Chapter

41 *The Deployment Editor*

The Deployment Editor is a tool for graphical partitioning of SDL systems, which allows you to model how SDL systems execute in their target environment. See also chapter 65, *Integration with Operating Systems*.

Diagrams in the editor, deployment diagrams, use notation from UML. Information from a deployment diagram can be used when targeting one or several SDL systems. See chapter 60, *The Targeting Expert*.

This chapter contains a description of the deployment diagram as well as guidelines for modeling different deployment scenarios. The connection with the Targeting Expert is also described.

Introduction

The Deployment Editor allows you to graphically deploy one or several SDL systems. SDL systems can be partitioned into components, i.e. executable files, which in turn are partitioned into threads of execution. Finally, SDL entities, i.e. objects, are mapped to the threads, showing the execution architecture at compile-time.

The information in a deployment diagram can be used by the Targeting Expert through the Partitioning Diagram Model. In the Targeting Expert, you can set build parameters on each component and build an application from this model.

Applications

You can use the Deployment Editor in two ways:

- As an editor for graphical partitioning of SDL systems.
- As a pure modeling tool.

The Deployment Editor gives you freedom in modeling partitioning and communication. There are few restrictions in drawing rules and no syntax checking is performed. There are, however, restrictions how to use deployment diagrams for generating partitioning data for the Targeting Expert. This chapter describes the requirements for using the Deployment Editor for partitioning. The diagram symbols are described as they are interpreted when used for partitioning.

Starting the Deployment Editor

You start the Deployment Editor the same way as the other editors in the SDL suite. There are four alternatives:

- Select *Tools -> Editors -> Deployment Editor* in the Organizer to start with an empty diagram.
- Select *Edit -> Add New...* in the Organizer and *UML -> Deployment* in the *Add New* dialog to add a new diagram.
- Select *Edit -> Add Existing...* in the Organizer to add an existing diagram.
- Right-click on an existing deployment diagram symbol in the Organizer and select *Edit -> Edit* in the pop-up menu.

The Graphical User Interface

The graphical user interface in the Deployment Editor is similar to that used in the other SDL suite editors. For a description of the user interface in general, see chapter 1, *User Interface and Basic Operations, in the User's Manual*.

Connection to the Targeting Expert

The Deployment Editor generates data that can be used by the Targeting Expert. Deployment diagrams are translated to the data format Partitioning Diagram Model (PDM), which contains build-related information. See "Generating Partitioning Diagram Data for the Targeting Expert" on page 1723.

To start the Targeting Expert with PDM data for a deployment diagram:

1. Click on the deployment diagram (.sdp file) in the Organizer
2. Select *Targeting Expert* from the *Generate* menu.

See chapter 60, *The Targeting Expert, in the User's Manual* for information on how to build using the Targeting Expert.

Using Information from SDL System(s)

The Deployment Editor has no automatic access to information about SDL systems in the Organizer. You must enter information about SDL objects manually in a deployment diagram.

For SDL objects, the following information is mandatory to enter:

- *Name*

  It is recommended that the name of the object in the Organizer is used.

- *Qualifier*

  This is the *path* to the SDL object in the Organizer. The qualifier should include the name of the object in the Organizer.

- *Stereotype*

  This shows the type of SDL object and should have one of the following values: `system`, `block` or `process`.

See "Object" on page 1713 for more information about SDL objects.

Note:

It is possible to deploy objects from many SDL systems in one deployment diagram. The only requirement is that the SDL systems must be contained within the same `.sdt` file. This `.sdt` file must be loaded in the Organizer when working with the deployment diagram.

The Deployment Diagram

In this section, the deployment diagram is described. The view and the symbols as well as drawing rules for the different integration models are explained.

Note:

Following the naming convention in the SDL suite, diagrams edited in the Deployment Editor are named deployment diagrams. This notion must **not** be mistaken for the UML deployment diagram. In this chapter, "deployment diagram" implicitly means the SDL suite diagram.

Note:

In the Deployment Editor, it is only possible to use one page per diagram. This differs from the OM Editor, where an arbitrary number of pages can be used. See "OM Editor Specific Information" on page 1647 in chapter 40, *Using Diagram Editors, in the User's Manual*.

The View

The deployment diagram uses a view that is similar to the UML component view. The most important similarities are:

- Both views are type-based, i.e. no instance-specific information can be modeled. This implicates that the thread and object symbols in the Deployment Editor diagram can be regarded as stereotyped UML class symbols.
- Both views are static, i.e. partitioning at compile-time is modeled.

There are some important differences between the Deployment Editor view and the component view:

- The Deployment Editor view uses the node symbol, which is not used in the UML component view. In UML, the node symbol is only used in the deployment view.
- The Deployment Editor view does not support the interface notion, which is used for modeling communication between components in the UML component view.

The Deployment Editor diagram shows a static view of the partitioning, i.e. you can deploy SDL systems by how they look at compile-time.

The Symbols

The deployment diagram contains four symbols that can be used for modeling partitioning: Node, Component, Thread and Object. Each symbol is described in the following subsections. See "About Symbols and Lines" on page 1591 in chapter 40, *Using Diagram Editors, in the User's Manual*.

> **Note:**
>
> The deployment diagram terminology must not be mixed up with the syntax used in buildscripts, i.e. scripts containing SDL Analyzer commands. Some terms, such as *thread*, are similar, while other, such as *component*, represent completely different things.

All symbols, except the thread symbol, contain information that can be edited in the *Symbol Details* dialog box. Properties and stereotype information are graphically visible on the symbols.

Node

A node symbolizes a computational resource, i.e. something with processing power and possibly some memory and other peripherals. A node should be seen as a platform on which software execute. Only components can be attached to nodes.

A typical stereotype for a node is a description of the processor and a common property is the operating system that is being used. Figure 314 shows a node called **My_Computer** with the stereotype **PC** and the property **Windows NT 4.0**.

Figure 314 A node with name, stereotype and properties

For a node, you can set the stereotype to *external*, indicating that the node does not execute any code from your SDL system(s). This makes it possible to model external non-SDL parts that communicate with your SDL system(s). Setting the stereotype to `external` implicates that the node is ignored when building a Partitioning Diagram Model from the deployment diagram.

Note:

Stereotype and property information for a node is ignored when generating partitioning diagram data. The only exception is when stereotype is set to `external`.

Component

A component symbolizes an executable file, i.e. a sort of container for software entities. Components contain objects, i.e. SDL systems, blocks or processes, that are organized in threads of execution. You can attach either objects or threads to a component.

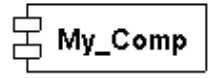

Figure 315 A component symbol named "My_Comp"

As for nodes, you can set `external` as stereotype on a component, which means that the component does not contain any SDL object from the system(s) that are deployed.

Note:

Stereotype and property information for a component is ignored when generating partitioning diagram data. The only exception is when stereotype is set to `external`.

For a component, you can select an integration model in the *Symbol Details* dialog box. The integration model reflects how the code that is generated is integrated with your target operating system. Three integration models are available:

- *Light Integration.* This is the default selection.
- *Tight Integration.*

- *Threaded Integration.*

See "Integration Models" on page 1716 for more information.

Thread

The thread symbol symbolizes a static thread of execution at compile-time. Threads can only be attached to components. Only objects, i.e. SDL entities, can be attached to a thread.

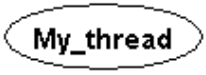

Figure 316 A thread symbol named "My_thread"

The thread symbol can only be used for modeling threaded integrations, i.e. for a Deployment Diagram with threads. Partitioning Diagram Data can only be generated if the component has "Threaded" as integration model.

You can make four settings for a thread through the "Symbol Details" dialog box. These are:

- *Thread Priority*
- *Stack Size*
- *Queue Size*
- *Max Signal Size*

The settings are available to the Targeting Expert through the Partitioning Diagram Model. The settings are common among real-time operating systems.

The values that are entered are platform-specific and are used when compiling the system for your target environment. You are free to enter any value, e.g. a number or a macro, that makes sense in your target environment.

You can choose to either explicitly set a value, or leave it empty. Any setting that is not filled in is considered undefined. In that case, a default value is used.

Note:

Your target operating system may have thread settings that do not exactly match the available thread settings in the Deployment Editor. If you know that a setting is not available in your Operating System, leave the text box empty.

Object

The object symbol is a general notion of an SDL system, block or process. The stereotype field is used to state the type of SDL symbol. This information is mandatory and has three possible values:

- `system`
- `block`
- `process`

The stereotype value is shown within the object symbol. Figure 317 shows an object with `process` as stereotype.

Figure 317 An object symbol with name and stereotype

Note:

It is recommended that the object name in the deployment diagram correspond to the name of the SDL object which is deployed. However, it is not required that the names are identical. For instance, if you add two SDL objects with identical SDL names to the same namespace, you might want to separate them by changing the object name of one the SDL objects. You can freely change the object name, as the object is uniquely identified by its qualifier.

For an object, it is mandatory to enter information in the Qualifier field in the Symbol Details dialog box. The qualifier consists of the *path* for an SDL object in the Organizer tree structure. '/' is used as a separator. The path must include the object name itself.

Some examples of object qualifiers for the AccessControl system are shown in the table below. You can verify the qualifiers by opening the

AccessControl system in the Organizer (`<SDL suite installation directory>/examples/AccessControl/Accesscontrol.sdt`) and comparing them with the entries in the table.

Object Name	Qualifier
CodeReader	`AccessControl/Local/CodeReader`
Controller	`AccessControl/Local/Controller`
Central	`AccessControl/Central`
Display	`AccessControl/Local/Display`
Doors	`AccessControl/Local/Doors`

An example of an object with all mandatory information entered is given in Example 299.

Example 299: Object Name, Qualifier and Stereotype

In the AccessControl system, the process *Central* in block *CentralUnit* has `AccessControl/CentralUnit/Central` as qualifier. For readability, the object is given the name `Central`. The stereotype is set to `process`.

Associations and Aggregations

The Deployment Editor diagram uses associations for modeling relations between nodes. Composite aggregation is used for the following relations:

- Node - Component
- Component - Thread
- Component - Object
- Thread - Object

There is a *Line Details* dialog box, which is common for all sorts of relations. For an association, you can model protocol, encoding and direction data. For a composite aggregation, only multiplicity data can be edited. For a more detailed description of the dialog box, see "DP Editor Specific Information" on page 1667 in chapter 40, *Using Diagram Editors, in the User's Manual*.

Note:

Associations between nodes are ignored when partitioning diagram data is generated.

Integration Models

The Deployment Editor supports three different integration models. The choice of integration model is set at component level.

Light Integration

In the light integration model, one or many SDL objects execute within one single task. The integration model is called *Light* in the *Symbol Details* dialog box.

The thread symbol is not used when modeling a light integration. All objects that should execute in a component with a light integration should be attached directly to the component. See Example 300 for an example of a light integration.

Example 300: A Light Integration ––––––––––––––––––––––––––––

This example (see Figure 318) shows a light integration containing two objects from the *AccessControl* system. All process instances run in one single task in one executable file. The objects are connected directly to the component, which has *Light* as integration model. The `Control` block has `AccessControl/LocalStation/Control` as qualifier and the `DoorControl` block has `AccessControl/LocalStation/DoorControl` as qualifier. Both objects have `block` as stereotype.

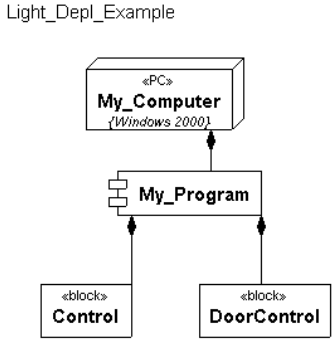

Figure 318 A light integration

––

Tight Integration

In a tight integration, each SDL process instance is put in a task of its own. The number of tasks varies at run-time, as process instances are created and terminated dynamically.

As described in "The View" on page 1709, the deployment diagram shows a static view of SDL systems at compile-time. The thread symbol symbolizes one static thread of execution, which makes it inappropriate for modeling tight integrations. Therefore, the thread symbol is not used for modeling tight integrations. Always attach objects directly to a component when you model a tight integration.

> **Note:**
>
> The Deployment Editor does not prevent you from attaching threads to a component with *Light* or *Tight* as integration model. The check is performed when partitioning diagram data is generated for the Targeting Expert. If you have an invalid diagram, you will get error messages in the Organizer Log.

An example of a typical tight integration is shown in Example 301.

Example 301: A Tight Integration ––––––––––––––––––––––––––––

This example (see Figure 319) shows a tight integration where two processes from the AccessControl system are executing in one component. The component is deployed on a node.

Each process instance of the two SDL processes will execute in an OS task of its own on the target operating system.

The component has *Tight* as integration model (not visible in the figure). DoorController has `AccessControl/LocalStation/DoorControl/DoorController` as qualifier and `process` as stereotype. Controller has `AccessControl/LocalStation/Control/Controller` as qualifier and `block` as stereotype.

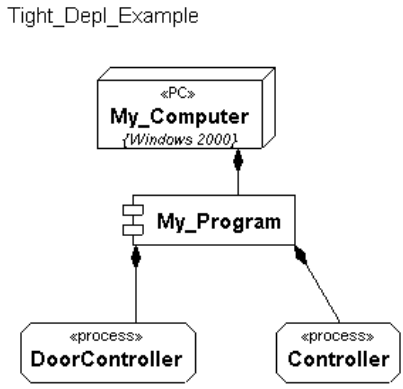

Figure 319 A tight integration

Threaded Integration

Using the threaded integration model, you can map an arbitrary number of SDL objects to an arbitrary number of threads. Each thread symbolizes a light-weight process in the executable file that is generated.

All objects that are connected directly to a component, having a threaded integration, are considered executing together in one implicit thread. Only default values for thread priority, stack size, etc. are available for an implicit thread.

The most common deployment case is that the smallest, indivisible entity is the process instance set. This is the smallest entity that can be graphically represented in a Deployment Diagram. However, it is possible to make each single instance of an object execute in a separate thread in the threaded integration model. This is modeled using multiplicity on the aggregation between a component and the thread that contains the SDL object. Set the multiplicity of the aggregation between the component and the thread to '*' to model this scenario. For a thread with '*' as multiplicity, the values of thread priority, stack size and queue size, etc. will apply to all threads that are created in the target environment.

A complete example of a threaded integration that illustrates the concepts is shown in Example 302.

Example 302: A Threaded Integration ––––––––––––––––––––––––

Figure 320 shows a component with many threads attached. The thread `T` has two objects (`CentralUnit` and `PanelControl`) attached to it. All process instances in these two blocks will execute within one thread. The component should execute on Windows 2000. Thus, Windows-specific thread settings are given. The following values are set: Thread Priority: `THREAD_PRIORITY_ABOVE_NORMAL`, Stack Size: `2048`. The other fields are left empty, as default values should be used.

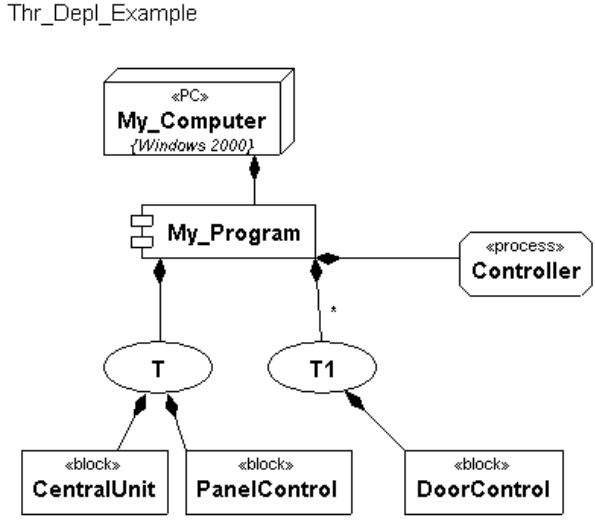

Figure 320 A threaded integration

The `Controller` object is attached directly to the component and will thus execute in an implicit thread. If there were more objects attached directly to the component, they would execute in the same thread as `Controller`. As the thread is implicit, default values are used in the thread settings.

The thread `T1` has '*' as multiplicity in the aggregation between itself and the component. This indicates that all process instances in objects attached to `T1` will execute in a thread of its own. The thread settings for

`T1` are: Priority: `THREAD_PRIORITY_BELOW_NORMAL`, Stack Size: `1024`. These settings will apply to all threads that are created.

All objects have stereotype and qualifier values set as shown in the previous examples.

Deployment Workflow

Drawing a Deployment Diagram

One or many SDL systems can be deployed in the same deployment diagram. However, it is necessary that all the deployed SDL systems reside in the same `.sdt` file.

Perform the following steps to draw a complete diagram:

1. Start the Deployment Editor with a new diagram. See "Starting the Deployment Editor" on page 1706 for instructions.
2. Name the diagram. As the diagram name is used for naming the target directory root, it is important to select a descriptive name.
3. Identify the nodes on which your system(s) will execute. Draw the nodes in the deployment diagram.
4. Name each node and edit stereotype and properties values in the *Symbol Details* dialog box, if desired.
5. For each node, identify the executables. Each executable is symbolized by a component.
6. Draw the components in the deployment diagram and draw aggregations from the node symbols to the components that should execute on them.
7. Set name and edit the integration model for each component.
8. Draw the threads that you wish to use and enter thread settings for each thread.
9. Draw the objects that you wish to deploy. Collect name, qualifier and stereotype information from the Organizer view of your SDL systems and enter this information in each object.
10. Connect each object to a component or thread, depending on the integration model chosen.

Tip:

Let the Organizer window be visible when you edit the diagrams for easy access to SDL object information.

Example 303 shows a deployment scenario where components with different integration models are combined.

Example 303: AccessControl Deployment ––––––––––––––––––––

This example shows a Deployment Diagram for the AccessControl system. The complete system is deployed on two nodes, as shown in Figure 321. On each node there is a component. The component `Prog1` has Threaded as integration model and uses threads. See Example 302 for a more detailed description of threaded concepts.

The node named `Ethernet_Hub` is contained in the diagram to show how the nodes are physically connected to its external environment in a network. As this node has `external` as stereotype, it is ignored when partitioning diagram data is generated. The associations between the nodes are also ignored.

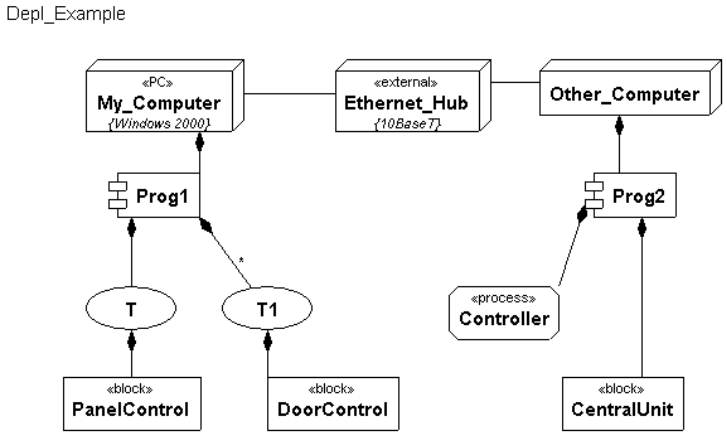

Figure 321 Deployment of the AccessControl system

The component `Prog2` has *Tight* as integration model. The SDL objects are attached directly to the component. Note that the integration model is not visible in the diagram. It is set in the “Symbol Details” dialog box.

––

Tip:

Draw several Deployment Diagrams to experiment with different deployment situations for your SDL systems. This can be useful when you experiment with different thread configurations.

Generating Partitioning Diagram Data for the Targeting Expert

The Targeting Expert uses the Partitioning Diagram Model (PDM) to structure an application that should be built. It uses the node and component concepts from the Deployment Diagram.

PDM files can be generated from Deployment Diagrams using the Organizer. In order to start the Targeting Expert with partitioning diagram data from a deployment diagram, perform the following tasks:

1. Right-click the deployment diagram symbol (`.sdp` file) in the Organizer.
2. Select *Targeting Expert* from the pop-up menu.

A partitioning diagram model is now generated. If there are errors or information messages, the Organizer Log is opened. If the generation passes, the Targeting Expert is started in interactive mode with the partitioning diagram.

The PDM file is saved in the same directory as the `.sdp` file and has the name as the `.sdp` file except for the extension, which is `.pdm`. As the PDM data is saved on file, it can be used by the Targeting Expert both in interactive mode and batch mode.

Note:

When the Targeting Expert is started in interactive mode with a deployment diagram, the `.sdt` file that contains the referenced SDL objects must be loaded into the Organizer. The Targeting Expert cannot be started from a deployment diagram without an `.sdt` file which has been saved.

PDM in Targeting Expert Interactive Mode

To start the Targeting Expert in interactive mode with data from a deployment diagram, you must right-click the .sdp file symbol in the Or-

ganizer and select Targeting Expert. The Partitioning Diagram Model is not visible from the Organizer.

The Partitioning Diagram Model for the deployment diagram is shown in the upper-left corner in the Targeting Expert window. The application is seen on node and component level. For each component, you make build settings. Depending on the integration model selected in the deployment diagram, specific predefined integration settings are available.

For more information on the Targeting Expert batch mode, see "Interactive Mode" on page 2836 in chapter 60, *The Targeting Expert*.

PDM in Targeting Expert Batch Mode

The Targeting Expert can use PDM files in batch mode by either setting the flag `-pdm` or by giving the command `Open-PDM` as a batch mode command. A PDM file from a Deployment Diagram must be generated using the Organizer. After that, the PDM file can be used in either interactive or batch mode.

For more information on the Targeting Expert batch mode, see "Batch Mode" on page 2883 in chapter 60, *The Targeting Expert*.