
# Core Information Model (CoreModel)

# TR-512.A.13 Appendix – Software Examples

Version 1.6

January 2024

ONF Document Type: Technical Recommendation
ONF Document Name: Core Information Model version 1.6

## Disclaimer



Open Networking Foundation
1000 El Camino Real, Suite 100, Menlo Park, CA 94025
www.opennetworking.org



## Important note

This Technical Recommendations has been approved by the Project TST, but has not been approved by the ONF board. This Technical Recommendation is an update to a previously released TR specification, but it has been approved under the ONF publishing guidelines for 'Informational' publications that allow Project technical steering teams (TSTs) to authorize publication of Informational documents. The designation of '-info' at the end of the document ID also reflects that the project team (not the ONF board) approved this TR.

# Table of Contents



# List of Figures



## Document History

| Version | Date | Description of Change |
|---|---|---|
| 1.4 | November 2018 | Version 1.4 (Initial Version) |
| 1.5 | September 2021 | Enhancements to model structure |
| 1.6 | January 2024 | Updated release and dates. |

# 1 Introduction to the document suite

This document is an addendum to the TR-512_v1.4 ONF Core Information Model and forms part of the description of the ONF-CIM. For general overview material and references to the other parts refer to TR-512.1.

## 1.1 References

For a full list of references see TR-512.1.

## 1.2 Definitions

For a full list of definition see TR-512.1.

## 1.3 Conventions

See TR-512.1 for an explanation of:

- UML conventions
- Lifecycle Stereotypes
- Diagram symbol set

## 1.4 Viewing UML diagrams

Some of the UML diagrams are very dense. To view them either zoom (sometimes to 400%) or open the associated image file (and zoom appropriately) or open the corresponding UML diagram via Papyrus (for each figure with a UML diagram the UML model diagram name is provided under the figure or within the figure).

## 1.5 Understanding the figures

Figures showing fragments of the model using standard UML symbols and also figures illustrating application of the model are provided throughout this document. Many of the application-oriented figures also provide UML class diagrams for the corresponding model fragments (see TR-512.1 for diagram symbol sets). All UML diagrams depict a subset of the relationships between the classes, such as inheritance (i.e. specialization), association relationships (such as aggregation and composition), and conditional features or capabilities. Some UML diagrams also show further details of the individual classes, such as their attributes and the data types used by the attributes.

## 1.6 Appendix Overview

This document is part of the Appendix to TR-512. An overview of the Appendix is provided in TR-512.A.1.

# 2 Introduction to this Appendix document

This document provides examples of the use of the CIM software model.

The examples in this document extend the simple examples given in TR-512.12.

# 3 General Examples

## 3.1 Routing 'Process' on a Router

Assume a scenario where an OSPF software process can support multiple OSPF instances.

The software is installed as a single binary (unit of installation).

The device lists the running processes, so we can track the routing software process.

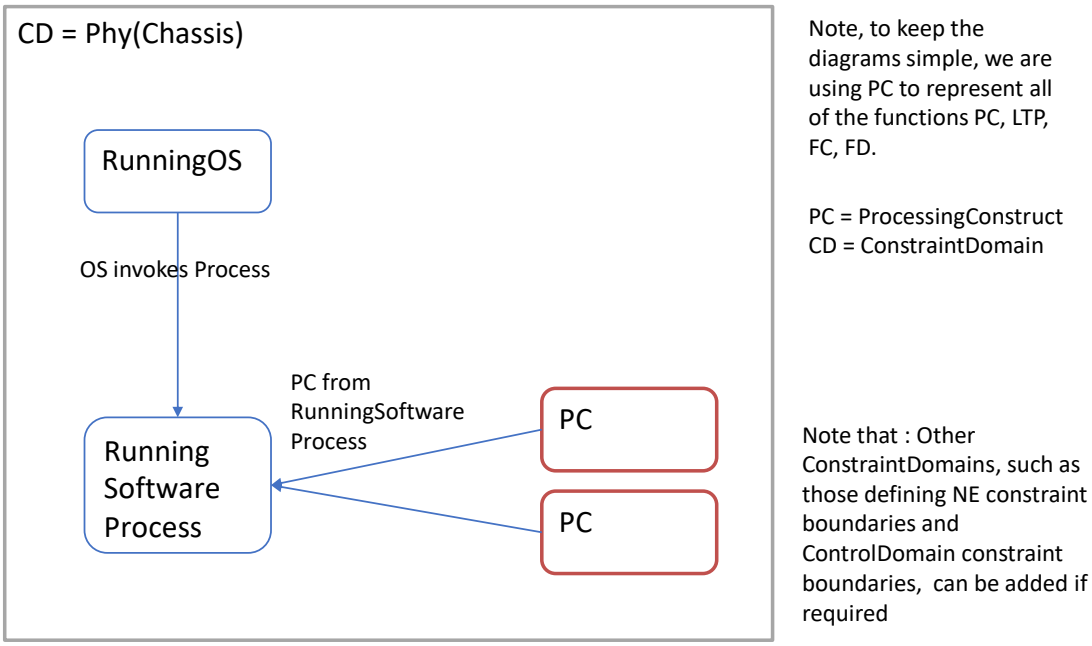



**Figure 3-1 The Routing "process" on router**

Figure 3-2 shows the instance diagram equivalent of Figure 3-1.

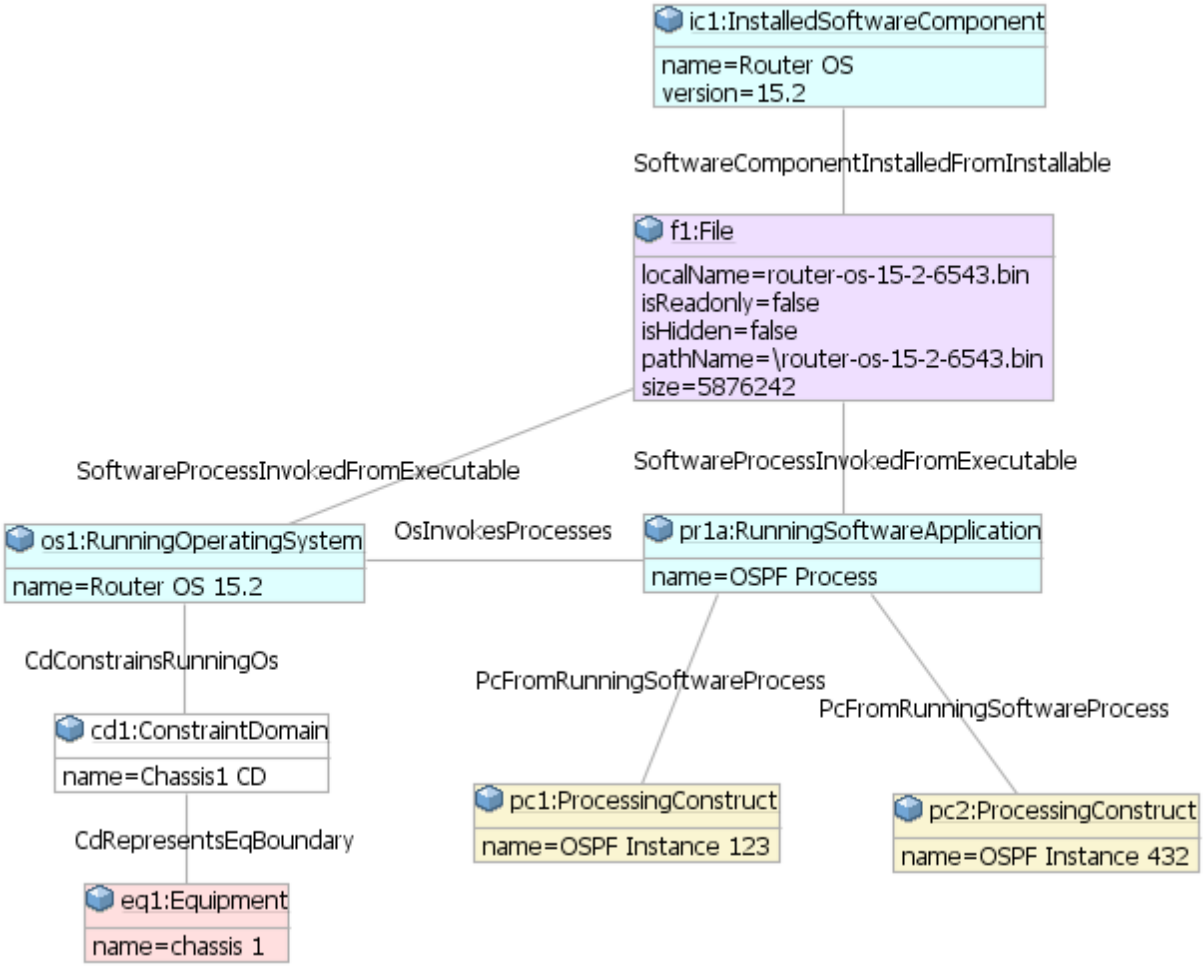



**Figure 3-2 The instance model for the routing "process" on router**

Note that here we are using a ConstraintDomain that is related to a physical boundary for the operating system. Another alternative could be to use an existing 'NetworkElement' boundary or to use an existing ControlConstruct control domain boundary.

## 3.2 Simple Host with Host OS VMM

This is a more complex example because there is also the VMM/VM virtualization layer to represent.

Assume that a host runs a VMM as a process under its operating system (rather than a bare metal VMM). The host operating system is also running other normal software processes. The VMM is running many VMs.

One thing to note is that there are two separate 'namespaces' in this example (which can be conveniently represented using a ConstraintDomain). Both the guest and the host operating systems can have separate files such as c:\word\myDoc.doc with different content, because these are in different FileSystems. Also the guest and the host operating systems can both have separate software processes with the same process id.

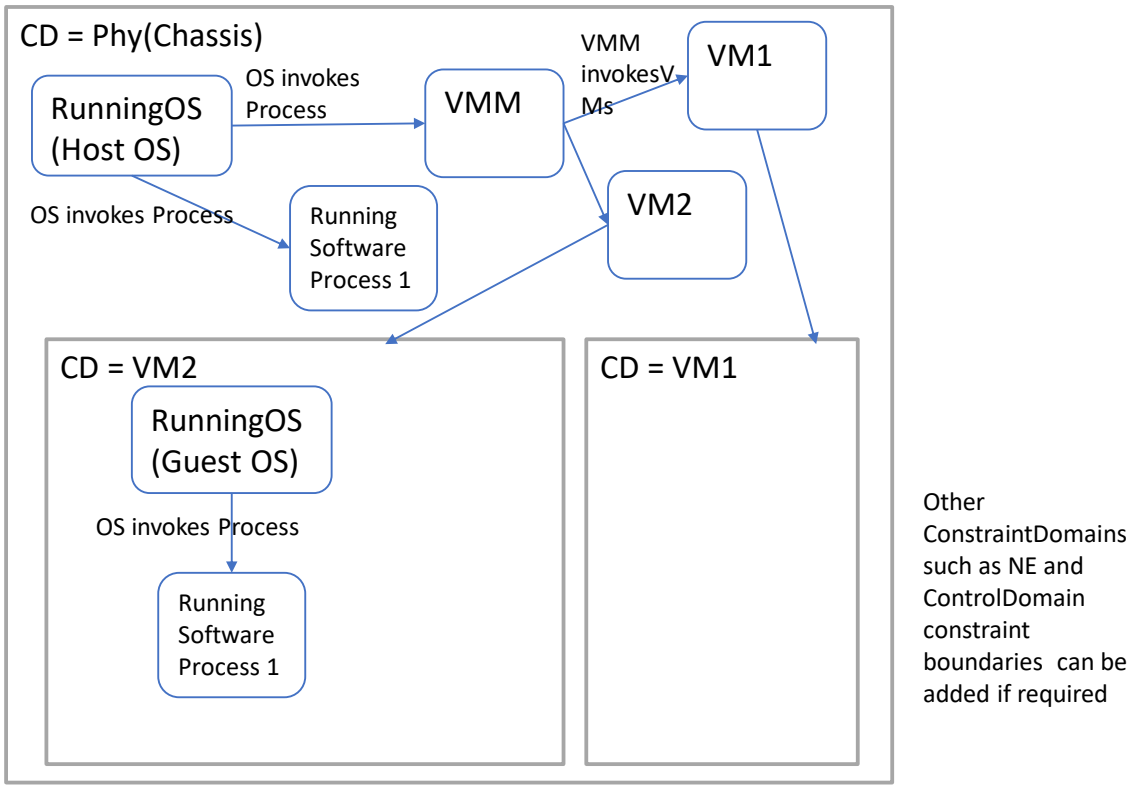



**Figure 3-3 Simple host with host OS VMM**

Note also that because the ONF CIM doesn't currently have a storage model, the FileSystem instance has been directly related to the ConstraintDomain (FileSystemWithinCD). If a storage model is added in the future, then this association needs to be removed and replaced with an association to the storage function.

The control point of view will be explored in a later example, so it has deliberately been omitted from these earlier examples to simplify the diagrams and help focus on the software aspects.

The figure below shows the instance diagram equivalent of the figure above.

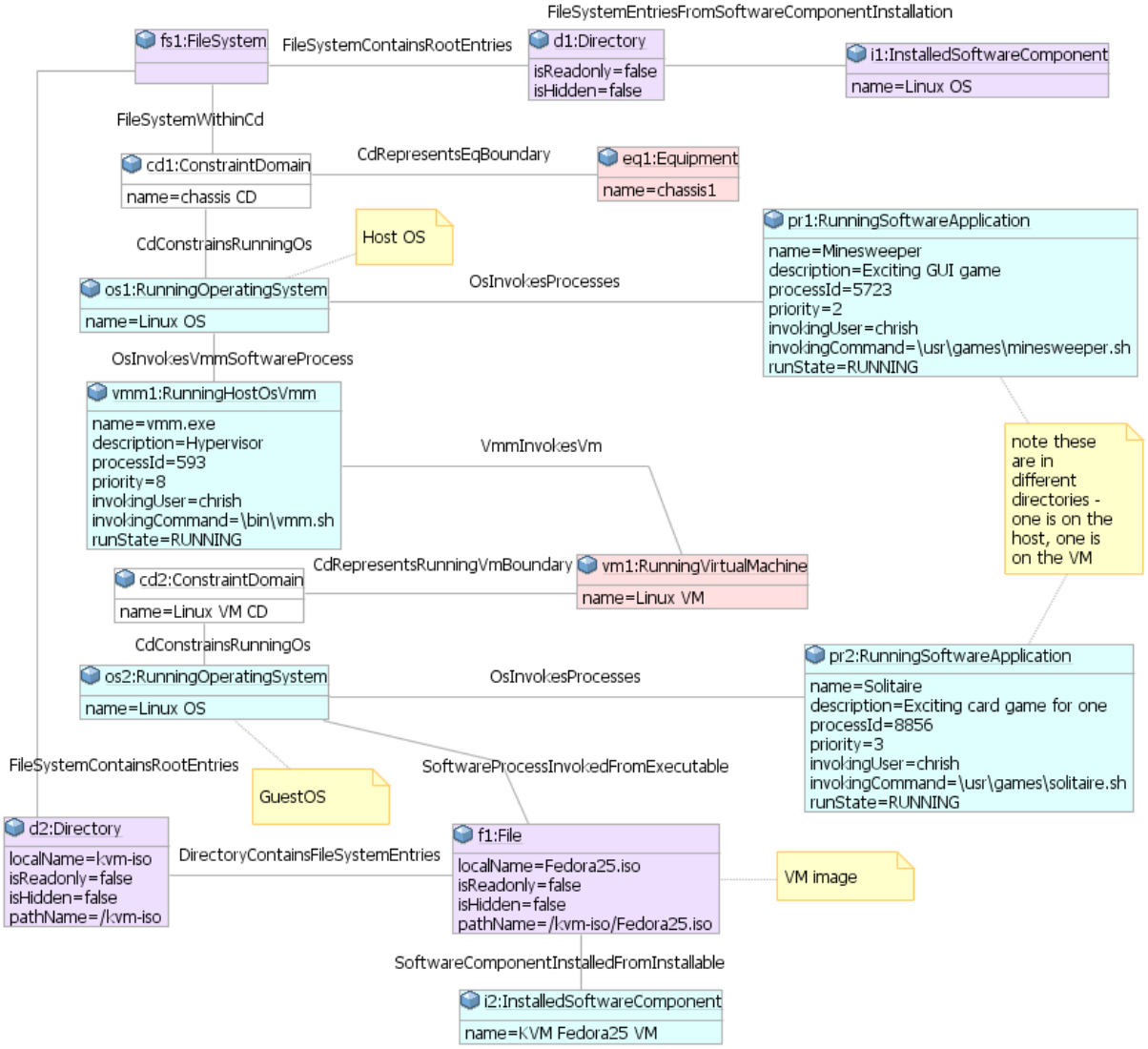



**Figure 3-4 Instance model for simple host with host OS VMM**

Note that there are two operating system instances in the example (host and guest).

Note that here we are using a ConstraintDomain that is related to a physical boundary for the host operating system. The guest operating system's ConstraintDomain is the Linux VM. Another alternative could be to use an existing 'NetworkElement' boundary or to use an existing ControlConstruct control domain boundary.

## 3.3 Simple Host with Container Engine and Containers

This is very similar to the VMM / VM case

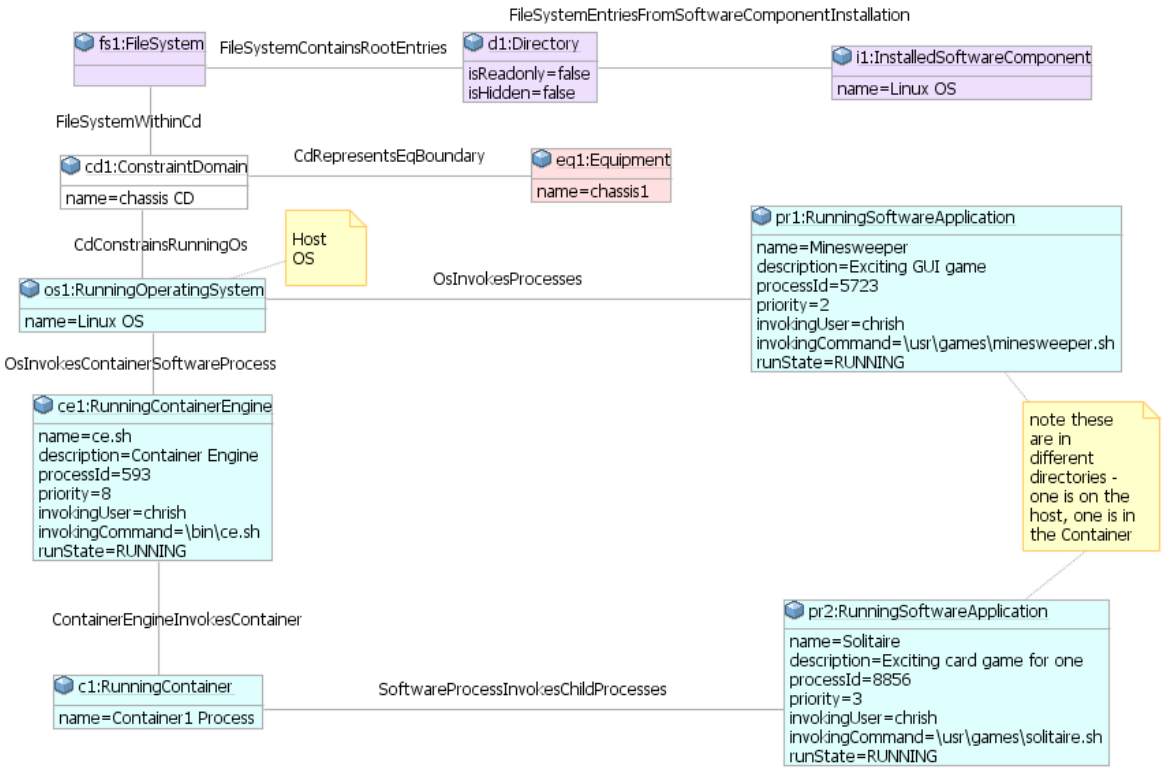



**Figure 3-5 Instance model for simple host with container engine and containers**

Note that there is only one operating system instance in the example.

Note that here we are using a ConstraintDomain that is related to a physical boundary for the operating system. Another alternative could be to use an existing ‘NetworkElement’ boundary or to use an existing ControlConstruct control domain boundary.

## 3.4 CPU, Memory & Storage Example

We now have all of the model concepts that we need to show how we can link software processes to hardware.

Assume that we have a chassis holding compute blades. Each blade is essentially separate, so we can create a constraint domain relating to each physical blade boundary.

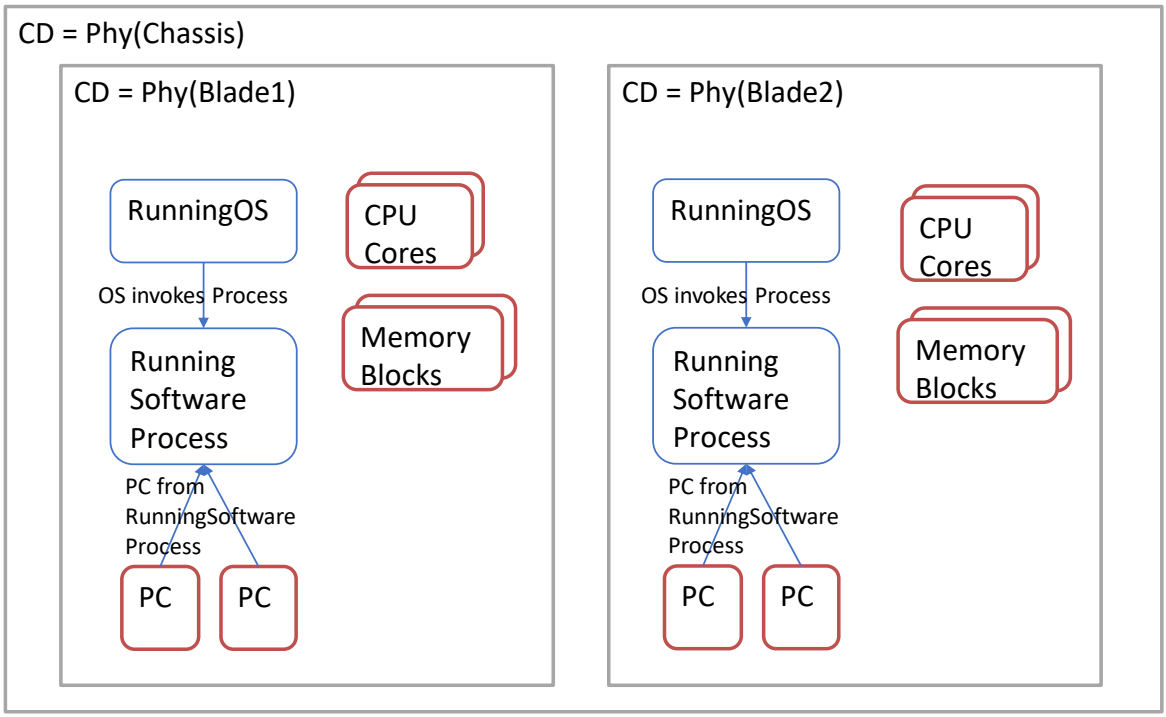



**Figure 3-6 Compute blade in a chassis**

Note that at this stage, we don't have classes to represent CPU or Memory (or Storage) in the ONF CIM, so what we are showing below are just placeholder instances showing how future classes could link in.

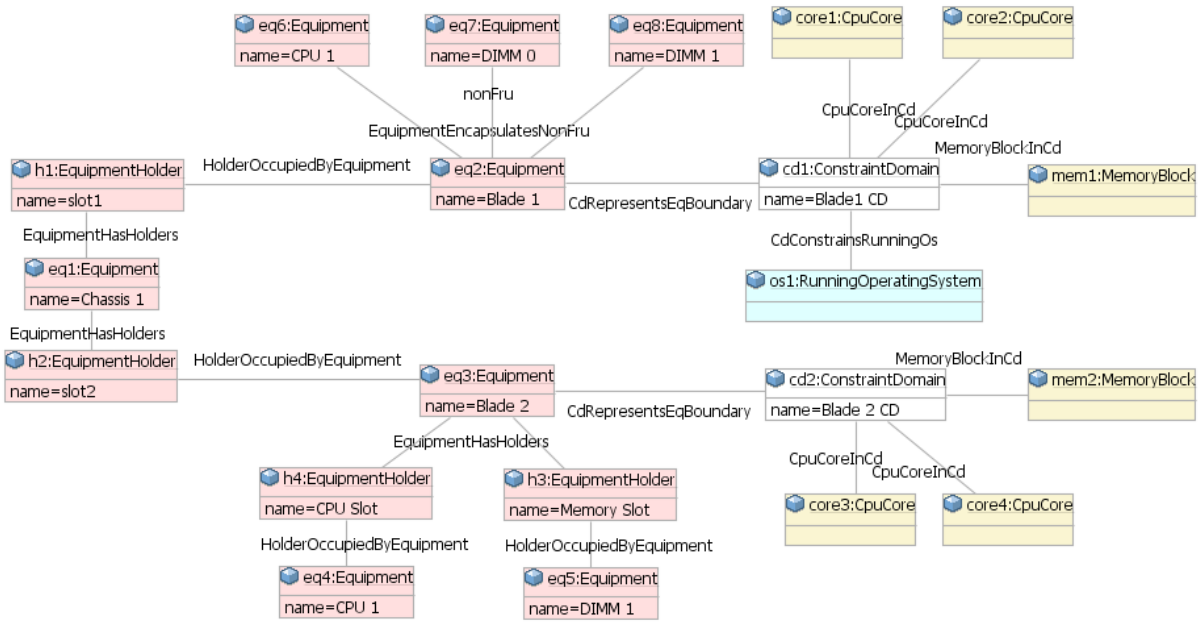



**Figure 3-7 Instance model for compute blade in a chassis**

## 3.5 FPGA Example

In this example we will focus on a single physical blade and then focus on a FPGA within that blade.

There could be many variations in an actual implementation, so this example should be seen as ‘illustrative’ rather than ‘definitive’.

Assuming that our physical unit is a blade server in a chassis, it will be represented as in our previous example.

We then create a new ConstraintDomain for our FPGA and represent the software and functions within the FPGA. Note that the information available from the FPGA may be limited or comprehensive, depending on the implementation.

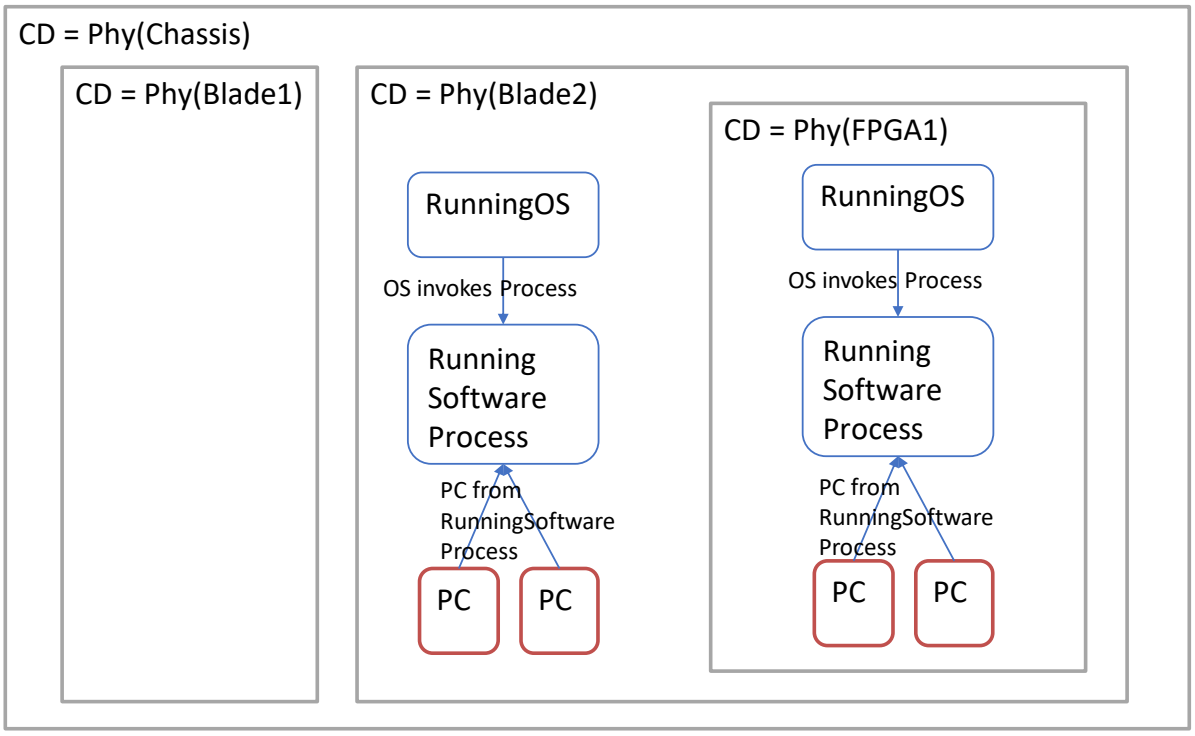



**Figure 3-8 Field Programable Gate Array (FPGA)**

Figure 3-9 shows the instance diagram equivalent of Figure 3-8.

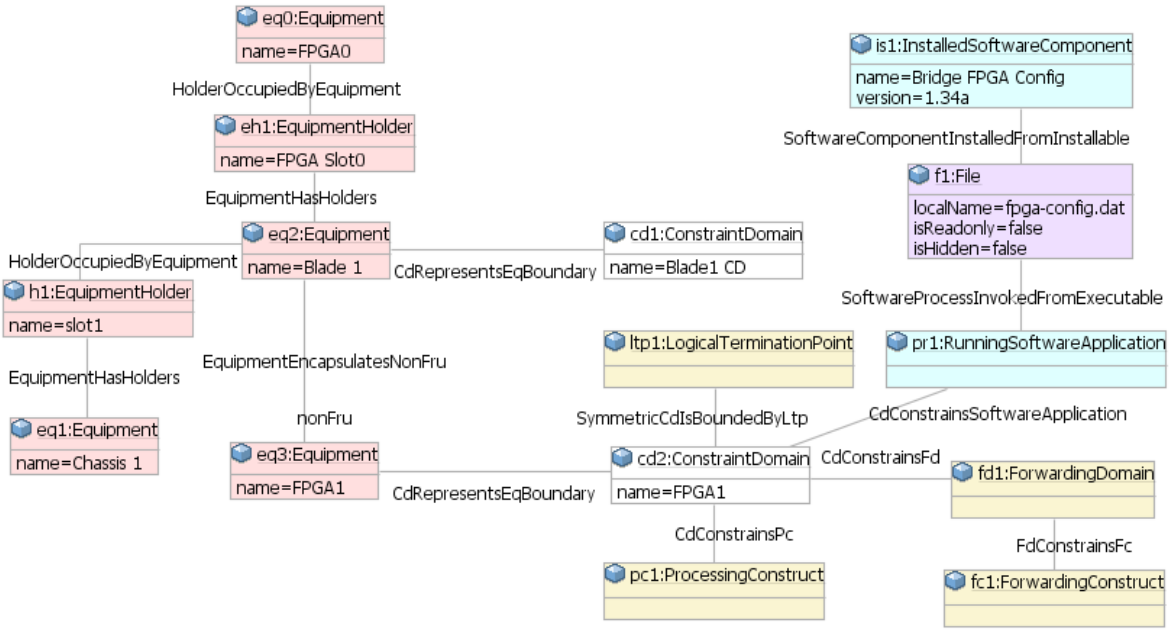



**Figure 3-9 instance model for an FPGA**

## 3.6 Soft Switch Example

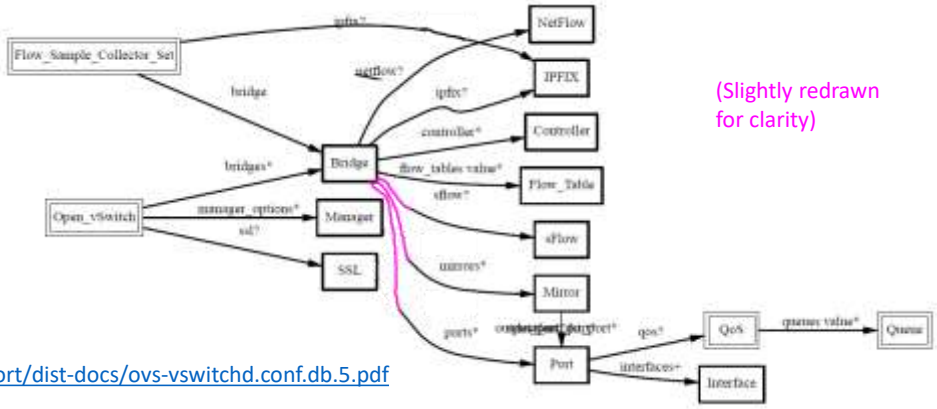
There are a number of soft switches available, both vendor proprietary and open source.

We will use Open vSwitch for our example because it is well known and has useful documentation.

An Open vSwitch could be running directly on a server or via a VM or container. This example shows the direct case, but it can be combined with the previous examples for the other cases.

"Open vSwitch is a multilayer software switch ... Open vSwitch is well suited to function as a virtual switch in VM environments. ... it was designed to support distribution across multiple physical servers."

Diagram from
http://openvswitch.org/support/dist-docs/ovs-vswitchd.conf.db.5.pdf




(Slightly redrawn for clarity)

| Table | Purpose |
|---|---|
| Open_vSwitch | Configuration for an Open vSwitch daemon. There must be exactly one record in the Open_vSwitch table |
| Bridge | Configuration for a bridge within an Open_vSwitch.<br>A Bridge record represents an Ethernet switch with one or more "ports," which are the Port records pointed to by the Bridge's ports column. |

Software Process

ProcessingConstruct

**Figure 3-10 Soft switch example**

We can see from the documentation extracts above that the vSwitch process can support many bridges (the functional switch blocks).

Our block diagram is consistent with our previous examples, but now we are also showing detail within each ProcessingConstruct and the related ForwardingDomain and its ports and the related LTP.

Note that in this case the legacy NetworkElement concept has been used to define the constraint boundary – of course a physical boundary or ControlDomain constraint boundary will also work.

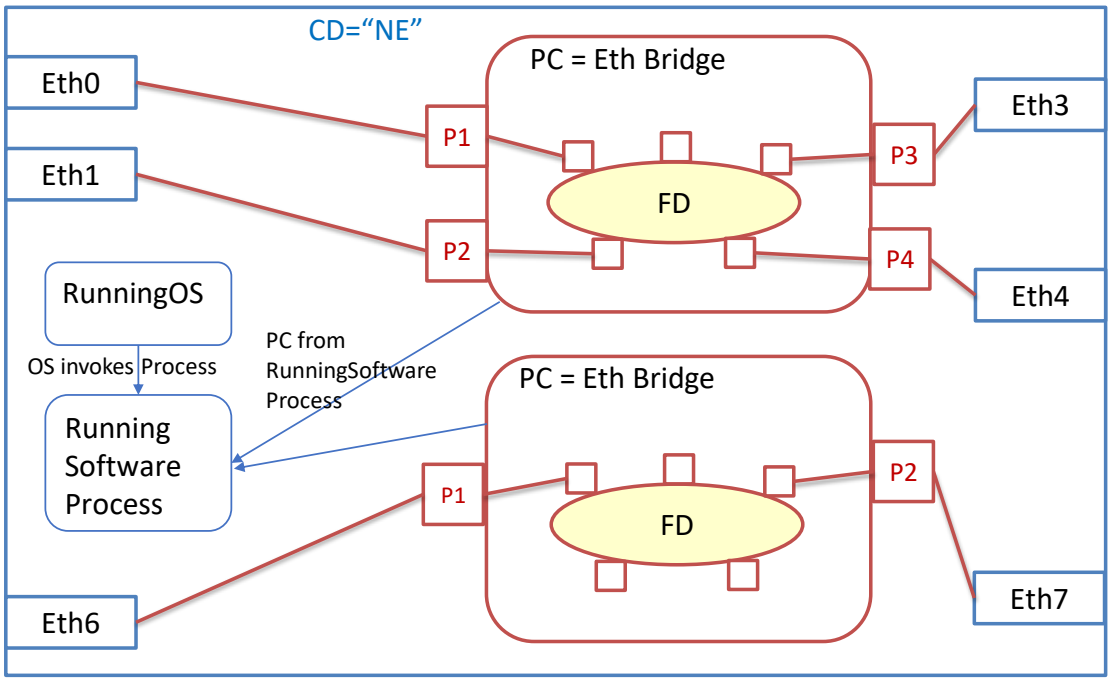



**Figure 3-11 Ethernet bridge**

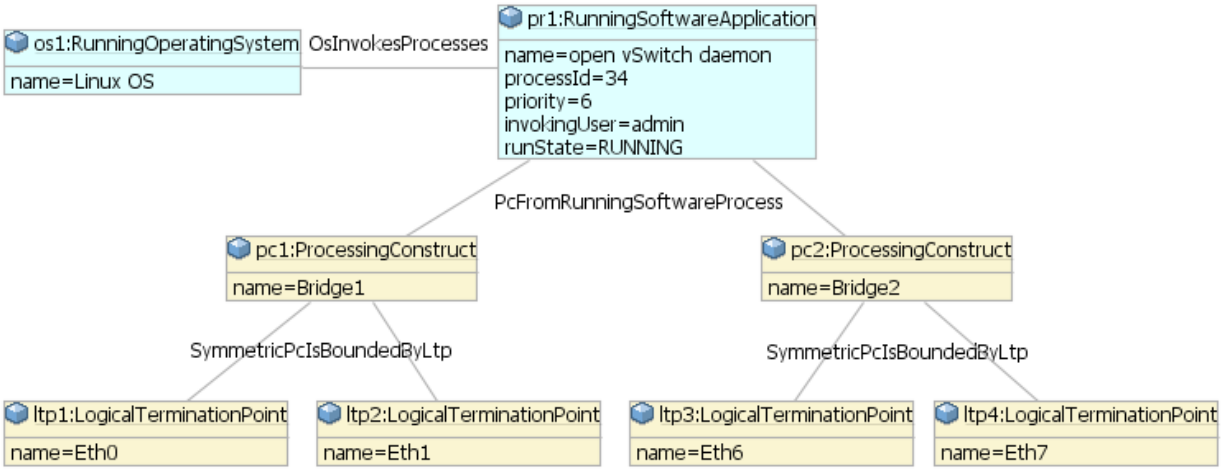



**Figure 3-12 Instance example for Ethernet bridge**

In some cases a port on one bridge and a port on another bridge may go to the same hardware port.

## 3.7 Constraint Domain Example

Now we will look at how the software model aligns with the ControlConstruct concept.

The figure below shows a diagram modified from the previous "Simple Host with Host OS VMM" example.

Adding the control information adds a lot of complexity to the diagram (which is why it has been left off the previous examples).

There may be a number of variations to this example, but a common scenario is where there is:

- a management interface to manage the host operating system
- a separate management interface to manage the VMM and to enable the creation, configuration and removal of VMs (often provided with the VMM software)
- a management interface to manage each of the guest operating systems

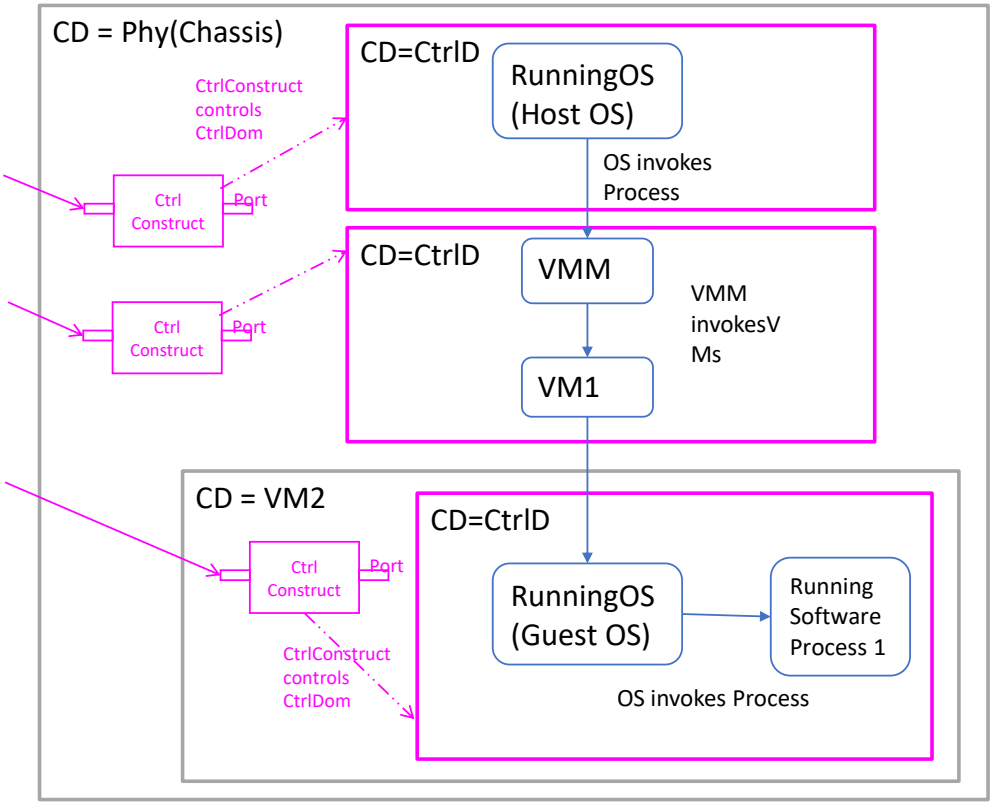



**Figure 3-13 Considering control**

Note that the diagram highlights the information available from each ControlConstruct. The references that cross the ControlDomain boundaries are the ones that a network management system will need to stitch together to provide a consistent end-to-end view.

**End of Document**