,-10},{10,10}},
5419            rotation=-90,
5420            origin={-50,-90})));
5421      Modelica.Blocks.Interfaces.IntegerOutput outSendMailServiceC1 annotation (
5422          Placement(transformation(
```

```
5423            extent={{-10,-10},{10,10}},
5424            rotation=-90,
5425            origin={68,52}), iconTransformation(
5426            extent={{-10,-10},{10,10}},
5427            rotation=90,
5428            origin={52,70})));
5429      Modelica.Blocks.Interfaces.IntegerOutput outSendMailServiceC2 annotation (
5430          Placement(transformation(
5431            extent={{-10,-10},{10,10}},
5432            rotation=-90,
5433            origin={74,-32}), iconTransformation(
5434            extent={{-10,-10},{10,10}},
5435            rotation=270,
5436            origin={52,-90})));
5437      Modelica.Blocks.Interfaces.RealOutput outMaxtFIFOCapacityC1
5438                                                                  annotation (
5439          Placement(transformation(extent={{52,64},{72,84}}),
5440            iconTransformation(
5441            extent={{-10,-10},{10,10}},
5442            rotation=90,
5443            origin={0,70})));
5444      Modelica.Blocks.Interfaces.RealOutput outMaxFIFOCapacityC2
5445                                                                  annotation (
5446          Placement(transformation(extent={{58,-12},{78,8}}),
5447            iconTransformation(
5448            extent={{-10,-10},{10,10}},
5449            rotation=-90,
5450            origin={0,-90})));
5451      Modelica.Blocks.Interfaces.RealOutput currentFIFOCapacityCircuit7
5452                                                                  annotation (
5453          Placement(transformation(extent={{-12,24},{8,44}}),
5454            iconTransformation(
5455            extent={{-10,-10},{10,10}},
5456            rotation=0,
5457            origin={110,10})));
5458      Modelica.Blocks.Interfaces.IntegerInput inOpen     annotation (Placement(
5459            transformation(extent={{-54,-72},{-14,-32}}),iconTransformation(
5460              extent={{-122,-46},{-100,-24}})));
5461      Modelica.Blocks.Interfaces.IntegerInput inCase1or2
5462                                                          annotation (Placement(
5463            transformation(extent={{-28,-92},{12,-52}}), iconTransformation(
5464              extent={{-122,-70},{-100,-48}})));
5465      Modelica.Blocks.Interfaces.IntegerOutput outAccumulatedLossesServiceC1
5466                                                                    annotation (
5467         Placement(transformation(
5468            extent={{-10,-10},{10,10}},
5469            rotation=-90,
5470            origin={34,76}), iconTransformation(
5471            extent={{-10,-10},{10,10}},
5472            rotation=90,
5473            origin={-50,70})));
5474      Modelica.Blocks.Interfaces.IntegerOutput outAccumulatedLossesServiceC2
5475                                                                    annotation (
5476         Placement(transformation(
5477            extent={{-10,-10},{10,10}},
5478            rotation=-90,
5479            origin={58,-64}),iconTransformation(
```

```
5480            extent={{-10,-10},{10,10}},
5481            rotation=90,
5482            origin={-50,70})));
5483    equation
5484      connect(client.singleOrderOut,reoCircuit7_1. portIn) annotation (Line(
5485          points={{-51.2,-0.28},{-36,0},{-21.48,0.7}},
5486          color={255,128,0},
5487          smooth=Smooth.None));
5488      connect(reoCircuit7_1.portOut1,m1_1. servicePortIn) annotation (Line(
5489          points={{8.76,5.8},{8.76,22},{20,22},{20,19.83},{25.44,19.83}},
5490          color={255,128,0},
5491          smooth=Smooth.None));
5492      connect(reoCircuit7_1.portOut2,m2_1. servicePortIn) annotation (Line(
5493          points={{8.4,-5.3},{8.4,-28},{20,-28},{20,-29.91},{25.72,-29.91}},
5494          color={255,128,0},
5495          smooth=Smooth.None));
5496      connect(client.outputCT, outputClient) annotation (Line(
5497          points={{-63.32,-10.22},{-63.32,-35.11},{-54,-35.11},{-54,-60}},
5498          color={255,127,0},
5499          smooth=Smooth.None));
5500      connect(client.singleOrderIDIn, inIdTable) annotation (Line(
5501          points={{-68.84,10.22},{-68.84,35.11},{-82,35.11},{-82,60}},
5502          color={255,127,0},
5503          smooth=Smooth.None));
5504      connect(inPriceTable, client.singleOrderPriceIn1) annotation (Line(
5505          points={{-50,68},{-54,68},{-54,10.22},{-57.56,10.22}},
5506          color={255,127,0},
5507          smooth=Smooth.None));
5508      connect(m2_1.outAccumulatedLosses, outAccumulatedLossesC2) annotation (Line(
5509          points={{30.9,-18.73},{30.9,-37.365},{32,-37.365},{32,-64}},
5510          color={255,127,0},
5511          smooth=Smooth.None));
5512      connect(m1_1.outAccumulatedLosses, outAccumulatedLossesC1) annotation (Line(
5513          points={{30.9,31.01},{30.9,43.505},{16,43.505},{16,78}},
5514          color={255,127,0},
5515          smooth=Smooth.None));
5516      connect(m1_1.outSendMailService, outSendMailServiceC1) annotation (Line(
5517          points={{49.1,31.01},{49.1,45.505},{68,45.505},{68,52}},
5518          color={255,127,0},
5519          smooth=Smooth.None));
5520      connect(m2_1.outSendMailService, outSendMailServiceC2) annotation (Line(
5521          points={{49.1,-18.73},{62.27,-18.73},{62.27,-32},{74,-32}},
5522          color={255,127,0},
5523          smooth=Smooth.None));
5524      connect(m1_1.outMaxFIFOCapacity, outMaxtFIFOCapacityC1) annotation (Line(
5525          points={{43.22,31.01},{43.22,51.505},{62,51.505},{62,74}},
5526          color={0,0,127},
5527          smooth=Smooth.None));
5528      connect(m2_1.outMaxFIFOCapacity, outMaxFIFOCapacityC2) annotation (Line(
5529          points={{43.22,-18.73},{49.07,-18.73},{49.07,-2},{68,-2}},
5530          color={0,0,127},
5531          smooth=Smooth.None));
5532      connect(reoCircuit7_1.currentFIFOCapacity, currentFIFOCapacityCircuit7)
5533        annotation (Line(
5534          points={{-5.64,10.9},{-5.64,21.45},{-2,21.45},{-2,34}},
5535          color={0,0,127},
5536          smooth=Smooth.None));
```

```
5537      connect(inOpen, reoCircuit7_1.open) annotation (Line(
5538          points={{-34,-52},{-34,-32},{-8.88,-32},{-8.88,-10.1}},
5539          color={255,127,0},
5540          smooth=Smooth.None));
5541      connect(inCase1or2, reoCircuit7_1.case1or2) annotation (Line(
5542          points={{-8,-72},{-8,-42},{-1.68,-42},{-1.68,-10.1}},
5543          color={255,127,0},
5544          smooth=Smooth.None));
5545      connect(m2_1.outAccumulatedLossesServices, outAccumulatedLossesServiceC2)
5546        annotation (Line(
5547          points={{37.06,-18.73},{37.06,-39.365},{58,-39.365},{58,-64}},
5548          color={255,127,0},
5549          smooth=Smooth.None));
5550      connect(m1_1.outAccumulatedLossesServices, outAccumulatedLossesServiceC1)
5551        annotation (Line(
5552          points={{36.78,31.01},{36.78,56.505},{34,56.505},{34,76}},
5553          color={255,127,0},
5554          smooth=Smooth.None));
5555      annotation (Diagram(graphics), Icon(graphics={
5556                                      Rectangle(
5557              extent={{-100,60},{100,-80}},
5558              lineColor={0,0,0},
5559              lineThickness=1), Text(
5560              extent={{-102,4},{102,-22}},
5561              lineColor={0,0,0},
5562              lineThickness=1,
5563              fillPattern=FillPattern.Solid,
5564              textString="C3 Simulink")}));
5565    end C3Simulink;
5566    annotation (uses(Modelica(version="3.2"), Modelon(version="1.6")),
5567      version="1",
5568      conversion(noneFromVersion=""));
5569  end RiskAnalysis;
```