

Rapide Tutorial



Table of Contents

[Introduction](#)

[Language Trail](#)

[Example Trail](#)

[Sw Development Trail](#)





[Tools Introduction](#)

[Syntax Reference Manual](#)

Introduction

This tutorial of the Rapide environment is structured in three trails with frequent cross links between them and some supporting modules. The [language trail](#) will take you through a description of the different elements of the language and how they fit together. In the [example trail](#) you will be able to learn by example how to write a Rapide model of your system. In the [Sw development trail](#), you will learn how to apply Rapide to the different phases of software development. As support, a [section](#) describing what tools are available for development in Rapide as well as a detailed evolutionary example describing possible usage of the tools is presented. Also, a [section](#) with the syntax constructs and their meaning is included.

This tutorial draws heavily on existing documentation of the Rapide language and system. It aims to provide a structured guide through it rather than replace it.

The links to the trails will be marked with  for the [language trail](#),  for the [example trail](#),  for the [Sw development trail](#) and finally  for the [syntax reference manual](#).

Language Trail

- [Architecture Construction](#) describes the concept of architecture and how Rapide approaches its description.
- [Conventional programming elements](#): Types, Procedural Language. Rapide has as one of its subcomponents a conventional procedural language that can be used to implement architecture modules. This section is a description of this procedural language and the underlying type system that supports it.
- [Event Model and posets](#): The central part of Rapide is the way it represents computations as Partially Ordered Sets of EventTs. Understanding this section is crucial to understanding the computational model which is the foundation of Rapide.
- [Structure of the process language](#): Events are generated by processes in Rapide, Processes are the unit of sequential execution and also the means of achieving parallelism.
- [Reactive Statements: Rules](#). Rapide offers a rule based approach to describe behavior. Rules are pieces of code that execute when certain preconditions are matched. The execution of a rule may result in some events being generated, or due to the type structure of Rapide where modules, interfaces and connections are valid working types, rules may affect the connection topology of an architecture,

making an architecture behave as a complete set of more "static" architectures.

- [Patterns](#): The left hand side of rules are patterns to be matched. Given that the atomic execution component of a Rapide system is the event, patterns are defined over the set of events an architecture can generate.
- [Advanced Topics](#): Rapide allows its own structural components like interfaces, connections and architectures themselves to be "first class" citizens of its type system. This means that architectures, modules and connections can be created and manipulated at runtime thus opening the door to expressing "families of behaviors" or "families of architectures".
 - [Generators](#): These are the factories where these structural elements can be created in a Rapide system.
 - [Usage of patterns in Architecture Descriptions](#): Patterns also allow very complex connection schemes to be described and expressed very succinctly.
 - [Maps and Constraints](#): Finally, Rapide allows some meta-information about architectures to be expressed as equivalence MAPS between architectures, that allow to asses whether two systems/architectures will behave in the same way, and therefore facilitates the description of "reference architectures" for a whole range of systems. Constraints in Rapide are asserted onto the topology of events a particular system should produce, thereby allowing for "non-prescriptive" specifications of systems, but rather a requirements based specification of them.

Example Trail

The purpose of this trail is to guide the reader through a variety of uses of Rapide, from a simple architecture where the main concepts of Rapide are shown to the usage of sophisticated specification tools like constraints and maps.

- [Architectures, Actions and processes](#): Five dinning philosophers are used to illustrate the basic building blocks of Rapide and how they fit together in a specification that is executable.
- [Connections and interfaces](#): A simple application/resource architecture is used to walk you through the construction of architectures using the Rapide tool set.
- [Architecture Refinement](#): Rapide offers a variety of concepts to support system design at several levels of detail. This example shows how to define a subarchitecture that implements the resource interface declared in the previous example.
- [Parametric Architectures and Pattern Matching](#): One of the most powerful features of Rapide is its ability to describe parametric architectures; architectures where the components and connections are determined at runtime from program parameters. In this example, the application/resource architecture is extended to handle an arbitrary number of resources that are specified by a parameter in the architecture.
- [Constraints and Maps](#): Rapide offers the possibility of defining a "reference" system by the events it can and cannot generate, and then use this definition as a checkable constraint on other system architectures. This example shows how to use the Rapide tools to define a reference architecture, express constraints on that reference architecture, how to interpret another system as if it were an instance of the reference, and check it for conformance with the reference architecture.

Sw Engineering Trail

Rapide is a language designed to support the development of large distributed and concurrent systems. This trail shows how Rapide can be used in different phases of the life cycle of a system. Rapide is mainly oriented towards the design and validation of systems, rather than towards the implementation phases. Accordingly, this trail is concerned with system [top level design and analysis](#), [design and rapid prototyping](#) and [test](#)

[generation and execution](#)

Syntax Reference Manual

This section of the tutorial is a reference manual for Rapide language syntax, where specific format of statements is given, the covered topics are:

- [Architectures](#)
- [Interfaces](#)
- [Actions and functions](#)
- [Services](#)
- [Procedural Statements](#)
- [Reactive Statements](#)
- [General Reactive Rule Form](#)
- [Module Generator](#)
- [Constraints](#)
- [Maps](#)