KLOCwork MSC to SDL Synthesizer Tutorial

Version 1.0

Printed on 18 February, 2002

KLOCwork Corporation
Toll-free telephone: 1-866-556-2967
E-mail:
sales@klocwork.com
support@klocwork.com
Website: http://www.klocwork.com

Offices:
1 Antares Drive, Suite 510
Nepean, Ontario
Canada
K2E 8C4
(613) 224-2277

1700 Montgomery Street
Suite 111
San Francisco, CA
USA
94111
(415) 954-7154

Contents

CHAPTER 1

Introduction

In This Chapter

About this tutorial

This tutorial on the KLOCwork MSC to SDL Synthesizer contains a selection of best practices and examples intended to help you understand the key concepts related to the KLOCwork MSC for SDL Synthesizer. The KLOCwork MSC to SDL Synthesizer analyzes scenario models specified as a set of Message Sequence Charts (MSC) and High-level Message Sequence Charts (HMSC) and automatically synthesizes executable state machine specifications in Specification and Description Language (SDL).

This document contains the following parts:

- Introduction. The rest of this section describes several ways in which the KLOCwork MSC to SDL Synthesizer can bring value to your organization.
- Using the MSC to SDL Synthesizer in stand-alone mode
- Using the MSC to SDL Synthesizer with Telelogic Tau

Please refer to the *KLOCwork MSC to SDL Synthesizer Reference Manual* for systematic coverage of the installation, operations, and language-related issues of the tool. Please refer to the *Cookbook of MSC Specifications* for comprehensive examples of formal modeling with scenarios.

The KLOCwork MSC to SDL Synthesizer is a versatile tool. The following paragraphs illustrate how it can add value in the software development process.

Use MSCs to jumpstart your SDL project

The KLOCwork MSC to SDL Synthesizer can be used to jumpstart your SDL project. Start with few easy-to-understand scenarios, then simply click a button and the Synthesizer automatically generates a complete SDL specification ready to run in an SDL tool.

1. Start Telelogic Tau.
2. Use your Telelogic MSC Editor (MSCE) to create MSC diagrams.
3. Use the Telelogic HMSC Editor (HMSCE) to create High-level MSC diagrams.
4. Run the KLOCwork MSC to SDL Synthesizer to produce an SDL model that corresponds to your HMSC model.
5. Use your SDL tool to review and simulate the model.
6. Refine the MSC model.
7. Continue SDL project by refining the generated SDL model.

Use MSCs to rapidly prototype requirements

Jumping into coding too early and not understanding the requirements can result in developing a product that does not meet customer expectations. By using scenarios, you can describe functional requirements in an easily understood format that can be discussed with non-technical stakeholders.

1. Use Telelogic MSC and HMSC Editors to capture functional requirements as scenarios that focus on the interaction between your system and its environment.
2. Simulate MSCs by using the KLOCwork MSC to SDL Synthesizer and Telelogic Simulator UI.
3. When you notice problem areas, go back to your MSC tool, add more scenarios to solve the problems. Rerun the Synthesizer to get the new SDL specification.
4. Repeat this process until you are satisfied with your prototype (scenarios and SDL model).

Use MSCs to perform automatic early fault detection

Even when the goals of your project do not include building an SDL model, the KLOCwork MSC to SDL Synthesizer can be used for early fault detection in you MSC models. It can dramatically extend the value of your MSC tools by adding the capability of automatic validation techniques, for example, those provided by the Telelogic Tau Validator.

The automatically synthesized SDL model can be imported into an SDL tool and analyzed using state-of-the-art state space exploration. The validation process automatically discovers certain faulty behaviors such as deadlocks or failed communication. These faulty behaviors are represented as scenarios using the MSC tool. Anomalies in the behavior of the synthesized model usually reflect problems or inconsistencies in the original MSC model.

1. Use your Telelogic MSC Editor (MSCE) to create MSC diagrams.
2. Use the Telelogic HMSC Editor (HMSCE) to create High-level MSC diagrams.
3. Run the KLOCwork MSC to SDL Synthesizer to produce an SDL model that corresponds to your HMSC model.
4. Use the Telelogic Tau Validator to automatically detect faults in the scenario model.
5. Correct problems by creating additional scenarios.
6. Rerun the KLOCwork MSC to SDL Synthesizer.

Use MSCs to automatically generate test cases

The KLOCwork MSC to SDL Synthesizer can automatically produce test cases as early as the requirements definition phase.

During scenario modeling, the black-box behavior of the system under development is described by specifying desired interactions between the system and its environment. Thus, the scenario model contains both the description of the system and the description of the system's environment.

The KLOCwork MSC to SDL Synthesizer can selectively produce a partial SDL model only for the environment part of the requirements definition. This environment model can be used as a test suite for the subsequent integration testing phase.

Use MSCs for architecture definition and validation

Automatic synthesis can help address the discontinuity between modeling and implementation. By specifying system scenarios that describe collaboration between architectural components and then synthesizing the SDL architectural model, you can achieve the following benefits:

- early architecture validation by simulating system scenarios
- concurrent production of system scenarios for different components for integration by the KLOCwork MSC to SDL Synthesizer

Use MSCs to design components

Once the components of the system have been identified and validated, you can shift the viewpoint of scenario models and use scenarios to define behavior of individual components. Using the powerful data extensions supported by the KLOCwork MSC to SDL Synthesizer, you can use scenarios as a design notation. This provides the following benefits:

- more intuitive designs
- improved collaboration between team members
- ongoing validation of designs by simulation of the synthesized models
- automatic code generation into the implementation language using SDL Codegenerator

Use KLOCwork MSC to SDL Synthesizer for SDL training

SDL is a complex language. Most people find it easier to understand scenarios than SDL. You can explain a scenario to a new employee then import the scenario into the KLOCwork MSC to SDL Synthesizer to automatically create the corresponding SDL model. Your new employee can learn SDL faster by observing the automatically synthesized SDL for an already familiar scenario.

CHAPTER 2

Using the KLOCwork MSC to SDL Synthesizer in stand-alone mode

In This Chapter

Introduction to the simple dialog exercise

This exercise contains an example of how to create a textual scenario model in MSC PR (Phrase Representation) and how to use the command-line interface of the KLOCwork MSC to SDL Synthesizer to produce an SDL model.

Setup

To perform all steps of this example, you need to have installed:

- the KLOCwork MSC to SDL Synthesizer
- a text editor (to create MSC PR files)

All source files for this example can be found in the following directory:

```
<KLOCwork MSC to SDL Synthesizer installation
directory> /doc/examples/Simple
```

Model: Simple dialog

This example describes a simple dialog between two parties (A and S). The parties communicate by passing messages. First, party A sends the message *Question* to S. Then, S replies to A with the message *Answer*. The desired behavior of this model is an infinite repetition of the message interchange between A and S.

In the following exercises we describe the steps for

- creating the text representation of the MSC specification of this model, and
- producing an executable state machine model using the command line interface of the KLOCwork MSC to SDL Synthesizer.

Create textual MSCs

In this exercise we create the text representation of the MSC specification for our example. The specification consists of two text files: an MSC describing the desired sequence of events for the example, and an HMSC describing the infinite repetition of this sequence.

1 Create a new directory, which will be used as a working directory for this example. This working directory must contain source files of the model. Enter the working directory.

2 Create a text file called *ask.mpr* in the working directory. It contains a single MSC Ask in the *Phrase Representation* (MSC PR) format. The file should be as follows:

File *ask.mpr*

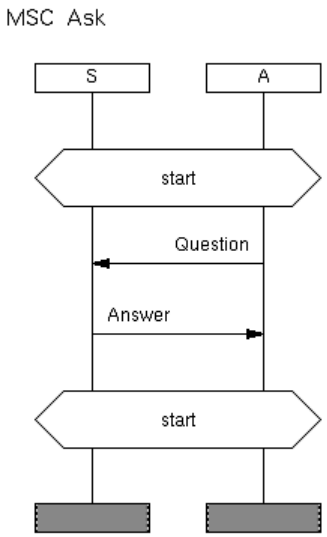
```
mscdocument Ask;
msc Ask;
S: instance;
A: instance;
all: condition start;
A: out Question to S;
S: in Question from A;
S: out Answer to A;
A: in Answer from S;
all: condition start;
S: endinstance;
```

A: **endinstance**;

endmsc;

The corresponding Graphical Representation (MSC GR format) looks like this:

Figure 1: MSC Ask in phrase and graphical representation



The MSC Ask specifies two instances (A and S). This MSC has two shared conditions with the same name *start* at the beginning and at the end of the sequence of events. Between the two conditions there is an interchange with messages between the two instances.

3 Create a text file called *dialog.mpr* in the working directory. It contains a single HMSC Dialog in the Phrase Representation (MSC PR) format.

File *dialog.mpr*

mscdocument Dialog;

msc Dialog;

expr L1;

L1: **condition** start **seq** (L2);

L2: (Ask) **seq** (L1);

endmsc;

4 The corresponding Graphical Representation (MSC GR format) looks like this:

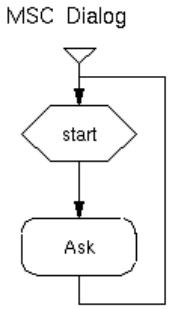


Figure 2: Model roadmap as HMSC (phrase and graphical representations)

This HMSC contains a start symbol at the top followed by symbol of condition *start* followed by an MSC reference symbol *Ask* and a flowline back to the start condition. Condition *start* in HMSC Dialog stands for both initial and final conditions in MSC *Ask.* This HMSC describes an infinite loop in which events from the MSC *Ask* are performed.

Summary

We have completed the textual specification of our model.

The next exercise will describe how to use the command-line interface to the KLOCwork MSC to SDL Synthesizer to produce an executable state machine model for our model.

Run the KLOCwork MSC to SDL Synthesizer

In this exercise, we run the KLOCwork MSC to SDL Synthesizer to produce a text file containing the SDL model for our example.

1 Make sure that MSC to SDL is in the PATH environment variable (the work directory contains the files *ask.mpr* and *dialog.mpr*) and perform the following command in your command shell.

```
plainsdl ask.mpr dialog.mpr
```

The KLOCwork MSC to SDL Synthesizer analyzes the model described in the files *dialog.mpr* and *ask.mpr*. The synthesizer analyzes the syntax and semantics of the MSC model. All MSCs referenced in the HMSC must be present in MSC PR files in the list of files in the command line. The synthesized SDL model is written to a text file called *System.sdl* (the default filename). File *System.sdl* contains a synthesized SDL model in the phrase representation (SDL-PR). For this example we used the *plain SDL* code generator.

Summary

We have produced the text specification of the SDL model.

In the next exercise we will inspect the synthesized model.

Inspect the synthesized SDL model

In this exercise, we inspect the text SDL file produced by the KLOCwork MSC to SDL Synthesizer.

1 Open the file *System.sdl* in your text editor. The contents of this file in phrase representation format are as follows:

```
system SynthesizedModel;
signal ANSWER;
signal QUESTION;
block SynthesizedModel_block;
signalroute R1
    from S to A with ANSWER;
    from A to S with QUESTION;
process S referenced;
process A referenced;
endblock SynthesizedModel_block;
endsystem SynthesizedModel;
process S;
    signalset QUESTION;
    start;
         nextstate START_1;
    state START_1;
         input QUESTION;
         output ANSWER via R1;
         nextstate START_1;
         save *;
endprocess;
process A;
    signalset ANSWER;
    start;
         join START_1;
  state ASK_0;
         input ANSWER;
         join START_1;
```

```
            save *;
    connection START_1:
            output QUESTION via R1;
            nextstate ASK_0;
endprocess;
```

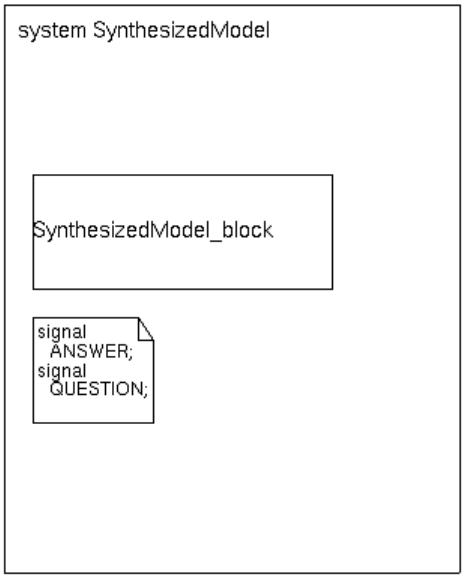
The synthesized SDL file is fully SDL-compliant. It contains a complete definition of the structure, behavior and data definitions required for the executable state machine specification corresponding to our example.

The corresponding graphical representations of the SDL model are shown and explained below.

Structure of the SDL specification

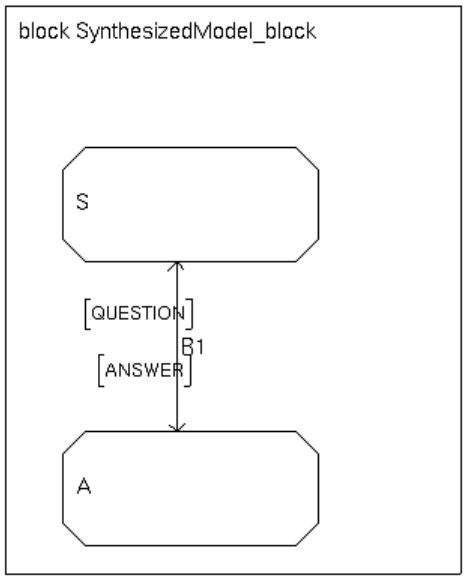
The structural part of the SDL specification consists of a hierarchy of a block and processes as well as some communication infrastructure. The synthesized SDL model contains the system *SynthesizedModel* that contains a block *SynthesizedModel_block* and some signal definitions. In our example, we have two signals: *question* and *answer* – named after the messages in MSC Ask.

Figure 3: SDL-GR for simple dialog (structure)



The block *SynthesizedModel_block* contains processes A and S that correspond to the instances in MSC Ask. The block also describes a signal route between these processes. It conveys signal *question* from process *A* to process *S* and signal *answer* in the reverse direction.

Figure 4: SDL-GR for simple dialog (structure) (2)

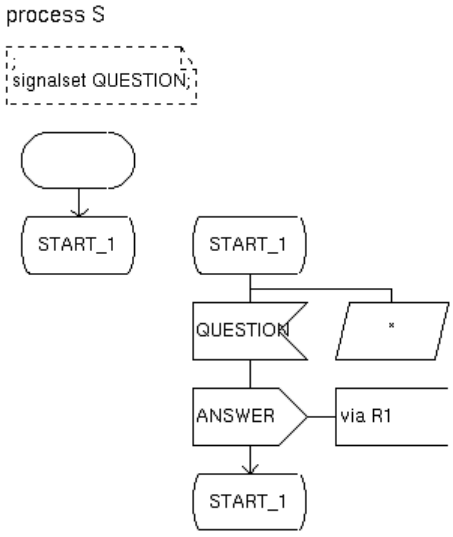


Behavior of the SDL specification

The behavioral part of the SDL specification contains state machine descriptions for processes A and S.

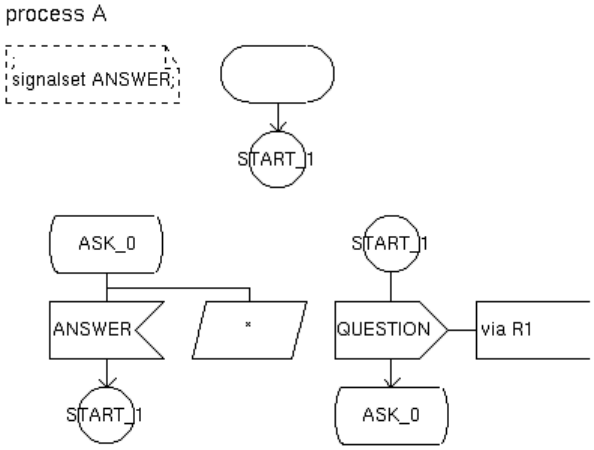
The state machine for the process *S* contains a single state *start_1*. The start transition of S enters this state. Process *S* waits in state *start_1* for an input signal *question.* When the signal *question* arrives, the process outputs the signal *answer* to process *A* and returns to the same state to wait for the next instance of the *question* signal.

Figure 5: SDL-GR for Simple dialog (behavior of S)



The state machine for process *A* contains a single state *ask_0* and a labeled transition *start_1.* The start transition is a join to the labeled transition *start_1* where the process outputs the signal *question* to the process *S* and then waits for the signal *answer* in the state *ask_0.* When the signal *answer* arrives, process *A* repeats all its actions by joining into transition *start_1.*

Figure 6: SDL-GR for Simple dialog (behavior of A)



All states contain the *save* * construct that allows processes to save all signals except those that are handled in this state.

The behavior of both processes corresponds to the behavior of the instances described in the input scenario model.

CHAPTER 3

Using the KLOCwork MSC to SDL Synthesizer with Telelogic Tau

In This Chapter

Introduction to the information server exercise

In this section we demonstrate how to use the KLOCwork MSC to SDL Synthesizer in combination with Telelogic Tau. Specifically, we demonstrate how to

- create graphical MSC diagrams using Telelogic Tau MSC and HMSC editors
- use the KLOCwork MSC to SDL Synthesizer to simulate MSCs
- seamlessly generate graphical SDL models using MSC to SDL Synthesizer and view them in the Telelogic Tau SDL Editor
- perform early fault detection using the KLOCwork Synthesizer and the Telelogic Tau Validator

Setup

To perform all steps of this example, you need to have installed:

- Telelogic Tau
- Telelogic Tau C code generator and a C environment.

The source files for this part of the tutorial can be found in the following directory:

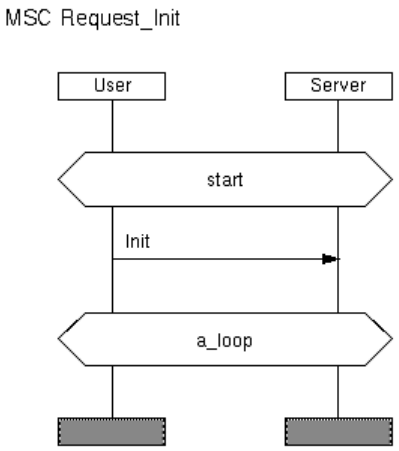
```
<KLOCwork MSC to SDL Synthesizer
installation>/doc/examples/InfoServer
```

Model: Information Server

The example used in this part of the tutorial describes a simple *information server*. The information server provides certain data to users. The information server is the system under development. The name of the system instance is *Server*. An instance with name *User* represents a user that receives the data from the server.

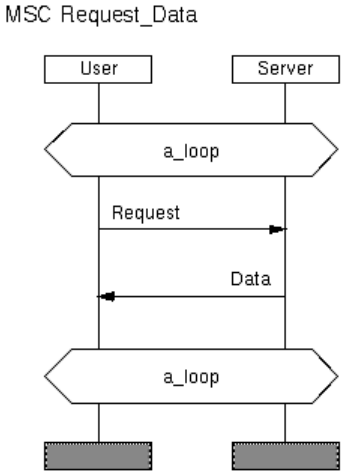
The user starts interaction by sending message *Init* to the server. Then, the user requests data units by sending the message *Request*. Finally, the user terminates the interaction by sending the message *Done*. This behavior is specified by the MSC scenarios *Request_Init*, *Request_Data* and *Request_Done*. HMSC *InfoServer* provides the roadmap of the three scenarios.

Figure 7: MSC diagram Request_Init



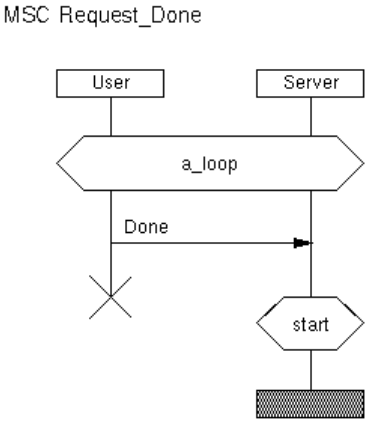
MSC Request_Init describes the initialization of the information server. The User sends message *Init* to the Server and they both enter a data exchange loop (condition *a_loop*).

Figure 8: MSC diagram Request_Data



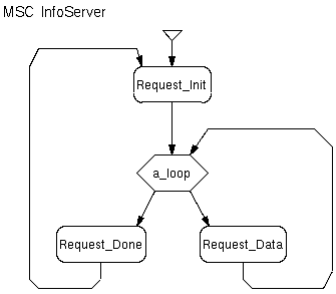
MSC Request_Data describes how the User sends message request to the Server and receives some data from Server (message Data). After this, both the Server and the User return to the data exchange loop (condition a_loop).

Figure 9: MSC diagram Request_Done



MSC Request_Done describes how the User sends message *Done* to the Server and breaks the connection between the User and the Server. The User terminates its involvement in the scenario (stop event at the end of the User instance at this MSC diagram). The Server continues its work.

Figure 10: HMSC InfoServer



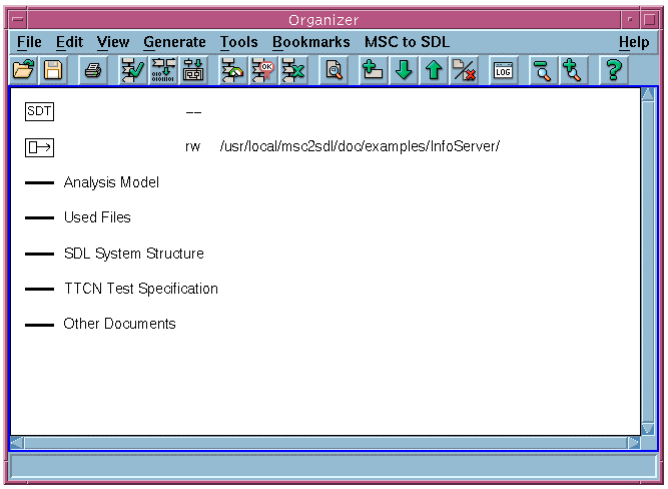
HMSC InfoServer describes the roadmap for the three MSC scenarios.

Create MSC and HMSC diagrams in Telelogic Tau

In this exercise, we use Telelogic Tau graphical editors to create our scenario model.

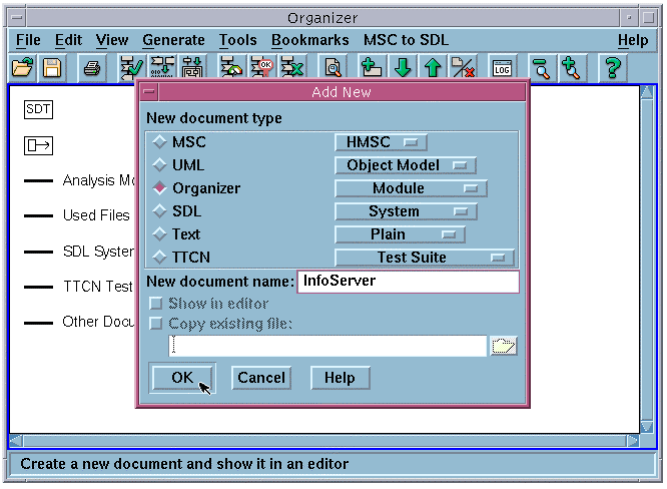
1 Start Telelogic Tau with the KLOCwork MSC to SDL Synthesizer (you may need to consult Telelogic Tau documentation and the KLOCwork MSC to SDL Synthesizer Reference Manual). You see the Telelogic Tau Organizer with the MSC to SDL menu.

Figure 11: Telelogic Tau Organizer



2 Create an Organizer *module,* which serves as the folder for all diagrams in this example. To do this, from the Edit menu, select Select Add New, then select the Organizer radio button and make sure that the corresponding pull-down menu reads *Module*. Type the name *InfoServer* into *New document name* field. The icon for the new module appears in the Organizer view.

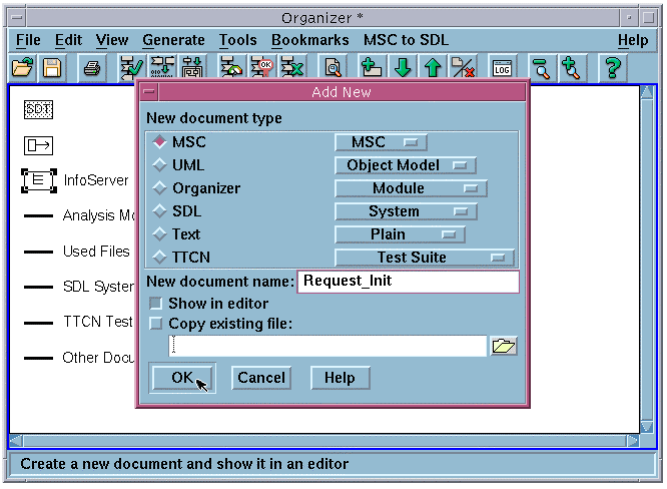
Figure 12: Creating a new Organizer module



3 On the Add New dialog, click the *Show in editor* checkbox to ensure that you will be able to see the icons for new MSC diagrams.

4 Create a new MSC diagram called Request_Init. To do this, from the Edit menu, select Add New, then select the MSC radio button. The corresponding pull-down menu should read MSC. Type the name *Request_Init* into *New document name* field. The icon for the new MSC diagram appears in the Organizer view and the Telelogic Tau MSC Editor opens. You can also start the MSC Editor by from the Organizer Tools menu (choose Editors, then MSC Editor).

Figure 13: Creating a new MSC diagram in Organizer



5 Create an HMSC diagram by adding a new diagram using the Organizer or the HMSC Editor:

From the Organizer Edit menu, select Add New, then check the MSC radio button and set the corresponding pull-down menu to HMSC.

or

Start the HMSC Editor (from the Tools menu, select Editors, then HMSC Editor).

6 Move the MSC diagram *Request_Init* into the Organizer module *InfoServer* by selecting the icon *Request_Init* and then clicking the *Move up* button in the quick button bar of the Organizer.

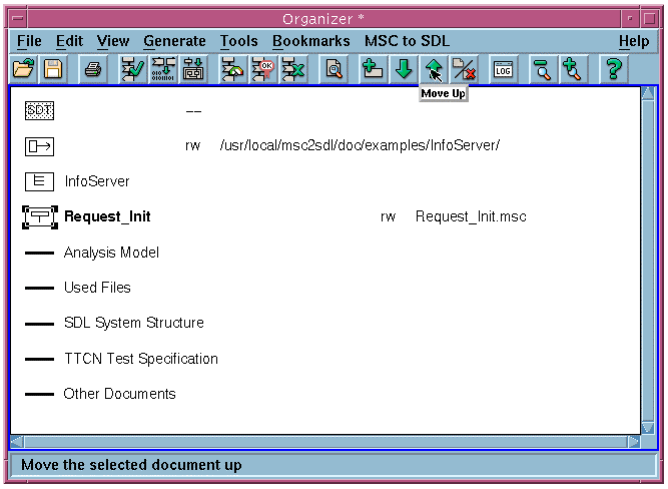


Figure 14: Adding MSC to the module

The following figure shows the Telelogic Tau MSC Editor after one instance has been created.

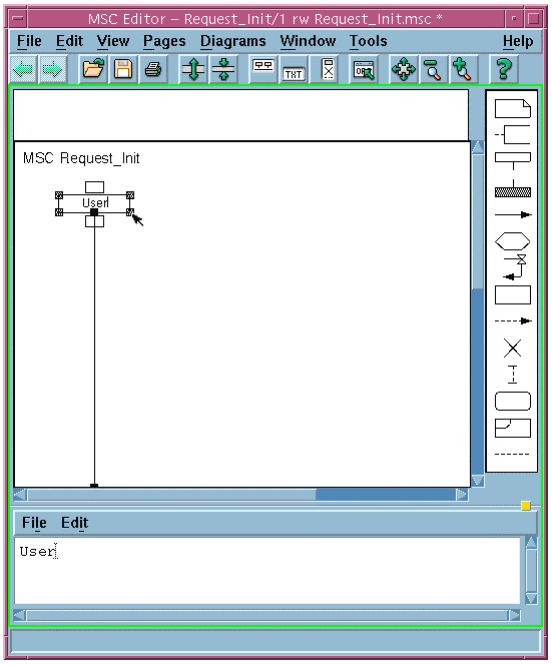
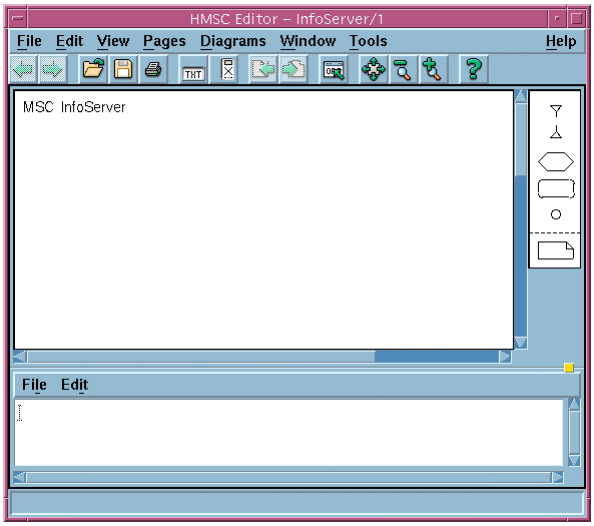


Figure 15: Telelogic Tau MSC Editor

The following figure shows an empty Telelogic HMSC Editor.

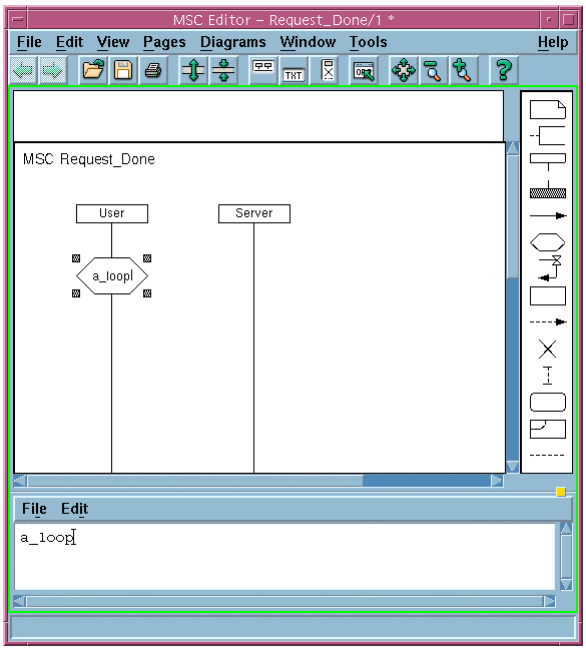
Figure 16: Telelogic Tau HMSC Editor



7 Add a new symbol in the MSC Editor using the *palette* of graphical symbols (right panel). Click on a symbol in the palette, then point at the graphical area to insert selected symbol.

The following figure shows MSC Request_Done after we added two instances and a condition symbol a_loop. A new condition symbol is always added as a local condition (attached to a single instance).

Figure 17: Creating a condition symbol in the MSC Editor



8 To create a global condition symbol you need to select the condition symbol in the diagram, then open the Connect dialog (Edit menu, select Connect) and click on the Global button. You can also use this menu to fine tune which instances share this condition. Clicking on an instance in the Instance List on the Connect dialog toggles sharing. After toggling the desired instances, click Connect.

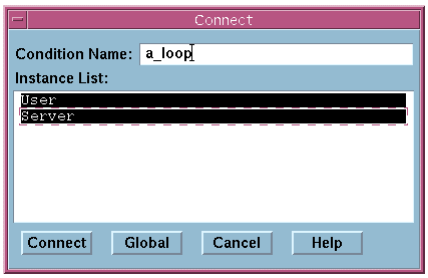


Figure 18: Connect dialog (Edit menu) in MSC Editor

You may need to consult Telelogic Tau documentation for more information about using the graphical editors.

9 Create MSC diagrams with the names Request_Init, Request_Data, and Request_Done and HMSC diagram InfoServer. See ***Model: Information Server*** (on page 19).

10 Save your work by clicking the Save button in the Organizer. You are prompted to save each diagram and the so-called *system file.* Save all files in your working directory.

Summary

In this exercise we created the Organizer module containing the three MSC diagrams (Request_Init, Request_Data and Request_Done) and an HMSC diagram InfoServer. You used the Telelogic Tau Organizer, MSC Editor and HMSC Editor. The Organizer window should look similar to this one:

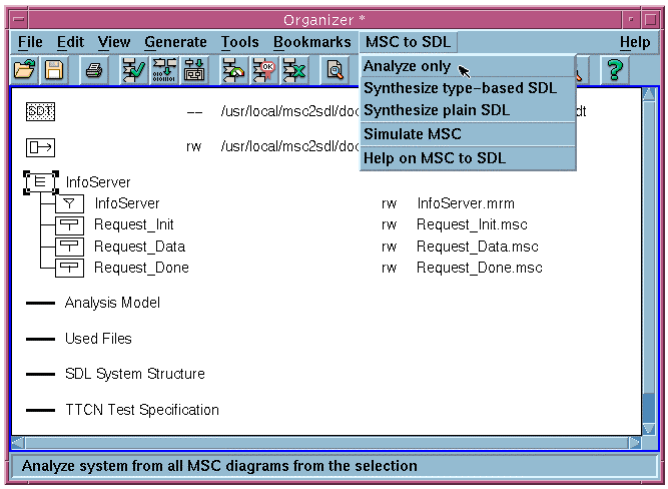


Figure 19: MSC to SDL menu, Analyze only command

Summary

In this exercise we demonstrated how to use the Telelogic Tau Organizer and graphical editors to create the MSC model.

In the next exercise we will use the KLOCwork MSC Analyzer to check the correctness of MSC models.

Analyze MSCs

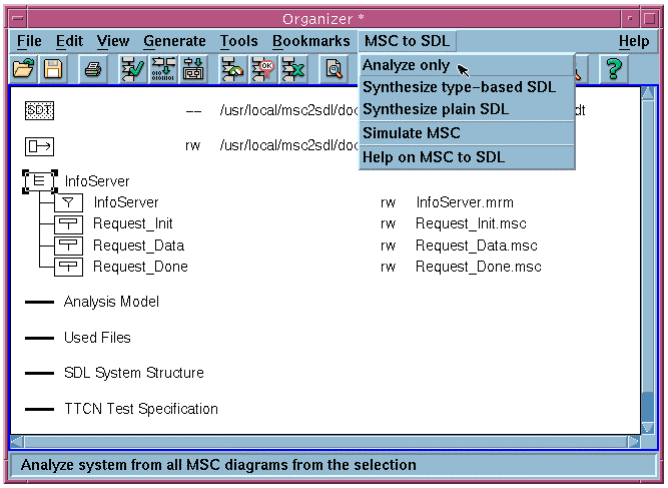
In this exercise we check correctness of our scenario model using the KLOCwork MSC Analyzer.

Setup

Make sure that the Telelogic Tau Organizer contains the module *InfoServer* with the four diagrams for the current example (InfoServer, Request_Init, Request_Data, and Request_Done).

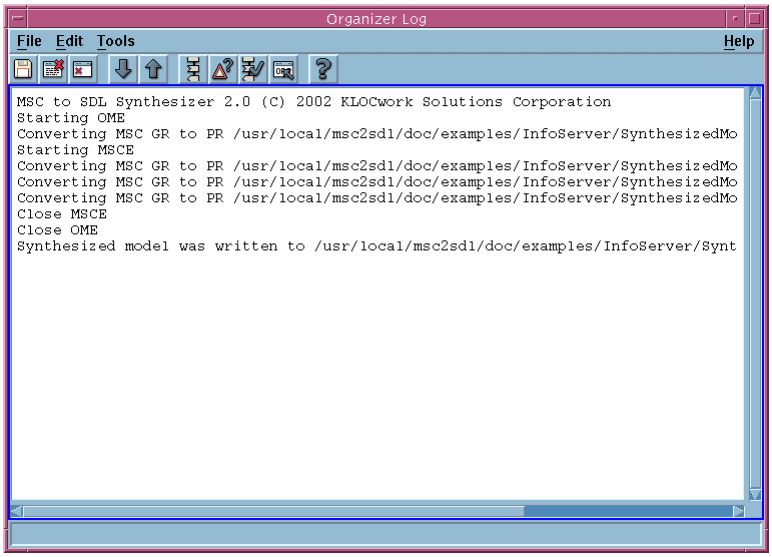
1 Click on the *InfoServer* module to select it. From the MSC to SDL menu, choose Analyze only.

Figure 20: MSC to SDL menu, Analyze only command



The KLOCwork MSC to SDL Synthesizer automatically converts MSC and HMSC diagrams from graphical representation to phrase representation and runs the KLOCwork MSC Analyzer. The steps are reported in the Organizer Log.

Figure 21: Organizer Log for analyzing MSCs



Any errors found in the input HMSC model are reported in the Organizer Log. You can navigate to the source of an error in the graphical representation using one of the Telelogic Tau editors (HMSC Editor or MSC Editor).

2 Let's introduce an intentional error into one of the diagrams using a graphical editor and explore the error reporting capability of the KLOCwork MSC Analyzer. Open the HMSC InfoServer by double-clicking on the corresponding icon in the Organizer. Select the Request_Init symbol and change the name of the referenced MSC to Request_Initxxx. Save the HMSC InfoServer by clicking the Save button in the quick button bar.

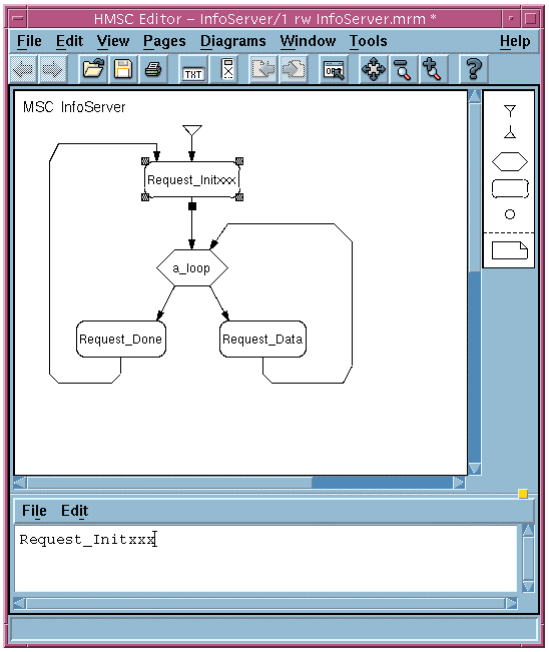


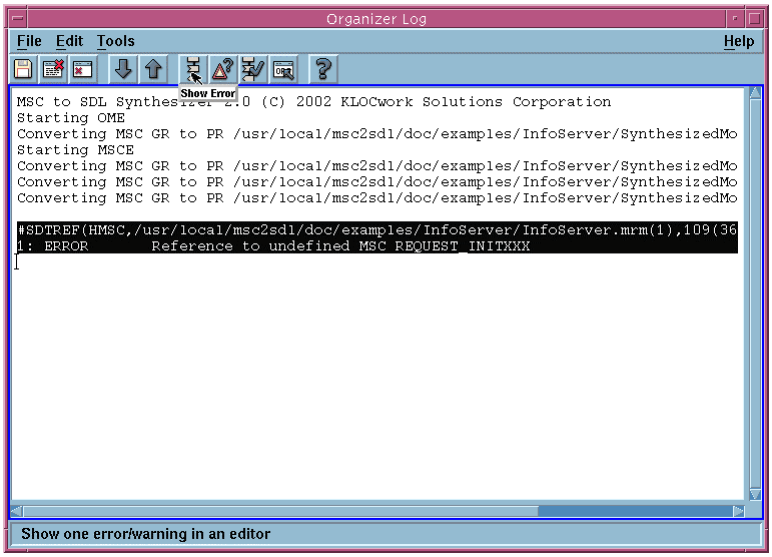
Figure 22: Introducing an intentional error into HMSC InfoServer

3 Select the icon for the InfoServer module in the Organizer and run the KLOCwork MSC Analyzer. The Organizer Log now contains the error message:

```
1:  ERROR Reference to undefined MSC
REQUEST_INITXXX
```

4 Select the error message and click the Show Error button. The HMSC Editor opens and highlights the corresponding symbol on HMSC InfoServer.

Figure 23: Navigating to the source of the error for the Organizer Log



The complete list of errors reported by the KLOCwork MSC Analyzer can be found in the KLOCwork MSC to SDL Synthesizer Reference Manual.

Summary

In this exercise we demonstrated the capabilities of the KLOCwork MSC Analyzer.

In the next exercise we will use the KLOCwork MSC to SDL Synthesizer to explore the behavior of our model by simulating MSCs.

Simulate MSCs

In this four-part exercise we explore the behavior of our model by simulating MSCs using the KLOCwork MSC to SDL Synthesizer.

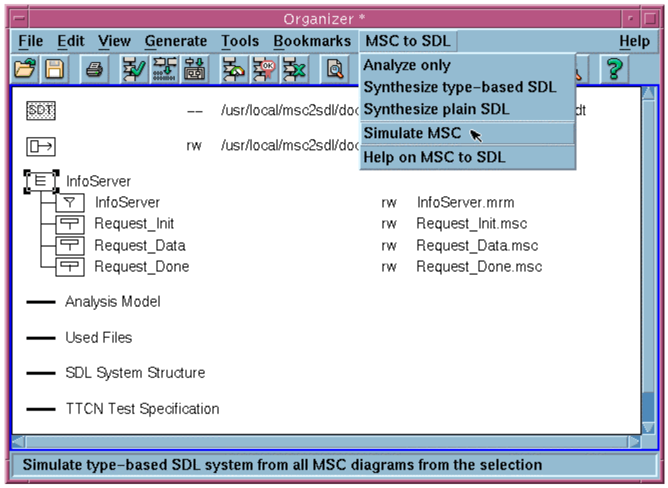
Setup

Make sure that the Telelogic Tau Organizer contains the module *InfoServer* with the four diagrams for the current example (InfoServer, Request_Init, Request_Data, and Request_Done).

Part 1: Starting MSC Simulation

1 From the Organizer MSC to SDL menu, choose Simulate MSC.

Figure 24: MSC to SDL menu, Simulate MSC command

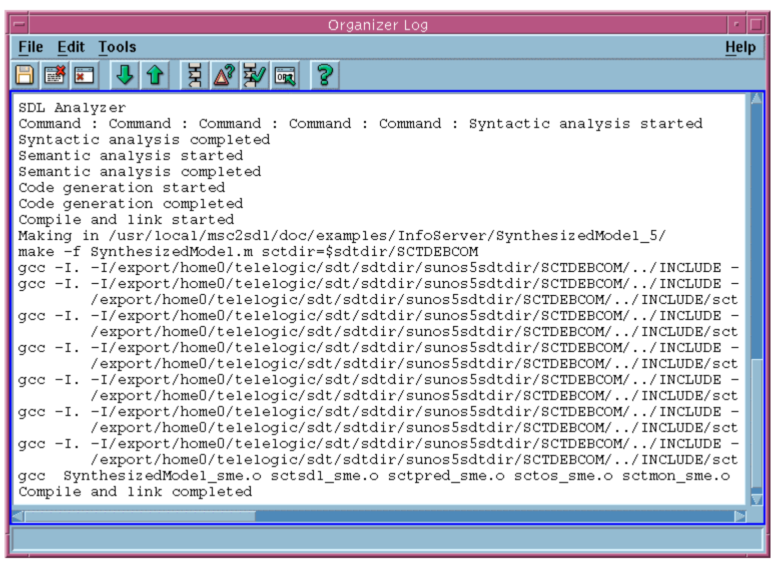


The Simulate MSC menu item starts the KLOCwork MSC to SDL Synthesizer in MSC Simulation mode. The KLOCwork MSC to SDL Synthesizer performs the following steps:

- Convert all diagrams from graphical to phrase representation
- Analyze the HMSC model
- Produce the executable SDL model
- Call Tau code generator to produce the corresponding C code
- Execute the generated makefile to compile and link the executable model
- Start the Tau Simulator UI

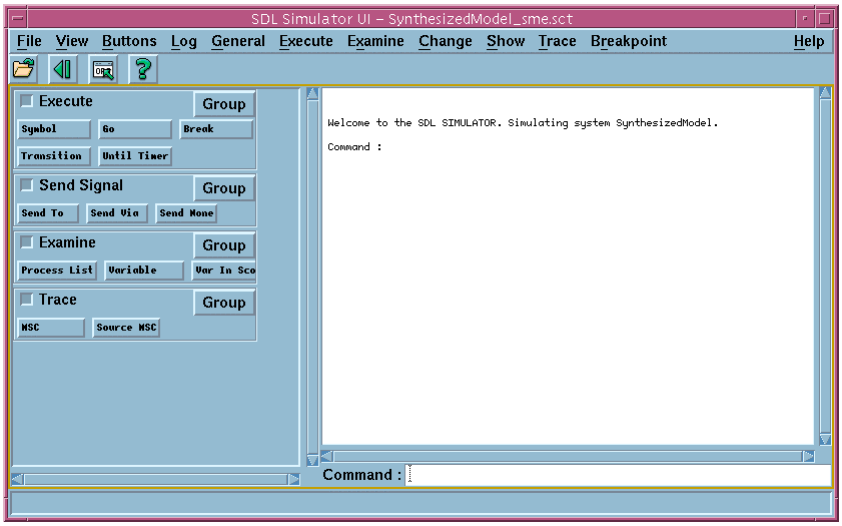
All steps are reported in the Organizer Log.

Figure 25: Organizer Log for Simulating MSC



The Tau Simulator UI window appears.

Figure 26: Telelogic Tau Simiulator UI



Summary

We have started the Telelogic Tau UI and are ready to continue with the exercise.

Part 2: Controlling the Telelogic Tau Simulator UI

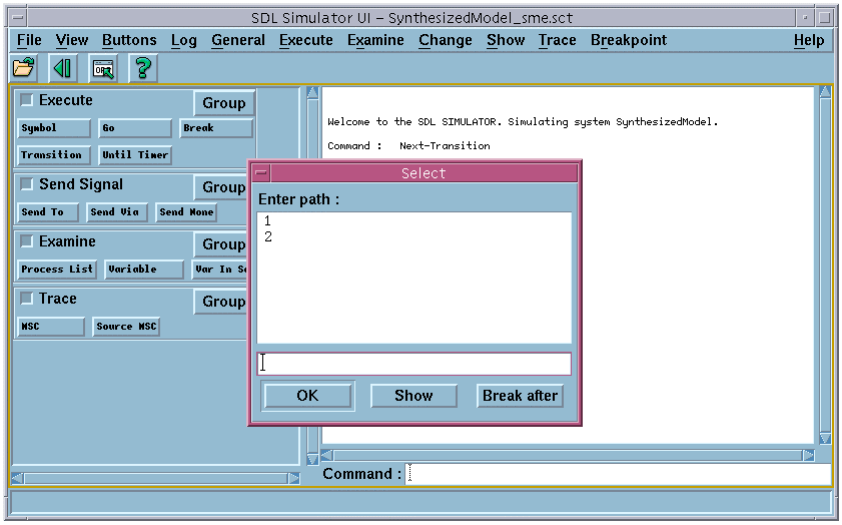
In this step we use the Tau Simulator UI to explore the behavior of our MSC model. We demonstrate how to control the Tau Simulator UI and observe the textual execution trace that corresponds to the input MSC model.

The Tau Simulator UI window has a *buttons panel* (left), a *text output* panel (right) and a *command entry* field (bottom). The text output panel displays the *execution trace*. The Tau Simulator is an interface to the automatically synthesized executable model produced by the KLOCwork MSC to SDL Synthesizer.

The model consists of several concurrent *processes* (one for each instance in the input MSC model). The processes exchange *signals,* in accordance with the input MSC. Each process in the synthesized model performs *transitions* between internal *states*.

1. Click the Transition button in the buttons panel to perform a single transition of the model. The activities of the model are reported in the text output panel.
2. Click the Transition button again. The Simulator UI opens the Path Selection dialog.

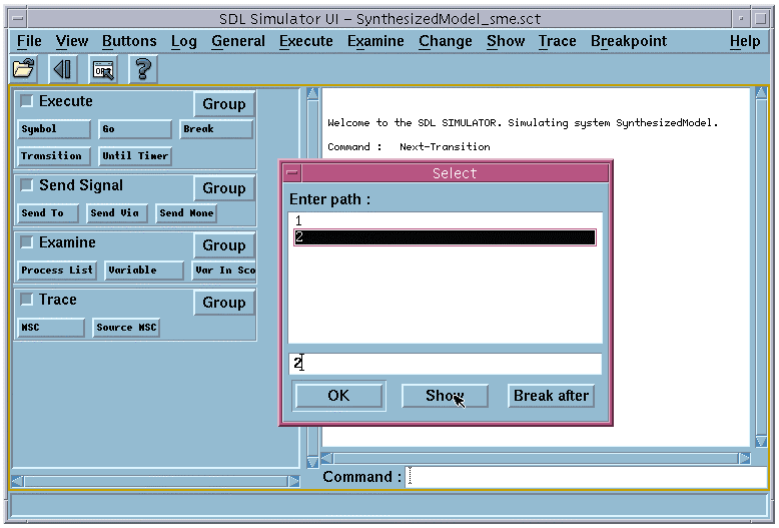
Figure 27: Path selection dialog



The input MSC model is *non-deterministic* because it does not provide enough information about the logic behind the actions of the User. According to the ***model*** (see "Model: Information Server" on page 19), the User can either send the Request message or the Done message to the Server. The model does not specify how the User makes the choice between these two messages. The KLOCwork MSC to SDL Synthesizer propagates this non-determinism to the executable model. The Telelogic Tau Simulator uses the Path Selection dialog to ask the user which of the alternatives of a non-deterministic choice to take.

3 Explore various alternatives of the non-deterministic choice by selecting a number in the Path Selection dialog and clicking on the Show button.

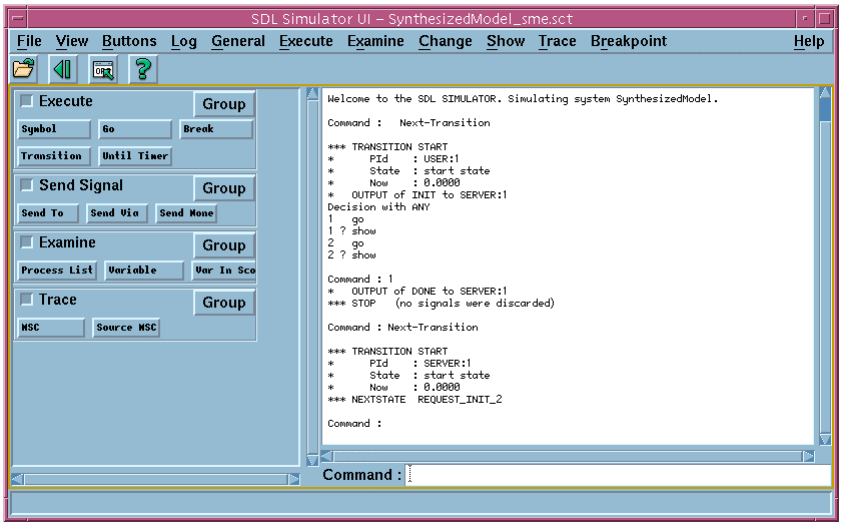
Figure 28: Explore alternatives of the non-deterministic choice



The Tau MSC Editor appears and the corresponding message is highlighted.

4 Select the first alternative by double clicking on the line with number 1 in the Path Selection dialog. This corresponds to sending the message Done from the User to the Server. The Simulator Trace is displayed in the text output panel.

Figure 29: Simulator trace



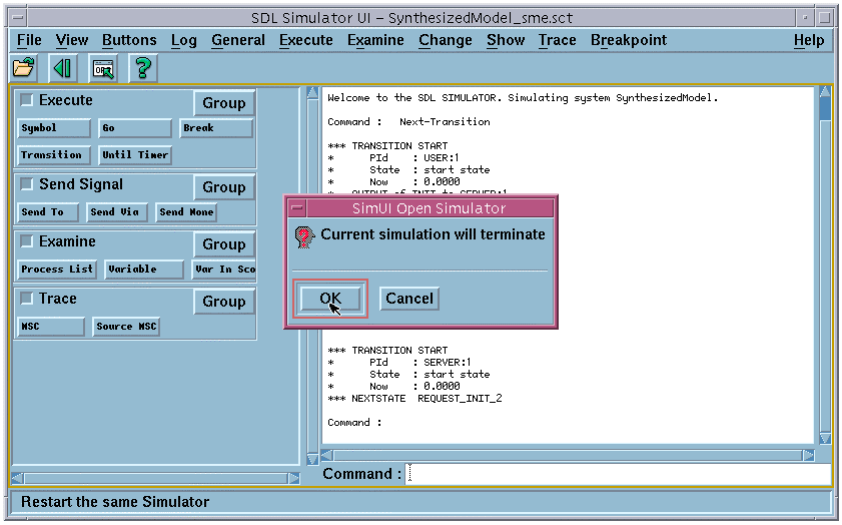
5 Click the Transition button until the Simulation trace displays the line "`No process instance scheduled for a transition`". This complete Simulator trace is presented below. The trace consists of 4 transitions (from the line "`*** TRANSITION START`" to the line "`*** NEXTSTATE `"). Transitions are performed by different instances of the MSC model (see the name of the instance in the second line in each transition section after the keyword *PId)*. The transition section also contains the input signal, or the transition trigger (see the name of the input signal in the fourth line of transition section after the keyword *Input)*.

```
Welcome to the SDL SIMULATOR. Simulating
system SynthesizedModel.

Command :   Next-Transition

*** TRANSITION START

*       PId     : USER:1

*       State   : start state

*       Now     : 0.0000

*   OUTPUT of INIT to SERVER:1

Decision with ANY

1   go

1 ? show

2   go

2 ? show
```

```
Command : 1
*   OUTPUT of DONE to SERVER:1
*** STOP   (no signals were discarded)
Command : Next-Transition
*** TRANSITION START
*      PId    : SERVER:1
*      State  : start state
*      Now    : 0.0000
*** NEXTSTATE  REQUEST_INIT_2
Command : Next-Transition
*** TRANSITION START
*      PId    : SERVER:1
*      State  : REQUEST_INIT_2
*      Input  : INIT
*      Sender : USER:1+
*      Now    : 0.0000
*** NEXTSTATE  A_LOOP_1
Command : Next-Transition
*** TRANSITION START
*      PId    : SERVER:1
*      State  : A_LOOP_1
*      Input  : DONE
*      Sender : USER:1+
*      Now    : 0.0000
*** NEXTSTATE  REQUEST_INIT_2
Command : Next-Transition
No process instance scheduled for a transition
```

6 To restart the simulation, from the File menu, choose Restart.

Figure 30: Restarting the current simulation



Summary

In this part of the exercise we demonstrated how to use the KLOCwork MSC to SDL Synthesizer to simulate MSCs. We explored the behavior of our MSC model using the textual trace of the Telelogic Tau Simulator UI.

In the next two parts of this exercise we will explore more powerful ways to visualize execution traces using the KLOCwork MSC to SDL Synthesizer and Telelogic Tau.

Part 3: Viewing Source MSC

The textual trace output demonstrated in the previous part of this exercise was not very intuitive. In this part of the exercise, we explore more advanced visualization capabilities provide by the KLOCwork MSC to SDL Synthesizer.

We use the source navigation capability of the KLOCwork MSC to SDL Synthesizer and Telelogic Tau to display all events from the execution trace in the graphical MSC Editor.

1. Restart the current simulation by clicking on the Source MSC button in the buttons panel of the Simulator UI.
2. Simulate the model by clicking the Transition button and selecting an appropriate path in the Path Selection dialog. The Tau MSC Editor appears showing one of the input MSC diagrams and highlighting the MSC event performed by the current transition. The text output panel of the Tau Simulator UI still displays the textual execution trace.

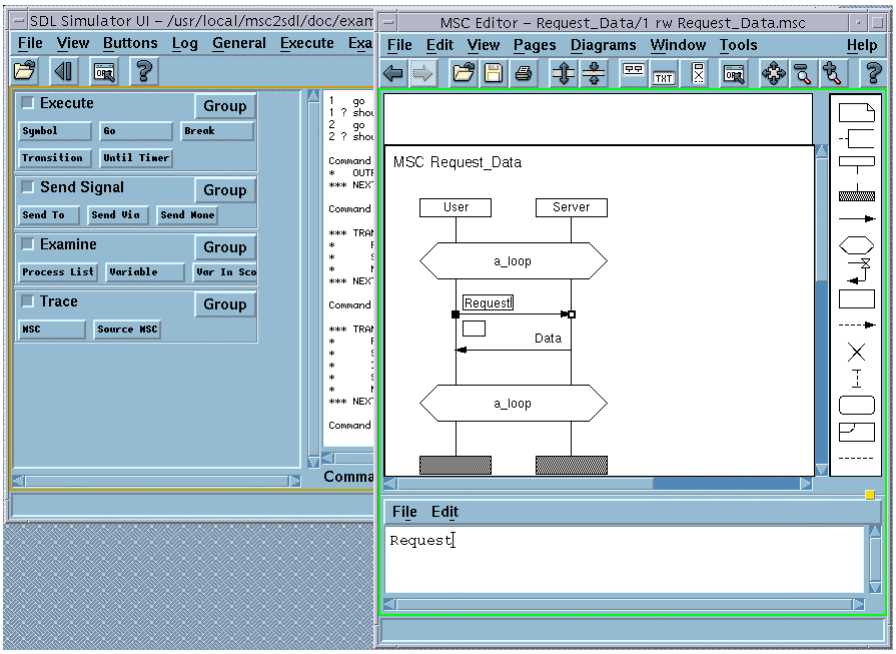


Figure 31: Viewing MSC Source during simulation

Summary

In this part of the exercise, we used the source navigation capability of the KLOCwork MSC to SDL Synthesizer and Telelogic Tau to display all events from the execution trace in the graphical MSC Editor.

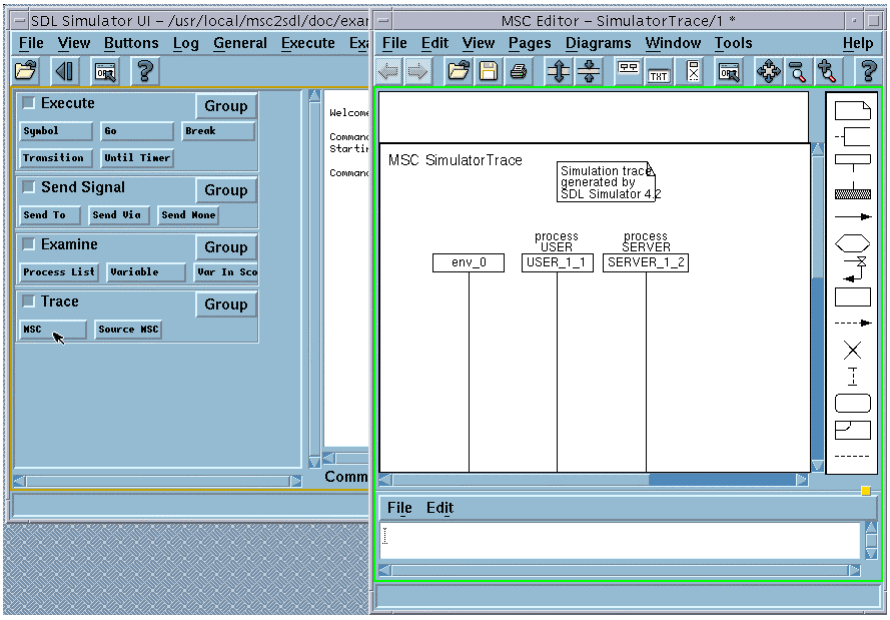
In the next part of this exercise, we will demonstrate producing graphical execution traces.

Part 4: Viewing an MSC trace

In the final part of this exercise on Viewing MSC traces, we will explore the capability of the KLOCwork MSC to SDL tool and the Telelogic Tau to produce graphical execution traces as MSCs.

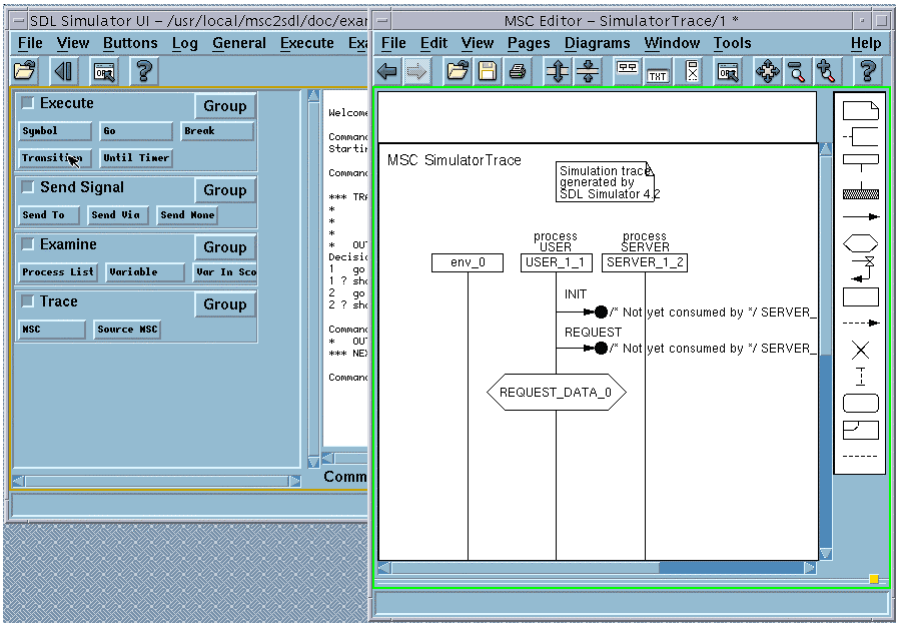
1. Restart the current simulation. Click on the MSC button in the buttons panel of the Simulator UI.
2. Simulate the model by clicking the Transition button and selecting the appropriate path in the Path Selection dialog. The Tau MSC Editor appears, however, instead of highlighting events at the input MSC diagrams it displays the execution trace as a new MSC diagram in the Telelogic Tau MSC Editor.

Figure 32: Initial state of the MSC trace



Note that the MSC trace in the MSC Editor shows three instance axes. Instances with names *USER_1_1* and *SERVER_1_2* correspond to the instances of the input MSC model. The third instance, named *env_0,* corresponds to the environment of the system. This additional instance is not involved in MSC trace at this step.

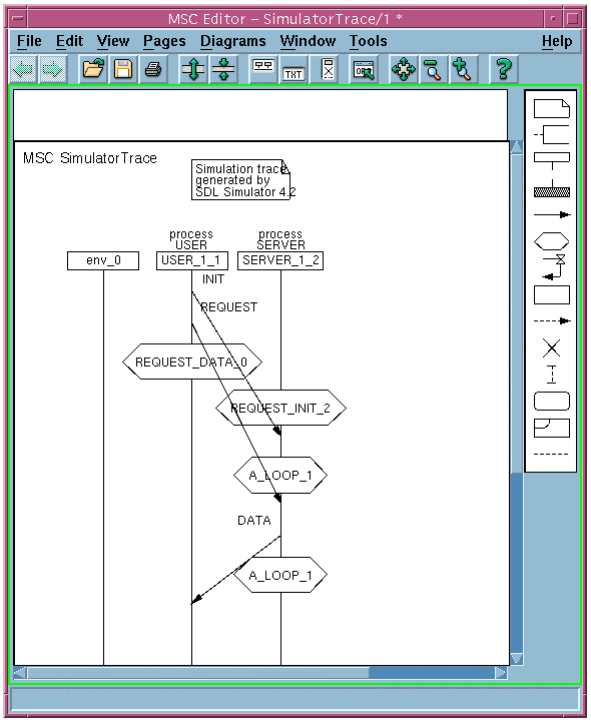
Figure 33: MSC trace after first transition



3 Now we are ready to perform the rest of the simulation. Click the Go button in the Execute group of the Simulator. The Simulator performs all transitions of the model. The Path Selection dialog appears.

4 You must decide whether the User instance will send to the Server the *Done* message (choice 1) or the *Request* message (choice 2). Select choice 2 and click OK. Several events appear on MSC trace.

Figure 34: First step of simulation



At its start transition, the User instance sends the *Init* message to the Server and then sends the message *Request*. Then it enters the state *REQUEST_DATA_0* where it waits for the Server instance to reply. At the MSC trace diagram states are displayed as conditions.

At its start transition, the Server instance enters the state *REQUEST_INIT_2,* where it waits for connection request from the User instance. In this state, the Server receives the message *Init* from the User instance and performs transition to the state *A_LOOP_1,* which means that connection was successfully initialized.

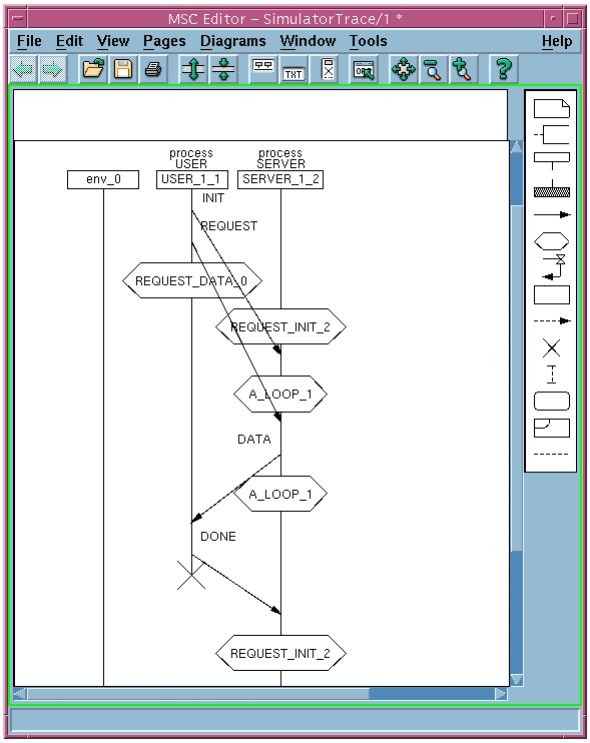
Next, the Server receives message *Request* from the User and sends the *Data* message as the reply to the User. Then the Server returns to the state *A_LOOP_1,* where it waits for the next requests from the User.

The User receives the *Data* message from the Server, and this finishes the first data interchange between User and Server.

5 The Path Selection dialog appears again. Select 1 (send message *Done* to the Server instance to terminate the connection with the Information Server).

Events, which occur after this, are shown in the following figure.

Figure 35: Second step of simulation



The User instance sends the message *Done* to the Server and then stops its execution, which means that connection is terminated.

The Server instance receives the message *Done* from the User and performs transition to the state *REQUEST_INIT_2*, where it waits for the *Init* signal.

Summary

In this exercise we explored the KLOCwork MSC to SDL Synthesizer's capability for simulating MSCs.

In the next exercise we will produce and explore the SDL model, corresponding to the MSC model of our example.

Synthesize the SDL model

In the previous exercise we explored the behavior of our MSC model through the MSC Simulation capability of the KLOCwork MSC to SDL Synthesizer. In this exercise we demonstrate how to produce an executable SDL model that corresponds to our MSC model. In the next several exercises we will explore the synthesized SDL model and will demonstrate some powerful validation capabilities provided by the KLOCwork MSC to SDL Synthesizer and the Telelogic Tau.

1 In the Telelogic Tau Organizer, select the InfoServer module and, from the MSC to SDL menu, choose *Synthesize plain SDL*.

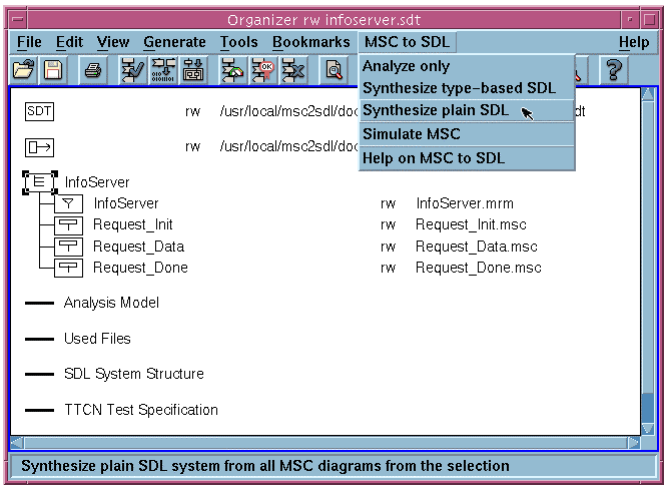


Figure 36: MSC to SDL menu, Synthesize plain SDL command

Diagnostic messages appear in the Organizer Log.

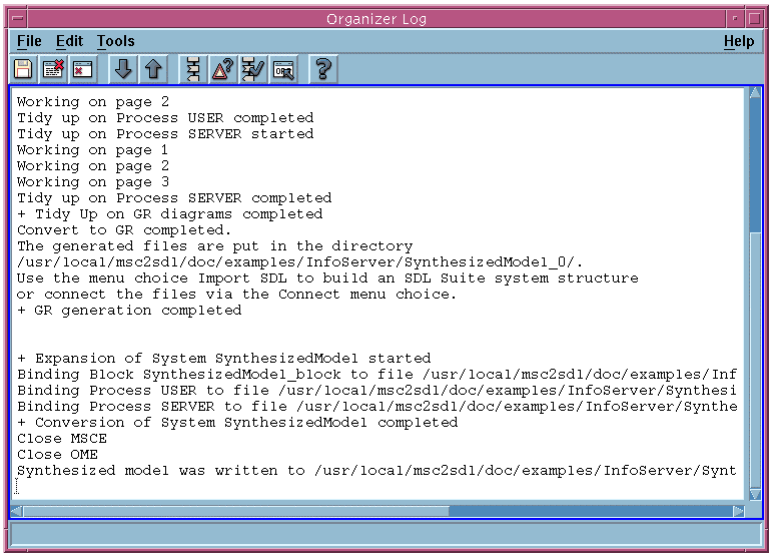
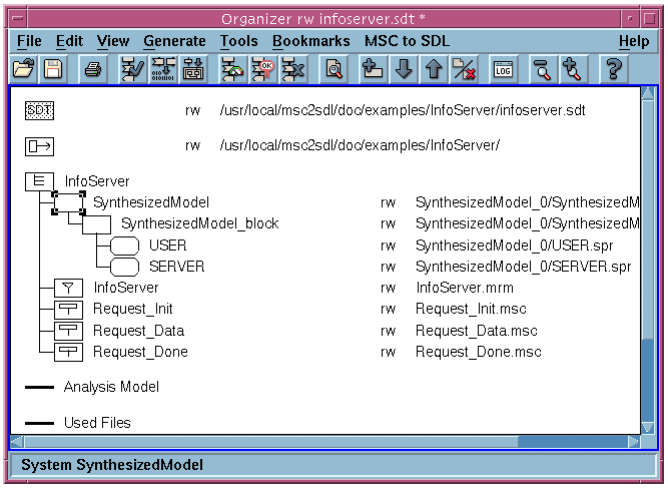


Figure 37: Organizer Log for synthesizing SDL

The KLOCwork MSC to SDL Synthesizer analyzes the input MSC model and automatically produces the SDL model. The synthesized model is then imported into the Organizer as part of the InfoServer module.

Figure 38: Results of the synthesis



Summary

In this exercise we used the KLOCwork MSC to SDL Synthesizer to produce the SDL model, corresponding to our example. The SDL model is essential to use the capabilities of the Telelogic Tau tool.

In the next exercise we will show how to use the graphical SDL Editor of the Telelogic Tau to work with the SDL model.

Inspect the synthesized SDL model

In the previous exercise we showed how to use the KLOCwork MSC to SDL tool to automatically produce the SDL model from the set of MSCs and HMSCs. In this exercise, we demonstrate how to use the Telelogic Tau SDL Editor to investigate the SDL model.

1 To start the SDL Editor, double-click on the SDL system symbol for SynthesizedModel in the Server module of the Organizer. The SDL Editor displays the diagram of the SynthesizedModel.

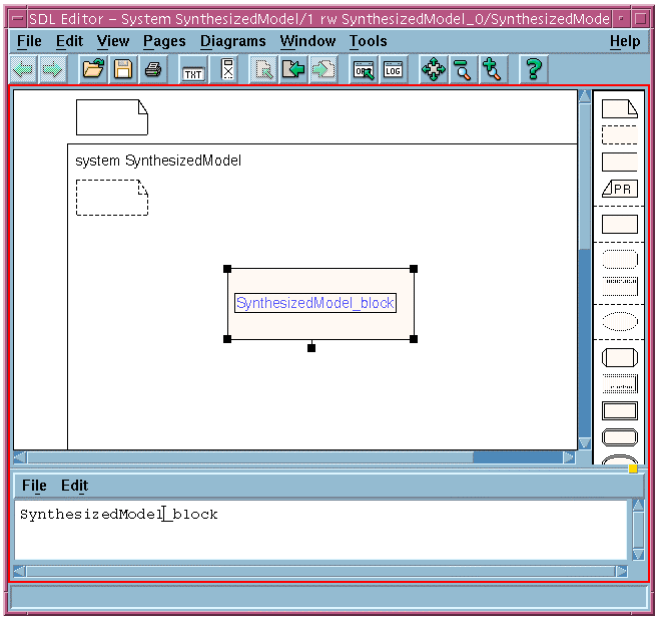


Figure 39: Top-level diagram of the Synthesized_Model

A complete set of SDL diagrams for the Server model follows:

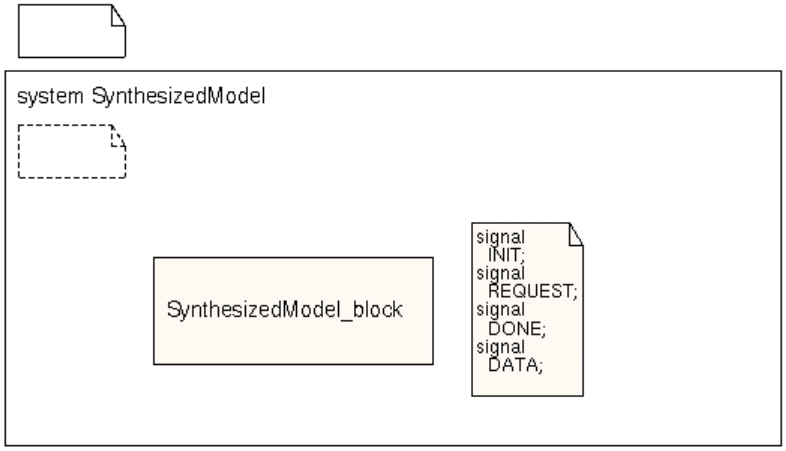


Figure 40: SDL system for the InfoServer model

Figure 41: SDL block for InfoServer model

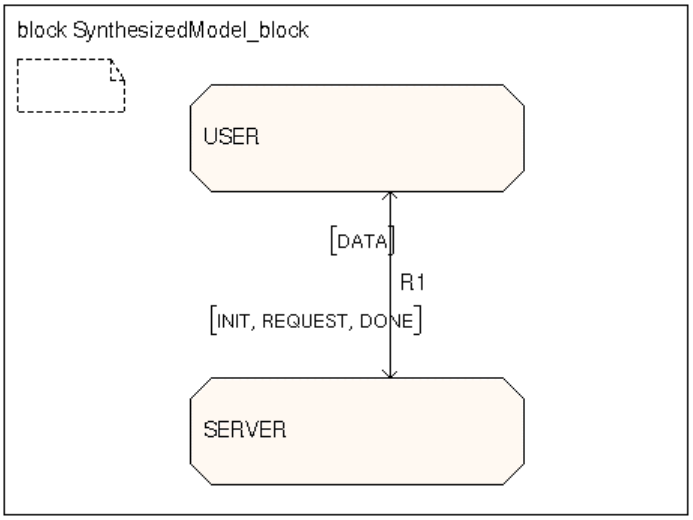
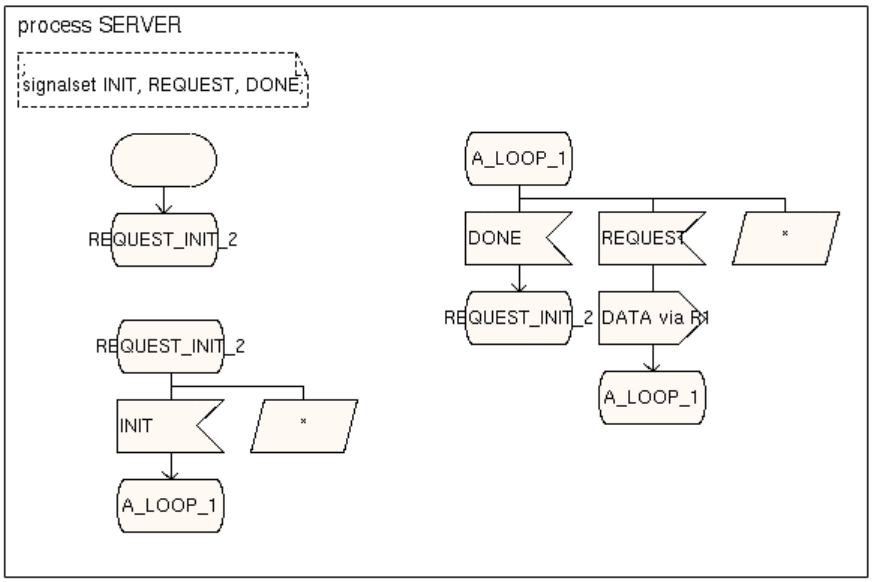


Figure 42: SDL process for the Server actor of the InfoServer model



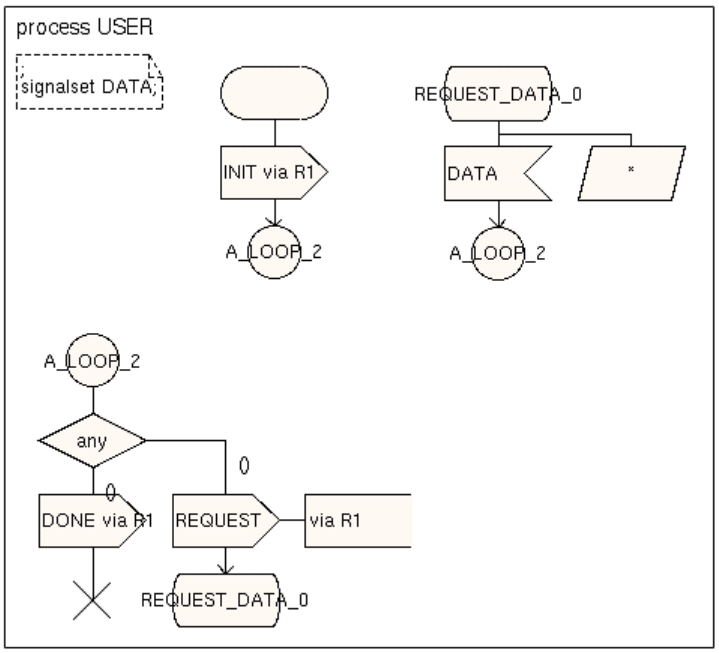


Figure 43: SDL process for the User actor of the InfoServer model

The SDL Editor has a *palette* of graphical symbols (right panel), a graphical editing area and a text editing area (bottom panel). When browsing through the generated SDL model you may want to turn off the graphical symbols palette and the text editing area to make for more room in the main graphical area. To toggle panels on and off, use their corresponding buttons in the quick button panel on top of the graphical editing area.

2 Usually, SDL diagrams consist of several *pages*. To navigate between pages, use the corresponding buttons in the middle of the quick button panel.

3 To enter a particular block, double-click on the corresponding symbol in the diagram.

4 To see the parent diagram, use the corresponding quick button in the middle of the quick button panel. Consult the Telelogic Tau documentation for more information about using the SDL Editor.

Summary

In this exercise we demonstrated how to use the Telelogic SDL Editor to inspect the automatically synthesized SDL model.

SDL diagrams in this exercise demonstrated the key points of the MSC to SDL mapping used by the KLOCwork MSC to SDL Synthesizer:

- Each MSC actor is mapped onto a separate SDL process
- All SDL processes are placed into a single SDL block
- If two MSC actors communicate, the corresponding SDL processes have a signalroute between them. The list of signals in this signalroute contain automatically generated signals corresponding to all possible messages between these two actors.
- A complete SDL process graph is synthesized for each SDL process.
- Definitions of signals are automatically generated.
- Data definitions are automatically generated.

In the following exercise we will show how to use Telelogic Tau to simulate the synthesized SDL for the InfoServer model. In the subsequent sections we will show how to use Telelogic Tau to perform automatic state space exploration on the synthesized SDL model in order to do an in-depth validation of the behavior input HMSC model.

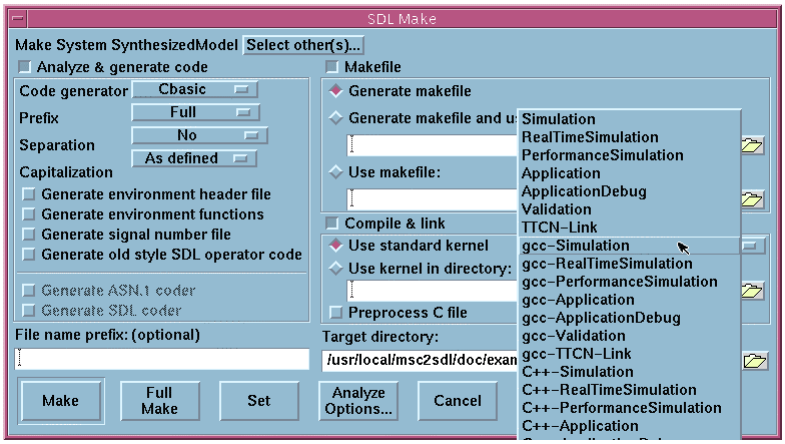
Simulate the synthesized SDL model

In this exercise we demonstrate how to simulate the synthesized SDL model using the Telelogic Tau Simulator UI.

Simulation of a synthesized SDL model is similar to the direct simulation of the input MSC model, demonstrated earlier. Direct simulation of MSCs allows the author of scenario-based specifications to interactively explore the behavior of the model without dealing with any complexities of state machines or details of SDL notation. However, at more advanced modeling stages, dealing with the SDL model becomes important because it offers a state-machine perspective of the behavior of the input model. Also, simulating the synthesized SDL model gives more advanced control over the process of simulation. Simulation of the synthesized SDL model is the topic of this exercise.

1 In the Telelogic Tau Organizer, select the synthesized system and, from the Generate menu, choose Make. Ensure that the simulation kernel corresponds to your C programming environment. In the figure that follows, we set the simulation kernel to "gcc-Simulation", which works for our environment. The SDL Make dialog offers several important options. For information on other options in the SDL Make dialog, refer to the Telelogic Tau documentation.

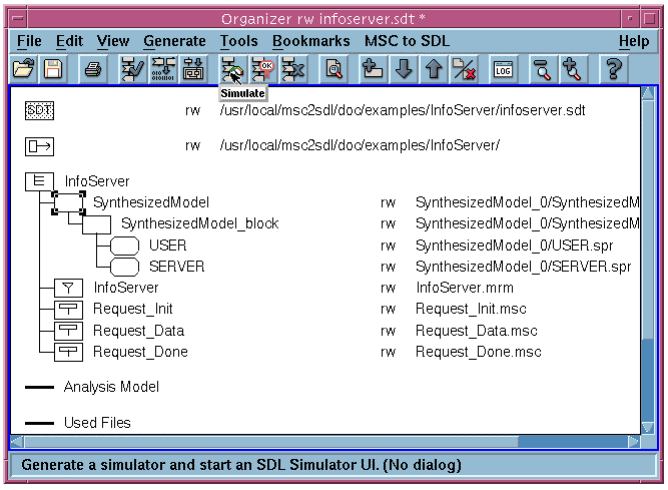
Figure 44: Selecting the simulation kernel



2 Click the Set button on the SDL Make dialog.

3 Now we are ready to start simulating the synthesized SDL model. Select the synthesized system and click the Tau Organizer.

Figure 45: Simulate button

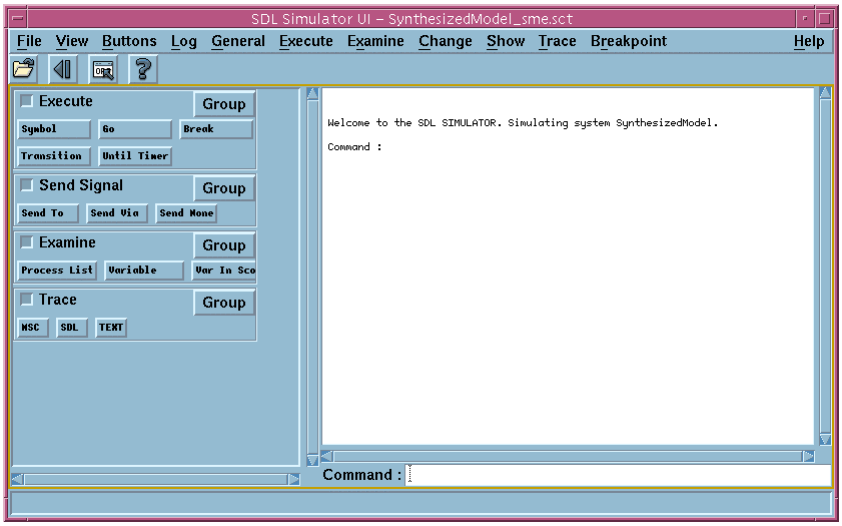


Telelogic Tau performs the following steps:

- Calls Tau code generator to produce the corresponding C code
- Executes the generated makefile to compile and link the executable model
- Starts the Tau Simulator UI

The Simulator UI, with standard SDL buttons, appears. No additional dialogs are produced.

Figure 46: Simulator UI with standard SDL-oriented buttons



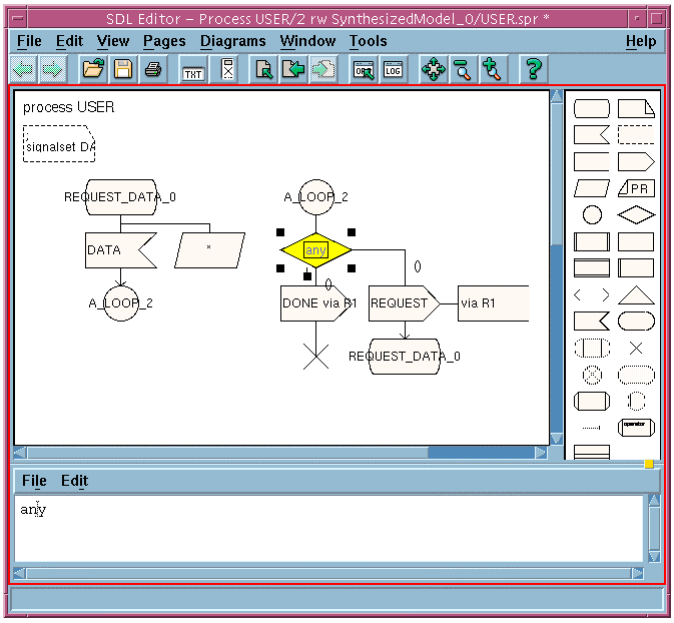
The sequence of steps in this exercise is the same as that in the Simulate MSCs exercise. However, when you directly simulate the synthesized SDL model using the SDL Make dialog you have more control over the steps.

The button definitions are generated to the target directory (file def.btns). The Simulate MSCs capability of the KLOCwork MSC to SDL Synthesizer is capable of generating model-specific buttons.

Note: To do this you may need to remove the file def.btns from your target directory in order to get standard buttons.

4 SDL-oriented buttons have the SDL button in the Trace group. Click the SDL button to allow single-stepping through individual SDL symbols as you simulate the synthesized SDL model. Run the simulation by clicking the Transition button.

Figure 47: SDL trace in SDL Editor



Summary

In this exercise, demonstrated how to simulate the synthesized SDL model using the Telelogic Tau Simulator UI.

In the next exercise, we will modify the InfoServer model and introduce some new behavior.

Extend the Server model

In this exercise we modify the InfoServer model and introduce some new behavior. The new behavior describes an exception to the original scenario, a situation in which the Server *fails* for some internal reasons and the execution terminates.

All source files (in MSC PR format), Telelogic Tau diagrams, and the synthesized SDL executable model are found in the following directory.

```
<KLOCwork MSC to SDL installation
directory>/doc/examples/InfoServer_step1
```

1 We use the HMSC Editor as the starting point of the modification. Add another alternative Failure at the a_loop condition.

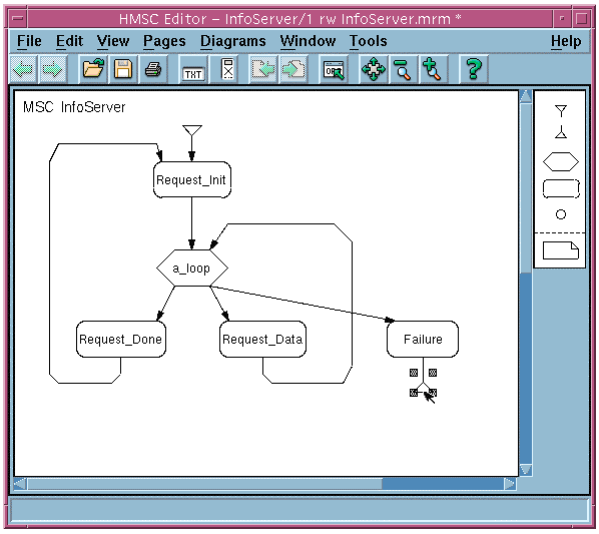


Figure 48: Adding new behavior in the HMSC diagram

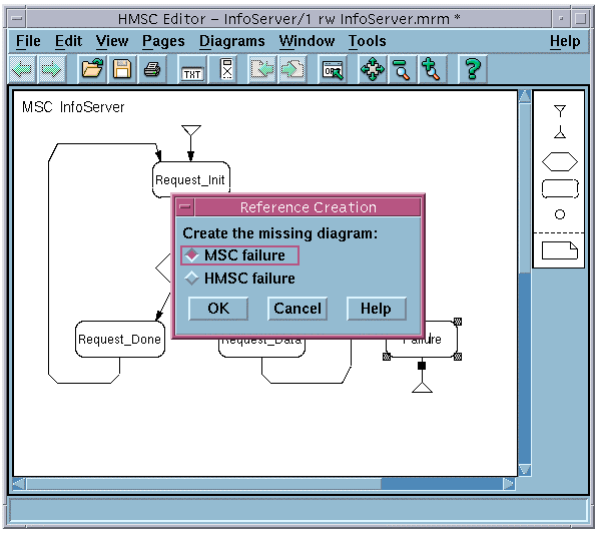
2 Add an MSC reference from the palette of graphical symbols at the right side of the HMSC Editor. Name the new symbol *Failure*.

3 Connect the condition a_loop to the new symbol by clicking the condition symbol and dragging the connection pin to the new MSC reference symbol and releasing it over that symbol.

4 Add an HMSC termination symbol from the palette of symbols. Connect the MSC reference Failure to the HMSC termination symbol.

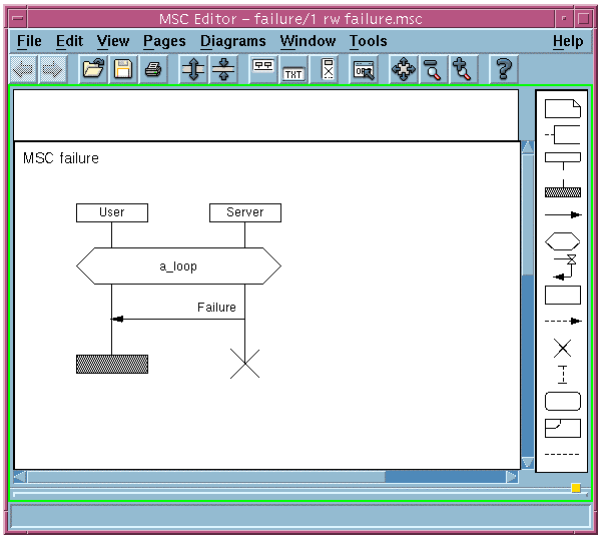
5 To create the corresponding scenario, double-click on the MSC reference symbol. This operation is supposed to automatically load the corresponding diagram. However, in our case this diagram is not yet available. You see the Reference Creation dialog that prompts you to select a type of the referenced diagram to create. The choices are: MSC diagram or HMSC diagram.

Figure 49: Diagram type selection dialog



6 Select *MSC Failure* and click OK. The missing MSC Failure is created and the MSC Editor opens.

Figure 50: New scenario in Tau MSC Editor



Summary

In this exercise we modified the InfoServer model and introduced some new behavior.

In the next exercise we will demonstrate how to perform automatic validation of the behavior of the MSC model.

Validate the extended Server model

In this exercise we use the Telelogic Validator to perform automatic validation of our modified Server model.

1 Run KLOCwork MSC to SDL Synthesizer to synthesize the SDL model from the updated HMSC model.

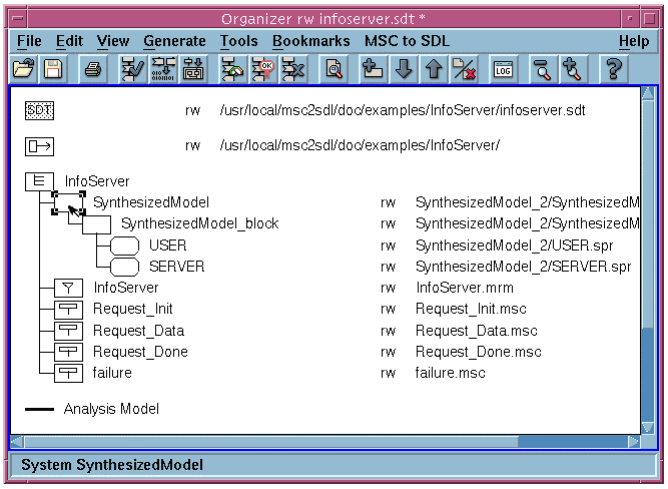


Figure 51: Generated SDL for the modified example

2 Select the synthesized SDL system symbol.

3 Select the correct validation kernel. From the Organizer Generate menu, choose Make. The SDL Make window opens.

4 Select the Use Standard Kernel option in the Compile & Link options panel and select the appropriate validation kernel from the pull-down list. (The kernel should match the C/C++ compiler installed at your machine.) Then, click the Set button.

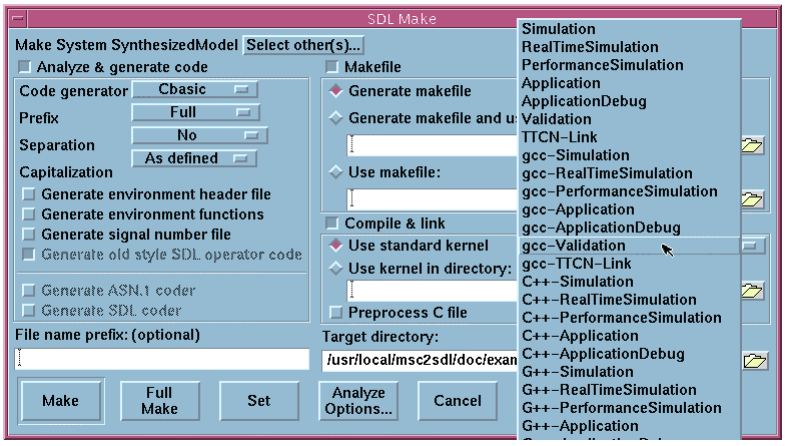
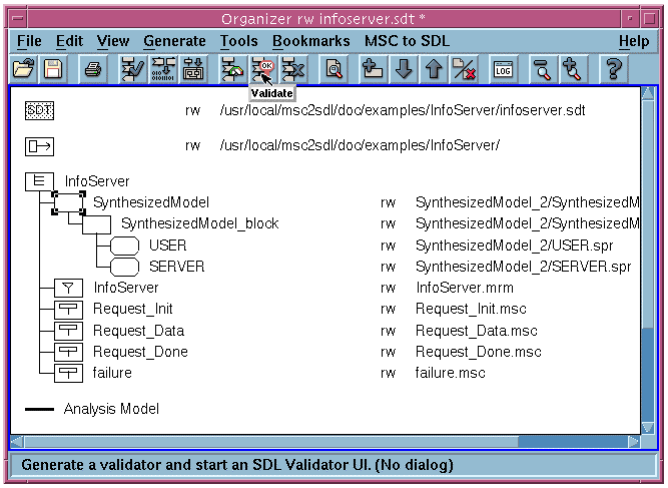


Figure 52: Selecting validation kernel

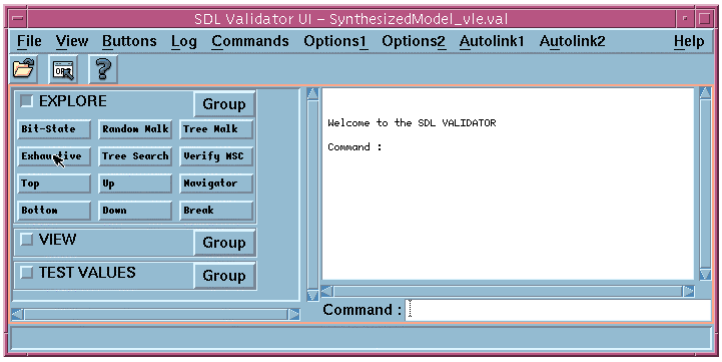
Prepare the Telelogic Validator and start the Validator UI by selecting the synthesized SDL system symbol and clicking the Validate quick button in the Tau Organizer.

Figure 53: Starting validation



The Validator UI starts.

Figure 54: Telelogic Tau Validator UI



5 Click the Exhaustive button to start the exhaustive state space exploration of the InfoServer model.

The Telelogic Validator tool analyzes the InfoServer model and produces validation reports. The reports are grouped into several categories. They include:

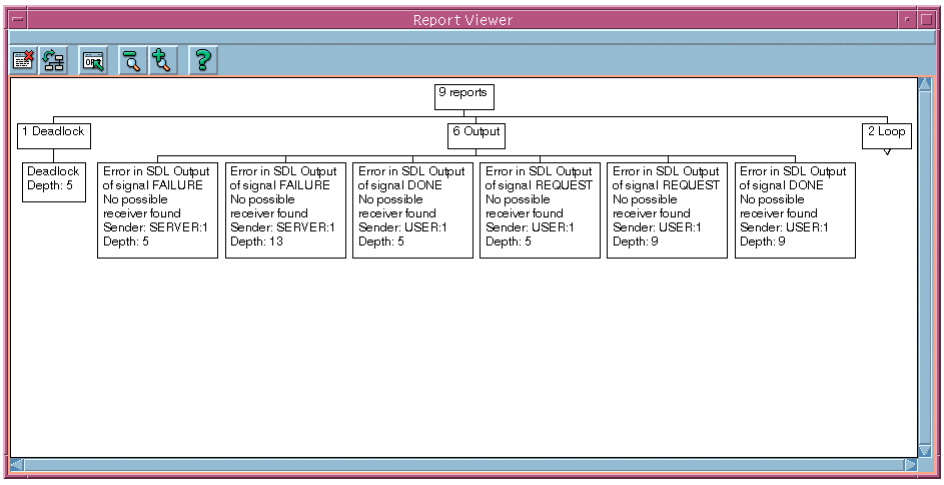
- 1 deadlock report
- 6 output reports
- 2 loop reports

Let's examine these reports.

The Telelogic Report Viewer lets you expand and collapse groups of reports in the tree.

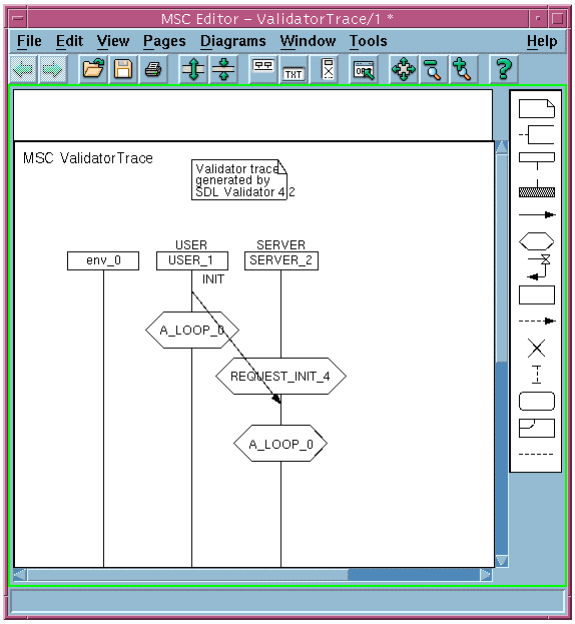
The following figure shows a validation report for our InfoServer model, produced by the Telelogic Validator using the automatically synthesized SDL model.

Figure 55: Results of validation



6 To investigate the first report, expand the deadlock report and double click on the report symbol (saying Deadlock, Depth: 5). The Report Viewer tool starts the MSC Editor to show the trace of events, illustrating the abnormal behavior reported by the Validator.

Figure 56: Deadlock, depth 5



7 Let's investigate the problems reported by the Validator. The first problem is a deadlock that happens when the User sends Init signal to the Server and then decides to enter the state *A_LOOP_0*, where it waits for a Failure signal (Failure scenario) from Server, while the Server at the same time enters the state *A_LOOP_0*, where it waits for Request or Done signals from the User instance. None of them are scheduled for transition.

The second problem is a failed output that happens when the User sends Init signal to the Server and then tries to perform the Done request (Request_Done scenario), while the Server at the same time decides to fail and sends the Failure message to the User. From observing the input scenarios is obvious that the User is not prepared to handle the failure of the Server when it has already committed to the Request_Done scenario.

Figure 57: Failed output (Failure, depth 5)

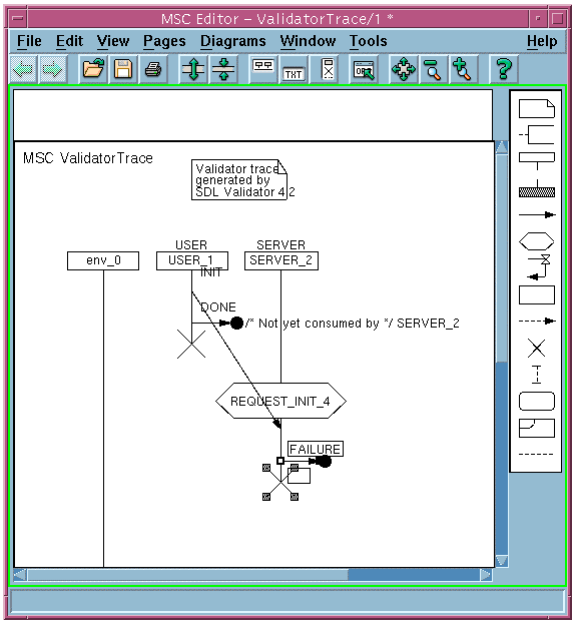


Figure 58: Failed output (Done, depth 5)

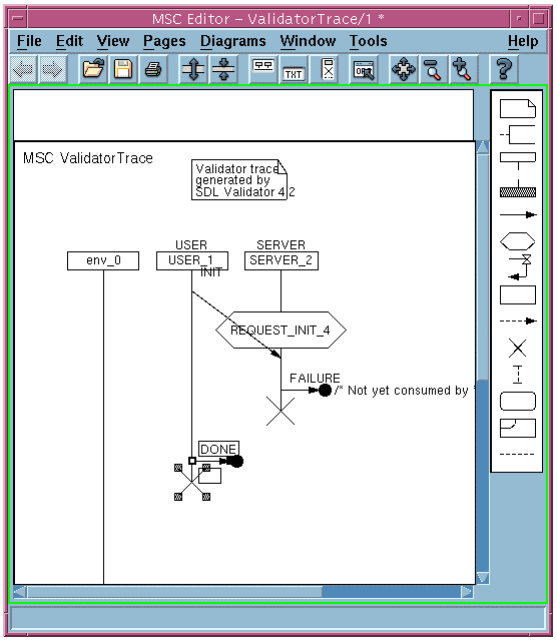


Figure 59: Failed output (Request, depth 5)

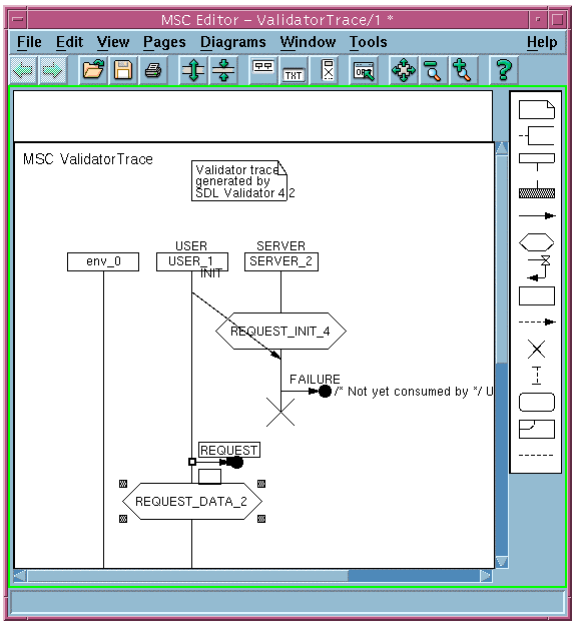


Figure 60: Failed output (Request, depth 9)

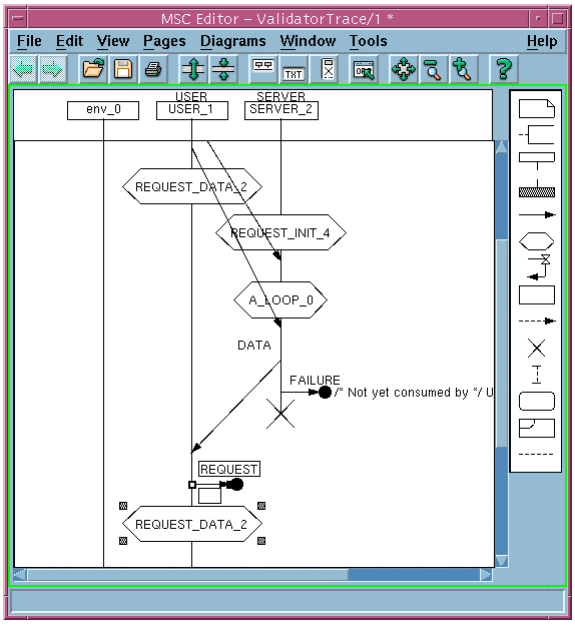
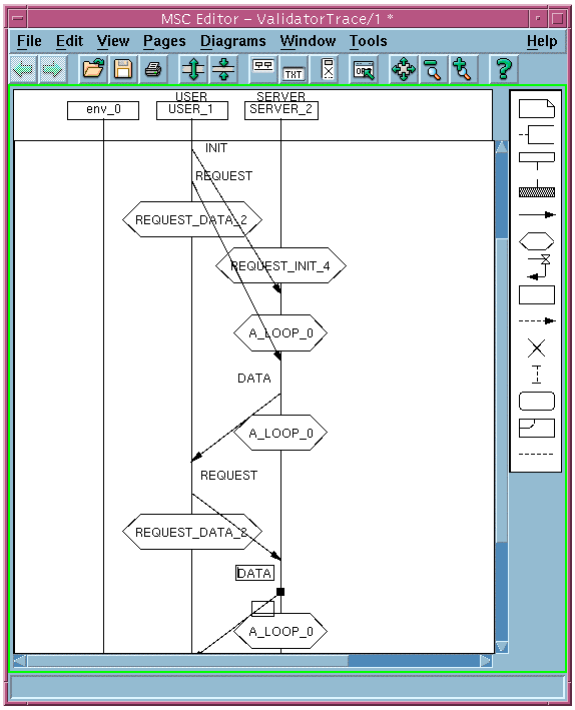


Figure 61: Infinite loop



Summary

In this exercise, we used the Telelogic Tau Validator to perform automatic validation of our modified Server model.

In the next exercise, we will fix the problem in the extended InfoServer model and perform another round of automatic validation.

Fix the problem and revalidate the MSC model

In this section we fix the problem in the extended InfoServer model and perform another round of automatic validation to verify that the problem is gone and no new problems were introduced.

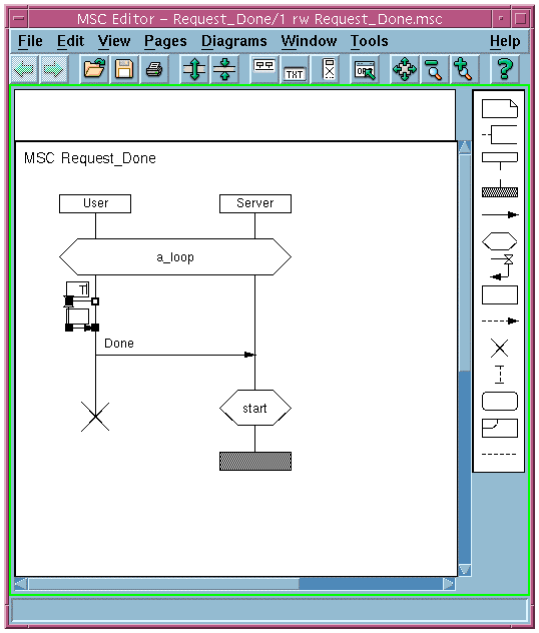
Setup

All source MSC PR files, Telelogic Tau diagrams, and the synthesized SDL model can be found in the following directory:

```
<KLOCwork MSC to SDL installation
directory>/doc/examples/InfoServer_step2
```

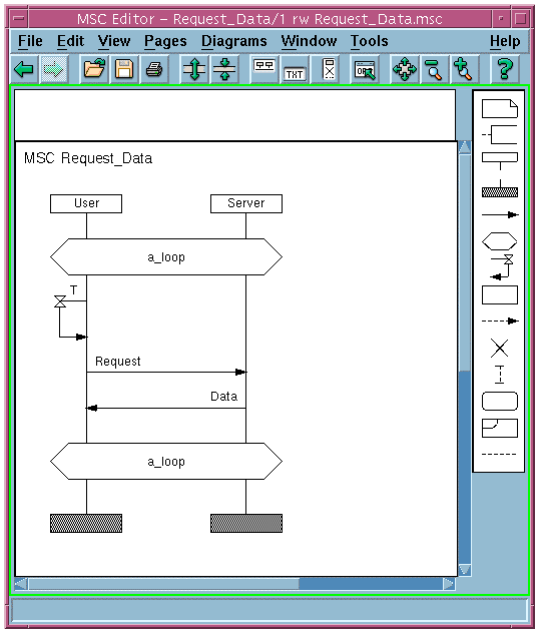
In order to fix the problem discovered by the automatic state space exploration technique used by the Telelogic Tau Validator, let's introduce a delay into the User actor so that it checks that the Server has not failed. The User then expects the Failure signal from the Server before sending any requests to it. The delay should be introduced into the Request_done scenario (as shown in the following figure).

Figure 62: Fixing Request_Done



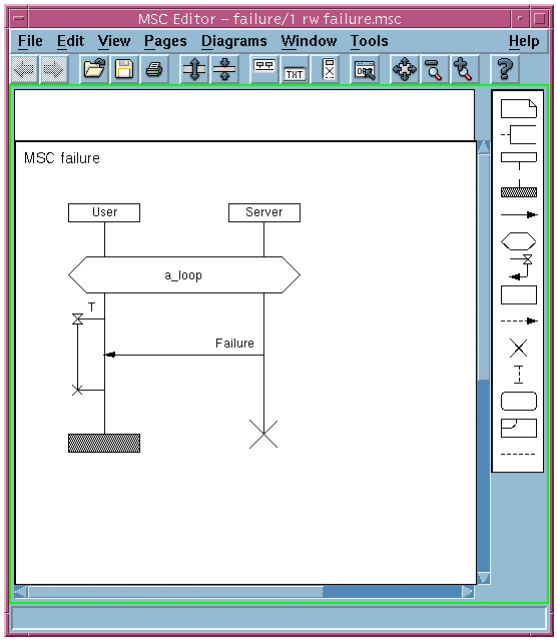
The delay should also be introduced into the Request_Data diagram (as shown in the following figure).

Figure 63: Fixing MSC Request_Data



If the Server fails during the wait period, the User will disengage itself from further interactions with the Server (as shown in the following figure).

Figure 64: Fixing MSC Failure



1. Save the updated Server model before performing the validation.
2. To perform the validation of the updated Server model do the following operations.

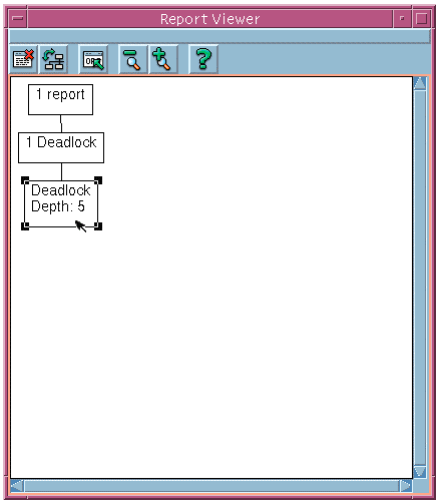
3 Synthesize the SDL model.

4 Check that the correct validation kernel is selected.

5 Select the synthesized SDL system symbol and start the Validator UI.

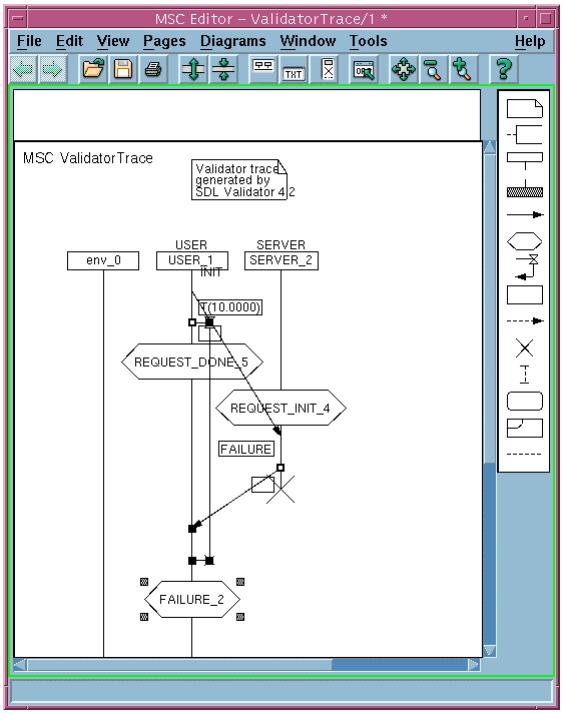
6 Select Exhaustive validation in the Validation UI. Now the Report Viewer offers only one report.

Figure 65: Revalidation results



7 Let's examine the remaining problem. Click on the symbol *Deadlock Depth: 5* in the Report Viewer. The Validator opens the MSC Editor and shows the sequence of events leading to this problem.

Figure 66: Insignificant deadlock



The MSC diagram describes the situation in which the Failure signal from the Server process arrives during the wait period of the User process. The User then disengages itself from the Server by leaving the a_loop state. The Server terminates (according to the MSC failure). This is the intended behavior. The Validator flags this situation as a deadlock because the process set of the synthesized system shows no further progress. The process User is now in a loop (state failure_2) and no further messages can force the Server process to leave this state. From the SDL perspective, this is a deadlock. However, from our modeling perspective, this deadlock is *insignificant* because, according to our input MSC model, the behavior of the system is undefined after we finish all the explicitly specified scenarios and reach the HSMC termination symbol.

To break the deadlock discovered by the Validator, we should extend the input specification of the Server and decide what the expected behavior of the User actor should be after the Server fails. One potential solution is to terminate the User actor in this situation.

Add simulation controls from the environment

In this exercise we show how to introduce environment controls into the model to provide more meaningful simulations. The goal of adding environment controls is to make the model *deterministic*. When we do this, we do not use the Path Selection dialog. Instead, we explicitly tell the model what to do by sending signals from the environment.

Setup

All source MSC PR files, Telelogic Tau diagrams, and the synthesized SDL model can be found in the following directory:

```
<KLOCwork MSC to SDL installation
directory>/doc/examples/InfoServer_step3
```

As we observed before, the InfoServer model is *non-deterministic* because the input scenarios do not contain information about the decision made between the Request and Done messages done by the User actor. The KLOCwork MSC to SDL Synthesizer propagates the non-deterministic choice into the synthesized SDL model.

1 Update the input MSCs as shown in the following figures.

Note, that the signal from the environment and the timeout signals arrive in the so-called MSC *coregion* (see the first two of the figures that follow). We use the coregion symbol to emphasize that the timeout (introduced in order to fix the failure problem) and the environment control (introduced in order to provide more control over the execution of the model) are independent.

The Telelogic Validator handled the non-determinism gracefully. It produces MSC traces for various combinations of actions by the User process. However, the manual exploration of the behavior of the Server mode through the Simulator UI used the Path Selection dialog. With the insertion of the environment controls, the model becomes deterministic. The logic of selecting between Request and Done signals now resides with the environment. The model only reacts to the incoming signals. By sending these signals, we control the behavior of the model.

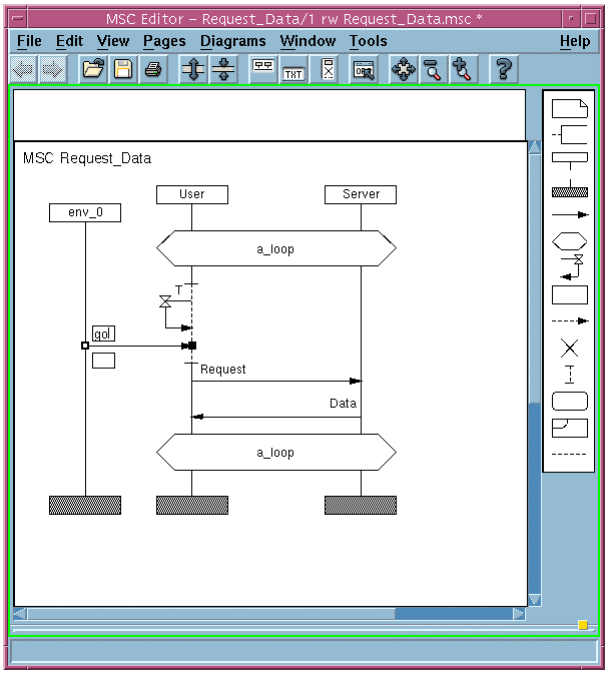


Figure 67: Adding environment control to MSC Request_Data

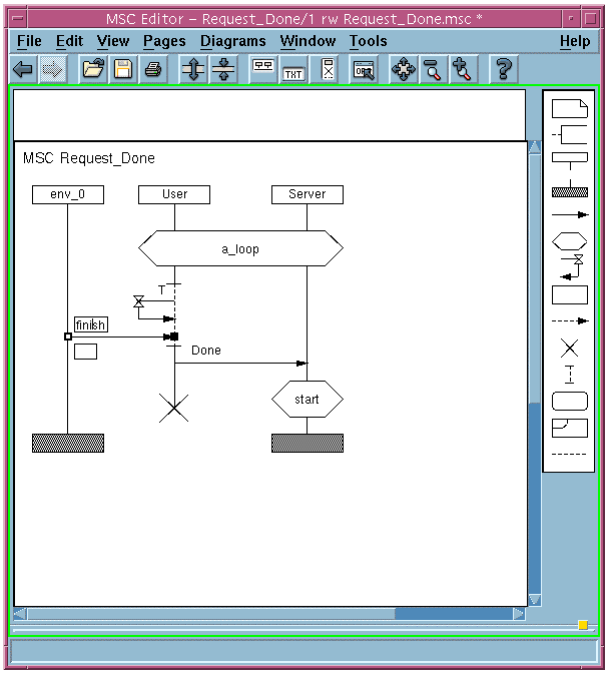
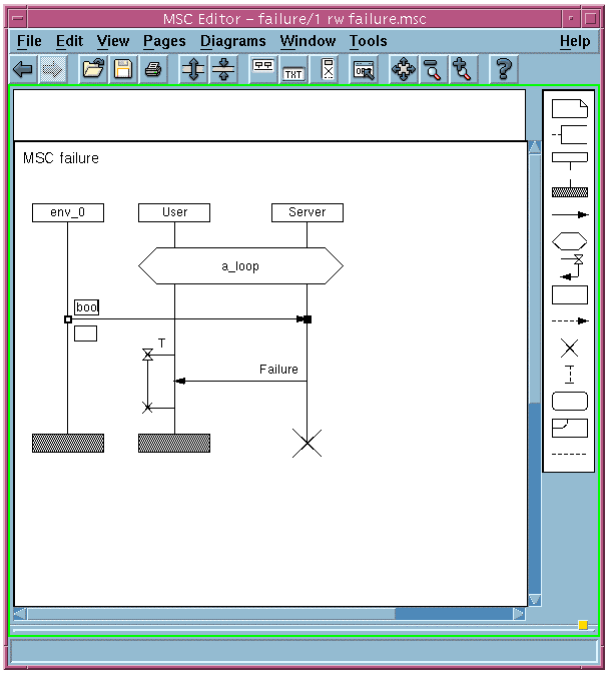


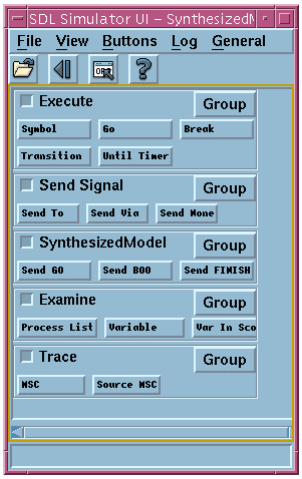
Figure 68: Adding environment control to MSC Request_Done

Figure 69: Adding environment control to MSC failure



2 Save the model and, from the MSC to SDL menu, choose Simulate MSC. Note that the Simulator UI now shows additional buttons that correspond to the new environment controls *go*, *finish* and *boo*. This provides a more friendly way to interact with the model.

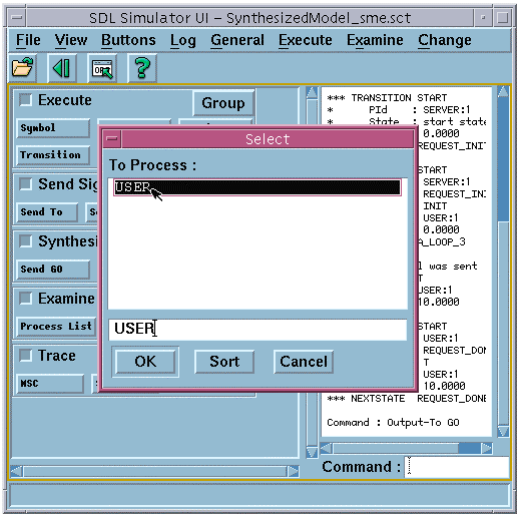
Figure 70: New buttons for SynthesizedModel



3 Click the MSC button in the Trace group. Start the simulation by clicking the Go button in the Execute group, then click the Send go button in the Synthesized_Model group.

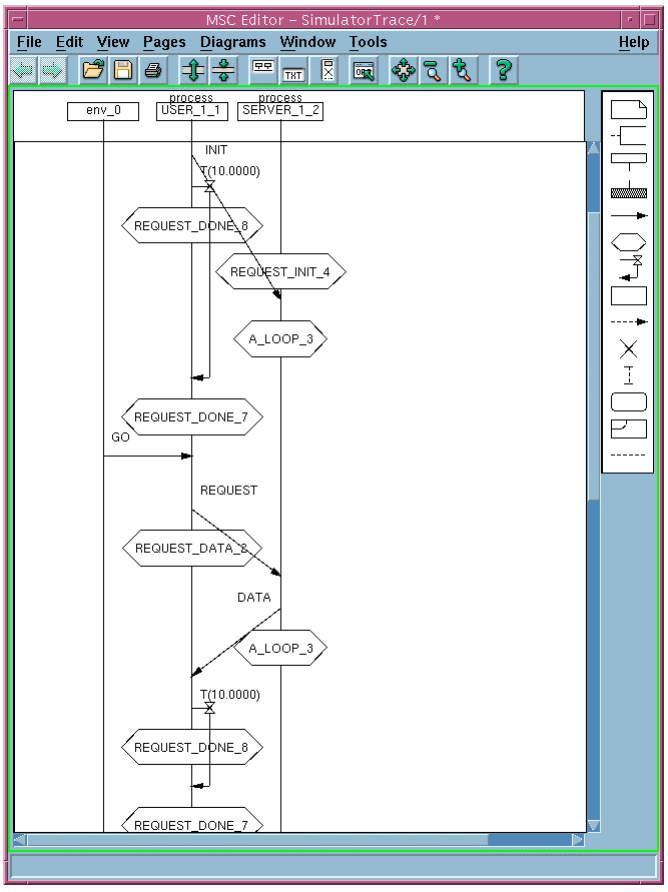
4 The Simulator prompts you for the name of the process to which to send the message.

Figure 71: Sending signal go to User



5 Select the User process. The Simulator shows the following trace.

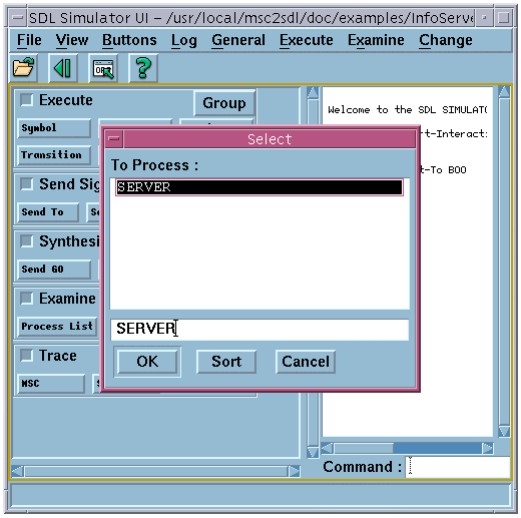
Figure 72: Simulation trace with environment controls



6 The simulation waits for more input from the environment. Now click the *Send boo* button in the SynthesizedModel panel to fail the Server.

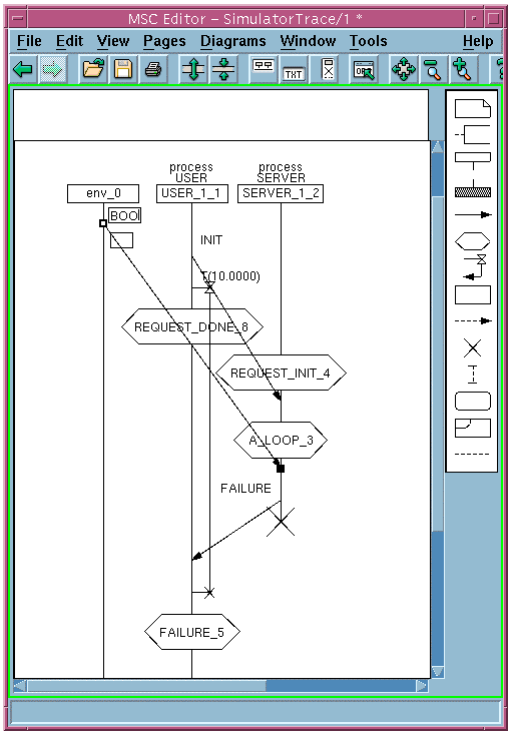
7 Select the Server process.

Figure 73: Sending signal boo to Server



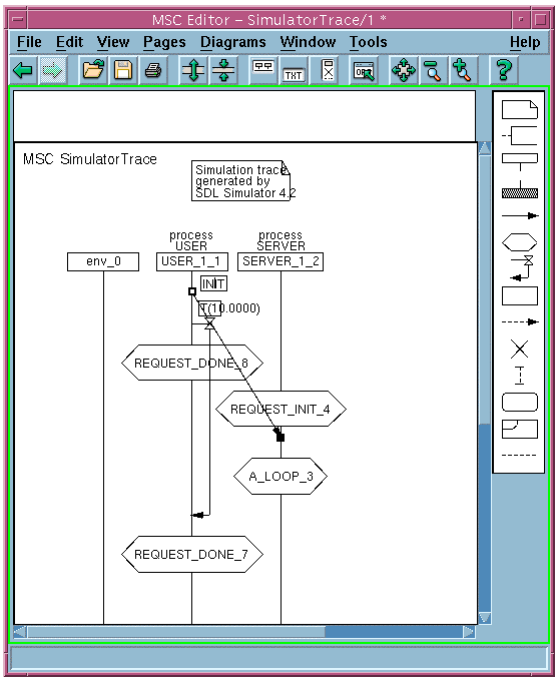
Simulation performs more events and then stops.

Figure 74: Simulation trace with environment control (part 2)



8 Restart the simulation. Click the MSC trace button. Click the Go button from the Execute group to start the simulation.

Figure 75: Simulator is waiting for a signal from the environment



Observe that the Simulator is waiting for the events from the environment. Note, however, that the Simulator is using the *event-driven simulation*. The timeout from the timer T occurred immediately after the timer was set. In the event-driven simulation this happens because the timeout is always the next scheduled event. This prevents us from exploring the failure behavior in which we need to inject the fail signal before the timeout.

Summary

In this exercise, we showed how to introduce environment controls into the model to provide more meaningful simulations.

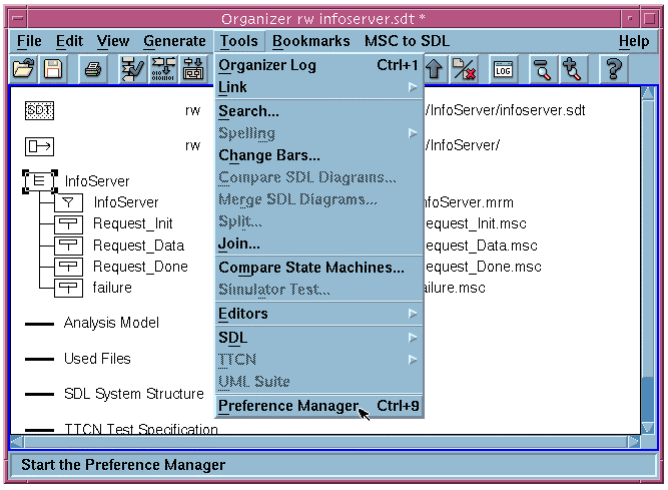
In the next exercise we will demonstrate how to run the a real-time simulation in which the events will be handled in exactly the same order as they occur. As the result, timeouts will be delayed for the exactly the amount of time for which the timer was set.

Run real-time simulation

In this exercise we show how to use the KLOCwork MSC to SDL Synthesizer and real-time simulation of the Telelogic Tau to explore behavior of the MSC scenario models.

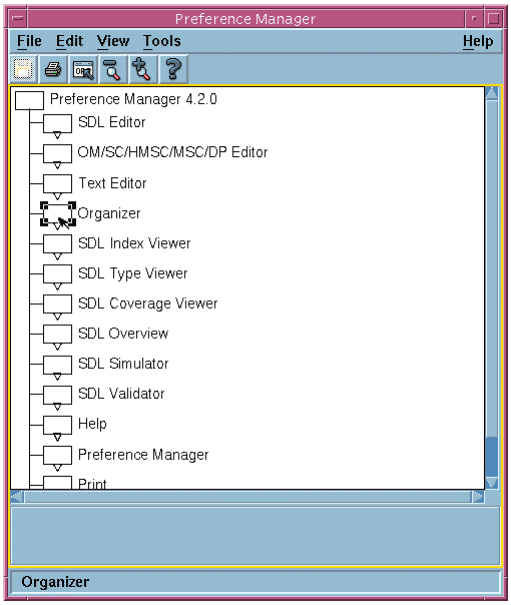
1 Use the Telelogic Preference Manager to set the default simulator to the real-time mode. To do this, from the Organizer Tools menu, choose Preference Manager. The Preference Manager window appears.

Figure 76: Telelogic Preference Manager



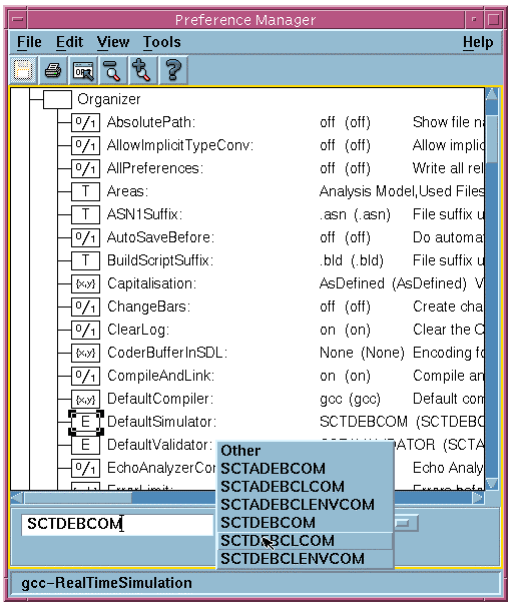
2 Double-click on the Organizer icon in the Preference Manager to see the Organizer preferences.

Figure 77: Organizer Preferences in Telelogic Preference Manager



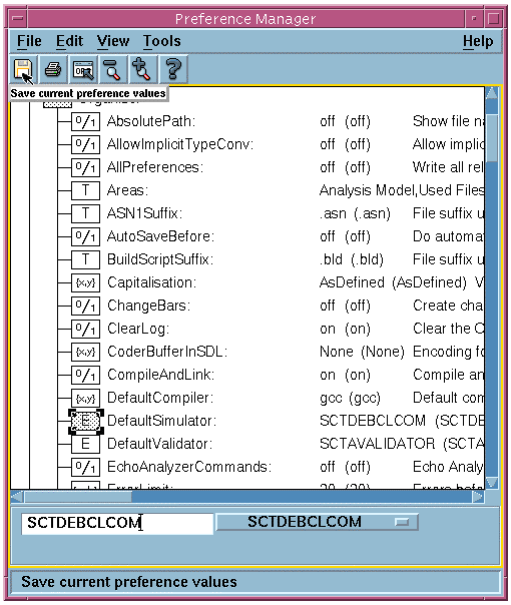
3 Select Default Simulator and choose SCTDEBCLCOM the pop-up menu.

Figure 78: Setting default simulator kernel



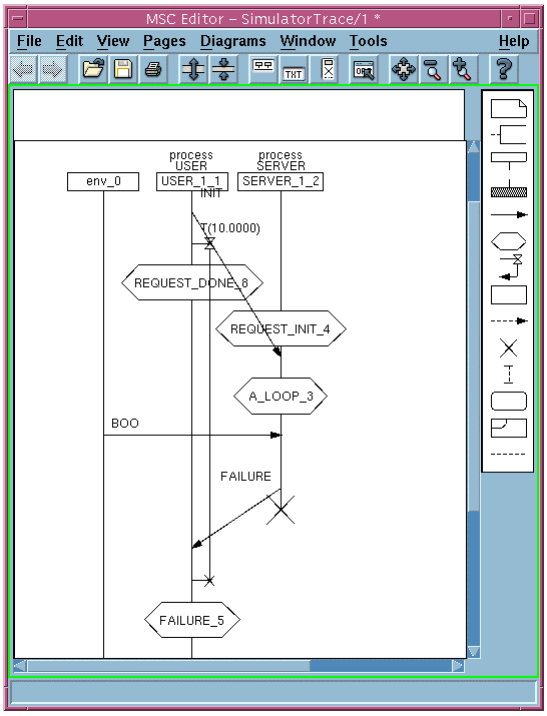
4 Save the preferences.

Figure 79: Saving the current settings in the Preference Manager



5 To restart the simulation in real-time mode, from the MSC to SDL menu, choose Simulate MSC. When the Simulator UI starts, click the MSC button in the Trace group, then click the Go button in the Execute group. Simulation starts.

Figure 80: Simulation trace for real time simulation



6 Click the Send boo button in the Synthesized_Model group before the timer T expires, then click Go again.

In real-time simulation, the timeout of the timer T occurs after the corresponding delay of 10 timer units. This allows us to send the signal boo to the Server and fail it, so that the Failure signal arrives before the timer expires. The model behaves correctly.

7 Restart the Simulator UI. Select the MSC Trace again and click the Go button to start simulation.

8 Click the Send go button to let the User do a data request, then send the boo signal to the Server to fail it.

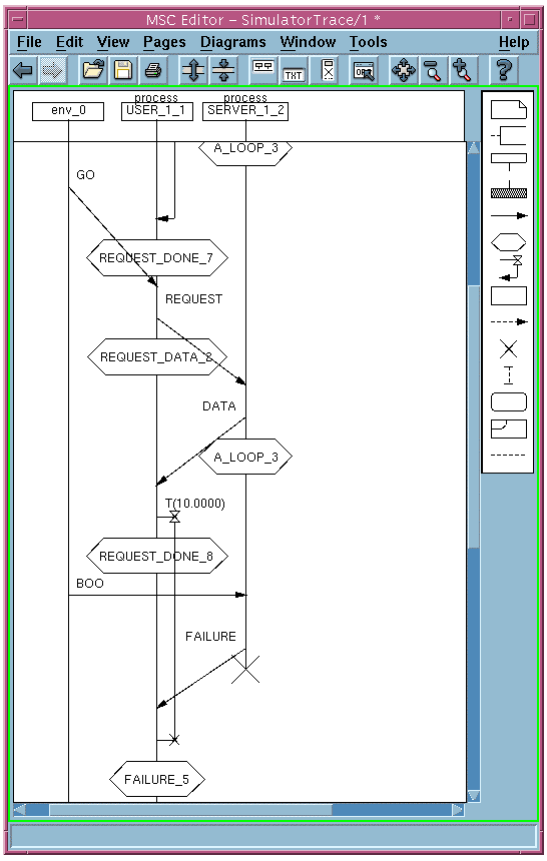


Figure 81: Signal from environment arrives during the wait period

Summary

In this exercise, we showed how to use the KLOCwork MSC to SDL Synthesizer and real-time simulation of the Telelogic Tau to explore behavior of the MSC scenario models.

In the final exercise, we will demonstrate how to use the KLOCwork MSC to SDL Synthesizer to produce *slices* of MSC models.

Simulate system slice

In this exercise, we demonstrate how to use the KLOCwork MSC to SDL Synthesizer to produce *slices* of MSC models. An MSC slice contains a subset of all instances from the original MSC model.

Our Server model consists of two instances (User and Server). The original Server model was *self-contained* (no messages to or from the environment of the model). In the previous exercises, we introduced environment controls for the User instance, thus making the model *open*. However, the instance User became trivial. All the decision-making for this user is now done in the environment of the model. The User only passes the signals to the Server model. MSC slice allows us to eliminate the User instance and interact directly with the Server instance.

Setup

All source MSC PR files, Telelogic Tau diagrams, and the synthesized SDL model are found in the following directory:

```
<KLOCwork MSC to SDL installation directory> /
doc/examples/InfoServer_step4
```

1. Copy the file **msc2sdl.ini** from the KLOCwork MSC to SDL Synthesizer installation directory to your working directory.

   The file should have the following contents:

   ```
   # comment
   OPT_UPPERCASE=yes
   OPT_ALL_WARNINGS=yes
    OPT_CROSS_REFS=yes
   OPT_CLEAN_GEN_PR = yes
   ```

2. Add the following two lines (shown in bold font) to the file.

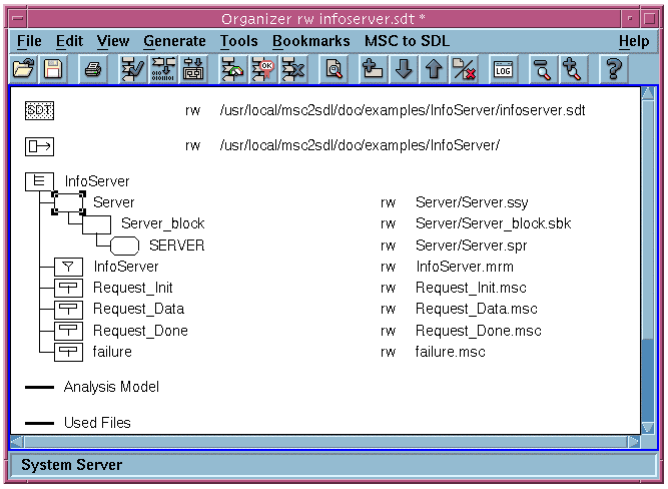
   GENERATED_SYSTEM_NAME = "Server"

   SLICE = "Server"

   The msc2sdl.ini file contains the options for the KLOCwork MSC to SDL Synthesizer. The GENERATE_SYSTEM_NAME option provides the name of the generated SDL model corresponding to the MSC slice. The SLICE option defines the list of MSC instances to be included in the MSC slice during the synthesis. For a complete description of KLOCwork MSC to SDL Synthesizer options, see the *KLOCwork MSC to SDL Synthesizer Reference Manual*.

3 Synthesize the SDL model that corresponds to the selected MSC slice. To do this, in the Telelogic Tau Organizer from the MSC to SDL menu, choose Synthesize plain SDL. The synthesized slice is placed into the Server module of the Organizer.

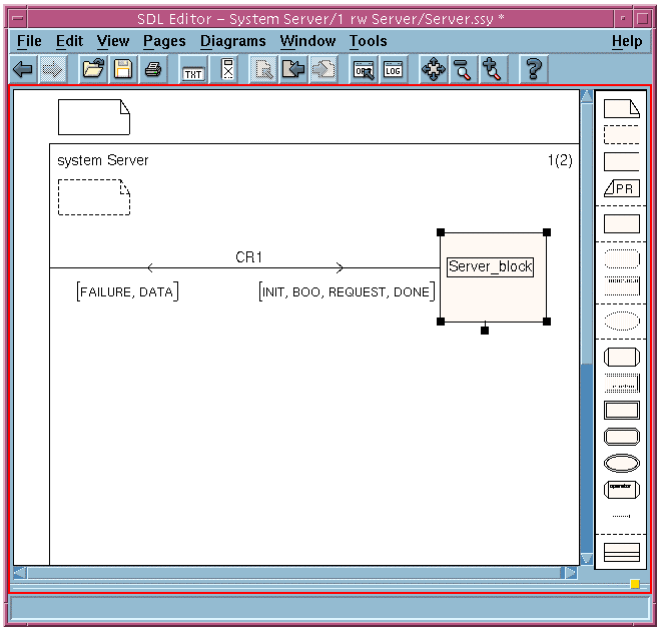
Figure 82: Synthesized slice



Observe that, in comparison to the complete generated SDL model, the slice contains only the SDL process SERVER. Note, also, that the synthesized SDL model has a different name (Server instead of the default name SynthesizedModel) as the result of setting the GENERATED_SYSTEM_NAME option.

4 Inspect the SDL model of the synthesized slice.

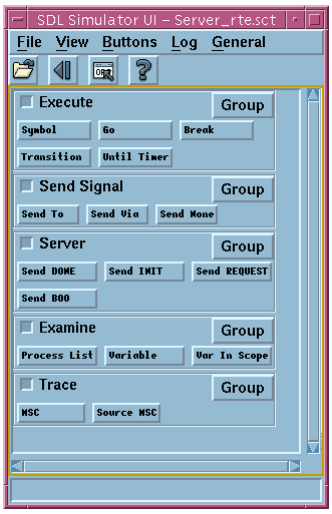
Figure 83: SDL system diagram for the synthesized slice



Observe that the synthesized SDL system Server has a channel to the environment. The list of signals for this channel corresponds to the interface of the Server instance according to the input MSC model.

5 From the MSC to SDL menu, choose Simulate MSC. Observe new synthesized buttons in the Simulator UI button panel. The synthesized buttons correspond to the input interface of the Server instance.

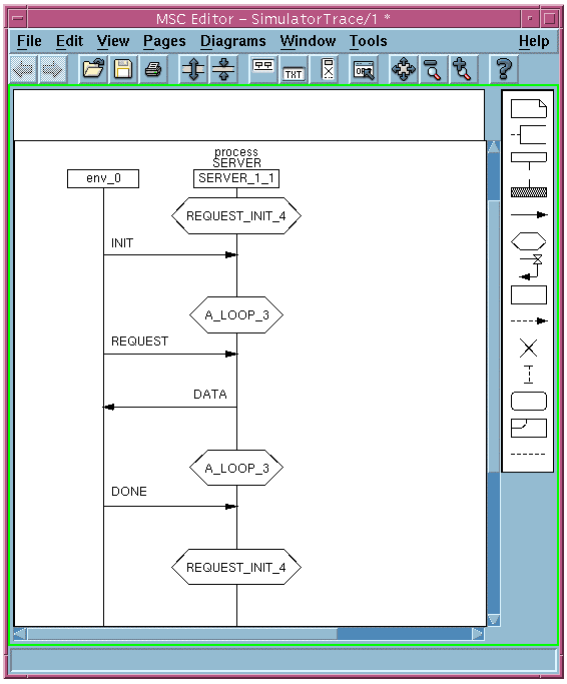
Figure 84: Synthesized buttons for the slice



6 Click the MSC button in the Trace group, then start the simulation by clicking the Go button from the Execute group.

7 Explore the behavior of the Server by sending the following signals: Init, Request, Done. Observe the generated MSC trace.

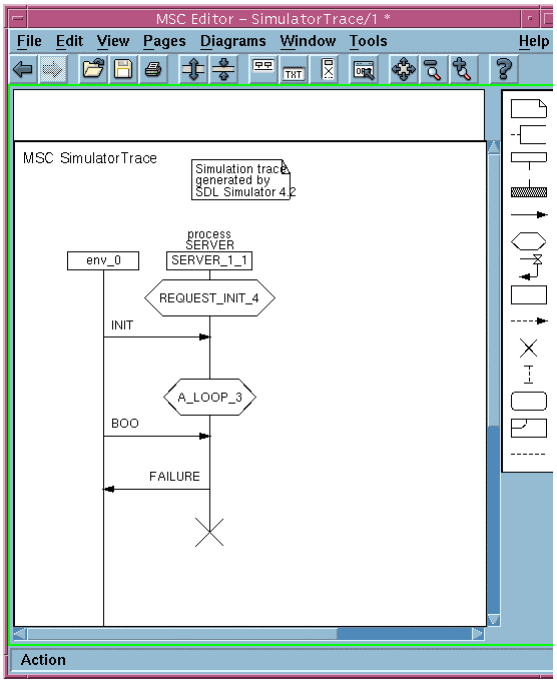
Figure 85: Simulator trace for the slice (Request; Done)



8 Restart the Simulator UI. Click the MSC button and start the simulation by clicking the Go button.

9 Explore the abnormal behavior of the Server by sending the following sequence of signals: Init, Boo. Observer the generated MSC traced.

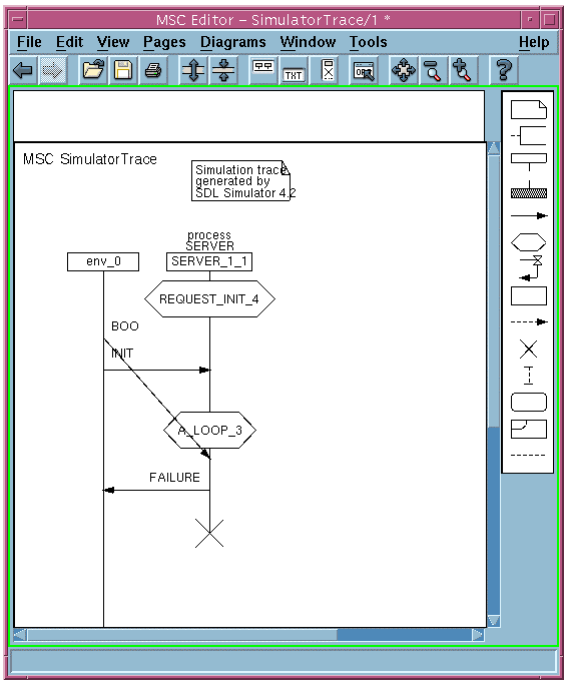
Figure 86: Simulator trace for the slice (Failure after Init)



10 Restart the Simulator UI. Click the MSC button and start the simulation by clicking the Go button.

11 Explore the abnormal behavior of the Server by sending the following sequence of signals: Boo, Init. Observe the generated MSC traced.

Figure 87: Simulator trace for the slice (Failure during Init)



Summary

In this exercise we demonstrated how to use the KLOCwork MSC to SDL Synthesizer to produce *slices* of MSC models.

Index

KLOCwork Corporation

Toll-free telephone: 1-866-556-2967
E-mail: sales@klocwork.com **or**
support@klocwork.com
Website: http://www.klocwork.com

Corporate Headquarters:
1 Antares Drive, Suite 510
Ottawa, Ontario, Canada
K2E 8C4
(613) 224-2277

US Headquarters:
1700 Montgomery Street
Suite 111
San Francisco, CA
94111
(415) 954-7154