POLYCHRONY
A TOOLSET FOR SIGNAL

Polychrony Software Development Plan

V1.0

	Author(s)	Checked by	Approbation
Name	**Members of the Espresso Team**	**Loïc Besnard ** Thierry Gautier**	**Jean-Pierre Talpin**
Company	**INRIA**	**INRIA, **CNRS**	**INRIA**
Department	**ESPRESSO TEAM**	**ESPRESSO TEAM**	**ESPRESSO TEAM**
Date	**November 2011**	**November 2011**	**November 2011**
Visa			
Summary	**This document presents the organization of the POLYCHRONY tool (Signal ToolBox, Signal GUI, SSME Platform) and its development.**		

Table of Contents

1 Preface

1.1 Table of versions

Version	***Date***	***Description & rationale of modifications***	***Sections modified***

1.2 Table of references and applicable documents

Reference/ Applicable	***Reference***	***Title & edition***	***Author or editor***	***Year***

1.3 Acronyms and glossary

Term	***Description***

2 Subject

2.1 Purpose of the document

This document is the development plan for the tool Polychrony.

It defines the organization and the management of the project.

More precisely, the SDP presents :

- the project objectives and assumptions
- the project organization and the role of the participants
- the expected products
- the roadmap
- the project management including the planning, the reporting, the risks management ...

2.2 Editing particularities

2.2.1 Changes identification

All the changes made since the previous publication are identified using the sign | in the left margin of each line holding a modification.

2.2.2 Temporary editing

Special points are signaled like this :

. ***temporary***

. ***incomplete***

. ***to be defined***

. ***to be confirmed***

.***TODO***

2.3 Application scope

This document is applicable for the tool Polychrony.

2.4 Edition and evolution of the document

2.4.1 Responsibilities

Author

The development plan is written by the members of the Espresso Team (INRIA Bretagne Atlantique-IRISA, Rennes).

Authorization

The development plan will be authorized by Loïc Besnard and Thierry Gautier.

Approval

The development plan will be approved by Jean-Pierre Talpin.

Diffusion

The development plan is provided to each member of the development team for application.

Application

The application of the development plan is ensured by the quality responsible of the team.

2.4.2 Evolutions

The members of the Espresso Team (INRIA Bretagne Atlantique-IRISA, Rennes) are responsible of the evolution of the document.

This document shall be modified when :

- The information relative to the project change (organization, context, objectives ...)
- The process is modified in order to improve it or to reflect the reality.

3 Context & Objectives

The **POLYCHRONY TOOLSET**, is an Open Source development environment for critical/embedded systems. It is based on SIGNAL, a real-time polychronous dataflow language. It provides a unified model-driven environment to perform design exploration by using top-down and bottom-up design methodologies formally supported by design model transformations from specification to implementation and from synchrony to asynchrony. It can be included in heterogeneous design systems with various input formalisms and output languages.

3.1 SIGNAL language

SIGNAL is a specification and programming language for critical/real-time embedded applications. The main features of the SIGNAL languages are synchronized flows (flows + synchronization) and processes: a process is (recursively) a set of equations over synchronized flows describing both data and control. The SIGNAL formal model provides the capability to describe systems with several clocks (polychronous systems) as relational specifications. Relations are mandatory to write partial specifications, to specify non-deterministic devices (for instance a non-deterministic bus), and to abstract the behavior of external processes (for instance an unsafe car driver). Using SIGNAL allows to specify an application, to design an architecture, to refine detailed components down to RTOS or hardware description. The SIGNAL model supports a design methodology which goes from specification to implementation, from abstraction to concretization, from synchrony to asynchrony.

More details can be found in a short introduction to SIGNAL language on the Polychrony web site.

3.2 POLYCHRONY TOOLSET

The POLYCHRONY TOOLSET provides a formal framework:

- to validate a design at different levels, by the way of formal verification and/or simulation
- to refine descriptions in a top-down approach,
- to abstract properties needed for black-box composition,
- to assemble heterogeneous predefined components (bottom-up with COTS).
- to generate executable code for various architectures

The POLYCHRONY TOOLSET (See Illustration 1), contains three main components:

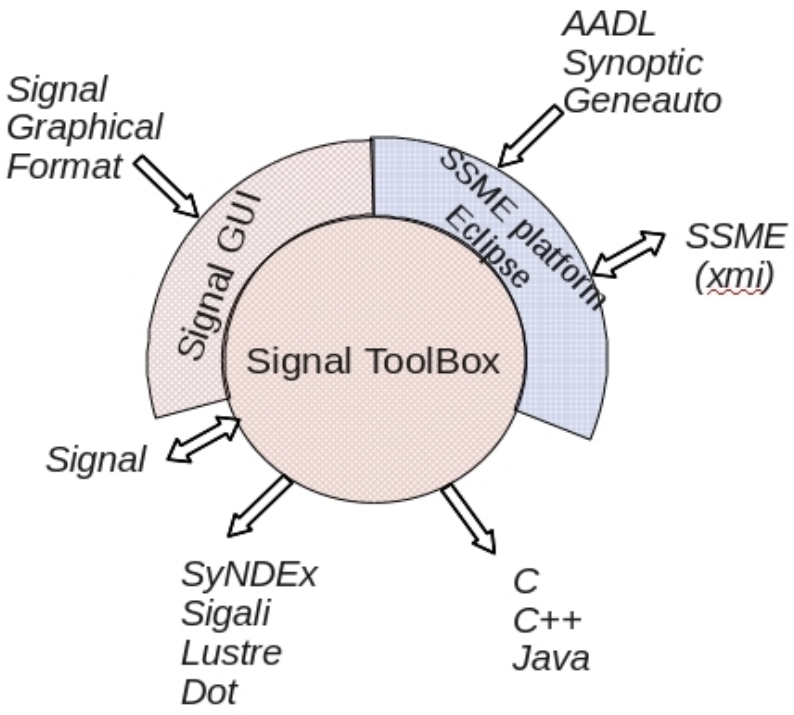
- The SIGNAL TOOLBOX, a batch compiler for the SIGNAL language, and a structured API that provides a set of program transformations. The SIGNAL TOOLBOX can be installed without the other components.
- The SIGNAL GUI, a Graphical User Interface to the SIGNAL TOOLBOX (editor + interactive access to compiling functionalities). SIGNAL GUI requires the SIGNAL TOOLBOX (or an other component that redefines the SIGNAL TOOLBOX APIs).
- The SSME PLATFORM, a front-end to the SIGNAL TOOLBOX in the ECLIPSE environment. SSME PLATFORM requires the SIGNAL TOOLBOX (or an other component that redefines the SIGNAL TOOLBOX APIs). SSME stands for Signal Syntax Model under Eclipse.

The POLYCHRONY TOOLSET, also provides:

- libraries of SIGNAL programs,
- a set of SIGNAL programs examples,
- user oriented and implementation documentations,
- facilities to generate new versions.

The SSME PLATFORM may be used to import other formalisms (AADL, Synoptic, Geneauto). These translators are not described here.

Illustration 1: Polychrony Tool set



3.3 Objectives

The integration of Polychrony in the Polarsys platform consists in the integration of the SSME platform, inside an **Eclipse** environment. The Eclipse plug-ins of the SSME platform correspond to:

- the definition of the SSME meta-model,
- a reflexive editor for editing SSME programs,
- a Signal textual editor,
- the compilation scenarios that allows to apply the functionalities defined in the SIGNAL TOOLBOX,
- the connection to the SIGNAL TOOLBOX,
- on-line documentation

4 Hypotheses & constraints

******incomplete******

The SSME platform depends, for the production of a new version, on the availability of

- the SIGNAL TOOLBOX,
- the TopCased tools which automatically generate editors for specific languages based on their meta-model.

5 Roadmap

incomplete

The first paper published on the Signal language was in 1983. The (current) SIGNAL TOOLBOX has been developed during some European projects such as the SACRES (1995), and SafeAir projects, among others. The SSME platform has been developed during the OpenEmbDD (2006-2009), TopCased(2007-2010) and SpaCiFY(2007-2010) projects, OPEES (2010-), CESAR(2010-).

The development of the tool(s) are defined inside some research projects. The milestones are induced by the projects.

6 Licences

Polychrony toolset components are distributed under the following licenses:

- The SIGNAL TOOLBOX is distributed under GPL V2 licence.
- The SIGNAL GUI is distributed under GPL V2 licence.
- The SSME platform is distributed under EPL licence.

7 Project organization

The members of the Espresso team can be found at the following address:
http://www.irisa.fr/espresso/membres

8 Inputs / outputs

8.1 Inputs

The development of the (current) version of the SIGNAL TOOLBOX began approximately in 1995 (SACRES project). Consequently, we may say that its current status depends on the following input documents

- The Reference manual of the Signal Language.
- Research reports
- PhD documents

As regards the Eclipse interface of the Polychrony tool for TopCased, its development began around 2006. The inputs to this development is to conform with Eclipse architecture, and the Signal language reference, by means of an Ecore meta-model (and its reflexive editor) called SSME, which will allow model to model transformations (i.e.

From Signal meta-model to other language meta-model), and providing a seamless interface to the SIGNAL TOOLBOX services.

8.2 Outputs

The different phases of the development of a new release of SIGNAL TOOLBOX starts with the definition of a new functionality or the definition of a new feature defined in the Signal reference manual. We have not a fixed time cycle for such a development. The new functionality induces its integration in the SSME platform (a new functionality is available for the batch compiler/the SIGNAL GUI/the SSME platform).

Specification, Design description, Source code

Currently the specification and the design are documentations associated with the source code.

Tests Plan, Test results

The tests are applied when a new release is delivered. For the application of the functionalities, the “batch compiler” is used. They are applied using scripts and the results produced must be (sometimes) revised by a human tester.

****TODO***: automatic synthesis of the results of a part of the tests (results of a simulation)****

Delivery

When a release is ready and the tests are satisfied, the release can be delivered with

- the User Guide
- the Installation and Administration Guide
- the release note
- the package (for the delivery)

Reporting ****TODO****

- record of the quality controls
- reporting

9 Strategy of Verification and Validation

Signal ToolBox

A set of tests are available for the Polychrony kernel. They can be applied using the “Batch compiler”. A test depends on the tested functionality: recall that SIGNAL TOOLBOX provides transformations, optimizations, formal verification, abstraction, separate compilation, mapping, code generation, simulation, temporal profiling....

Different test targets are considered:

- the elementary tests: each class of the operators of the Signal language are considered
- the composition of several operators

- the fixed bug programs which are added in the benchmark suite
- the examples described in the Signal reference Manual
- the applications provided as examples in Polychrony distributions

For a new feature, some tested examples are provided with the distribution.

This tests are applied for each new release. The trace of the tests can be kept. The tests can be done on different systems (Linux, Solaris, Windows, Mac OS (Intel and PowerPC architectures)).

Note that it is not always possible to generate code for simulation for correct Signal programs. In this case, the tester must look at a new generated Signal program (resulting of the application of functionalities). The tester must know very well the Signal language and its semantics.

Eclipse Plugins

Test of the Eclipse Interface for Polychrony is through a set of examples provided with the distribution, as well as integration tests for the three platforms for which Polychrony is provided. Those are Linux, Windows and Macintosh.

Integration tests are followed by tests with examples considering editing, loading of models and randomly generated compilation scenarios.

10 Documentation management

10.1 Documentation identification

No rules are used.

There are the user documents provided for the SIGNAL TOOLBOX services and the user documents provided for the SSME platform.

The documentations about the Signal language, the kernel services are provided in the website of the Espresso team http://www.irisa.fr/espresso/Polychrony/. They are updated with a new release.

The documentation for the source of the kernel services is provided with the distribution. The version of the documentation is tagged with the version of the Polychrony kernel.

10.2 Editing rules

Documentation for the two tool parts, the SSME platform under Eclipse and the SIGNAL TOOLBOX services, is done in two apparently different ways.

For documenting the Polychrony services permanent members of the Espresso team are in charge of describing the features and functionalities of this part. It is an activity that has evolved since 1995 of joint work and has been largely revised and improved.

By contrast, documentation of the Eclipse interface does not require such a rigorous editing process, because it only describes what is already provided by the SIGNAL TOOLBOX services from the point of view of a specific IDE, that of Eclipse. The editing

here is done by the developers of the Eclipse interface, there is no group editing here.

For the research reports, the leader of the team must authorize their publication.

10.3 Templates

Templates for research reports are those defined at INRIA Rennes Bretagne Atlantique/ IRISA.

11 Project management

11.1 Planning

The development of the tool(s) are defined inside some research projects. The milestones are induced by these projects.

11.2 Progress management

The development of the tool(s) are defined inside some research projects. The progress depends on the meeting progress of these projects.

11.3 Risks management

List of the risks (***incomplete***)

- No continuation of the Espresso project. It depends on the result of periodic evaluations.
- Departure of the developers. Some work is realized by non permanent team members. This is not the case for the SIGNAL TOOLBOX developers.

*11.4 Reporting****incomplete****

The members of the Espresso team publish the results of their work in conferences, research reports or PhD theses. Moreover, the different developers report of the evolution of their work during meetings. Demonstrations of the tool(s) are also performed at conferences.

12 Specific tools and standards

The following tools are used for the development and management of the SIGNAL TOOLBOX:

- cmake, used for compiling and installation
- doxygen, used for the management of the documentation