PRAGMADEV
real time development tools

For the very first time all these languages are combined in one consistent environment : **Real Time Developer Studio** in conformance with SDL-RT specification.

Real Time Developer Studio

The best of all worlds at your fingertips

Increasing complexity of embedded software brings source code to reach an incredible number of lines. Graphical abstraction is needed while still keeping legacy code and years of experience in C or C++ source code. Recent experiments have shown there is no graphical language that does it all, but some representations are best suited to describe some parts of the system. Based on this experience, SDL-RT[1] has been defined to keep the good old source code and build on top of it a graphical abstraction based on industry proven standard languages :

- **SDL[2] to describe the architecture and the dynamic aspects of a system,**
- **UML[3] to describe static objects and their relationships,**
- **C for real time parts needing precision and high performance,**
- **C++ for static shared code.**

a Standard object oriented graphical representation

Because using graphical representation is about communicating with the other members of the team, with a customer, or a sub-contractor; the graphical representation is based on standards and precise enough to avoid ambiguity.

Since very few projects start from scratch, re-usability is a top priority feature in order to integrate legacy code and to write re-usable components.

b From specification to target

The development process from specification and design down to integration on target does not suffer from gaps or inconsistency. Specification is up to date with the design to ease round trip engineering.

c Self documented and legible code

Software documentation is up to date from high level specification to final code and is available to the whole organisation.

d Keep control

When it comes to design, the real time designer keeps control of what the tool does and how, because embedded software is often an optimisation problem.

e Ease testing

The test environment is automatically set up. And whatever the type of testing, documentation is easy to produce.

f Integration with third party tools

The development environment is open to access all third party tools needed at some point during the development process.

g Secure your investment

Nowadays, nobody has the time to verify all aspects of investing in a tool or a technology and history has shown even buying from the market leaders might lead to a disaster. Still, a number of items can be checked prior to decision making to secure the investment.

Features & Details

Innovative concept

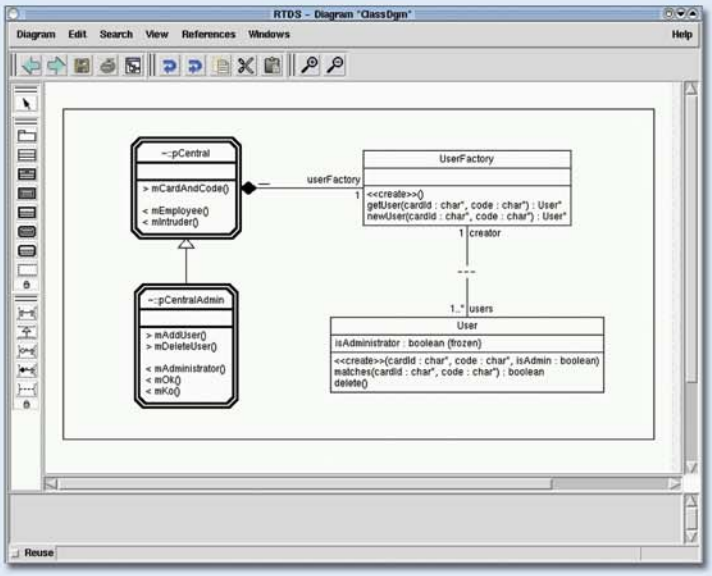
a

SDL-RT combines UML and SDL

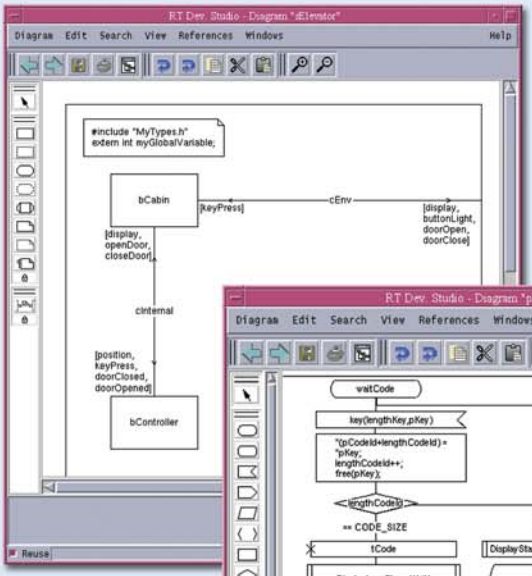
SDL-RT is the very first graphical language that combines well known object oriented graphical languages UML and SDL embedding C and C++ for the development of embedded software.

UML class diagram

Business **libraries** are defined with the class diagram and organized in packages. Traditional UML classes are preferably used to define static objects and SDL classes are preferably used to define dynamic objects such as processes or functional blocks.



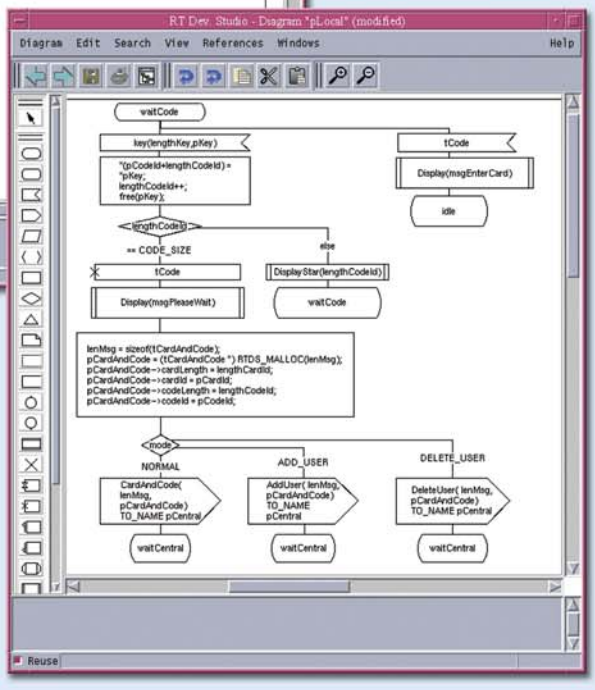
The active class pCentralAdmin (task) inherits from pCentral and uses 1 instance of UserFactory (C++) class.



Functional blocks define the architecture of the system, channels define the interfaces.

SDL block and process diagrams

Architecture and **interfaces** are defined with the SDL block diagram where instances of classes defined in the UML class diagram are plugged in. SDL-RT editors are based on an easy to use, intuitive interface including : copy/paste, unlimited undo/redo, automatic symbol insertion, syntactic and semantic verification, on-the-fly contextual symbol syntax help.

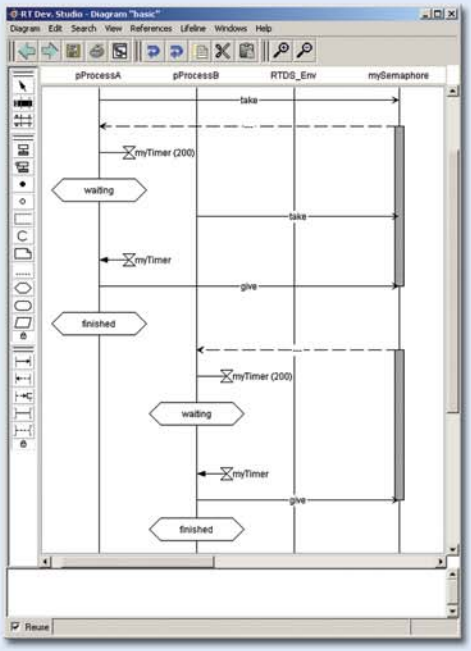


Process pLocal in waitCode state can receive key or tCode messages. A test on mode will lead to different message sendings.

MSC / Sequence diagram

MSC (SDL Message Sequence Chart) or sequence diagram (UML) representation provides a detailed description of the **dynamic** behaviour of the system. It can be used to specify a behaviour or to trace the executed system. Each component of the system such as tasks, semaphores, or C++ objects is seen as a line on the diagram where time flows from top to bottom. Key events in the system have a graphical representation such as: internal states modification, message inputs and outputs, timers, semaphore manipulations, tasks creations or deletions…

MSC diagrams can be organized using the SDL HMSC diagram or the UML use case diagram.



An SDL-RT MSC dynamic view

Consistent tool chain

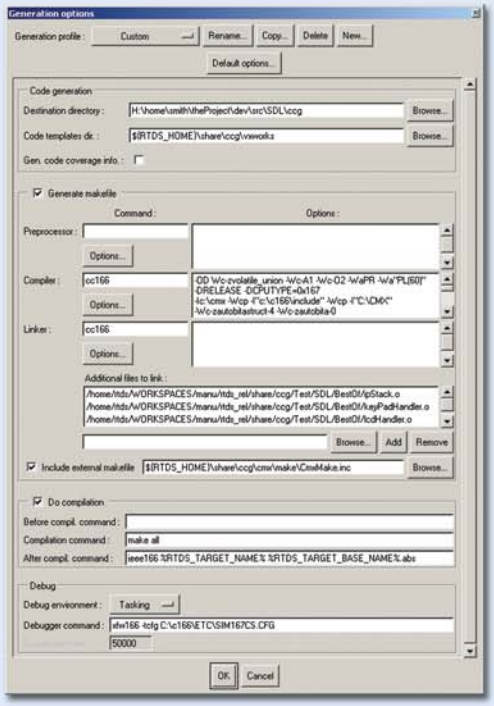
b Specification and design

The same language is used all through the development process from specification to design. The only difference is how detailed or enriched the model is.

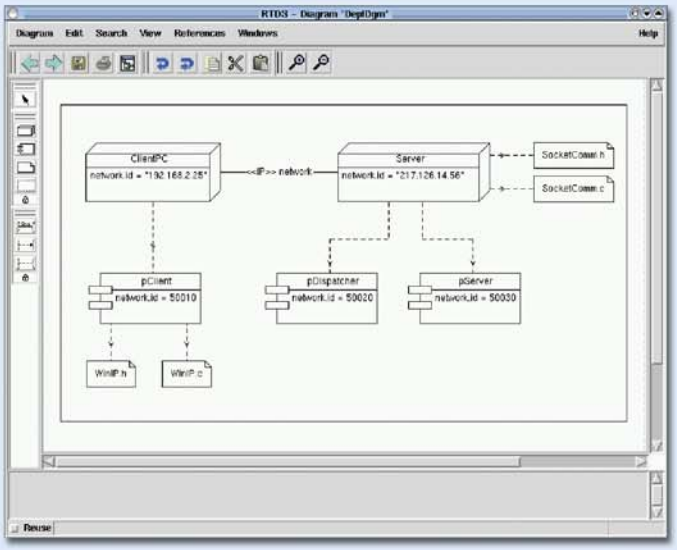
Specification can be validated on host, and design can be debugged on target. In both cases, the graphical link is never lost.

Code generation

The RTDS code generator generates macro based ANSI C code from the SDL behaviour description, and C++ stubs from UML class diagram. An associated **generation profile** translates the C macros to the desired RTOS but the generated code is always the same, only the macros definitions change. Adaptation to major RTOS is delivered as source code allowing customisation on any RTOS.



A generation profile example with CMX RTOS and Tasking debugger



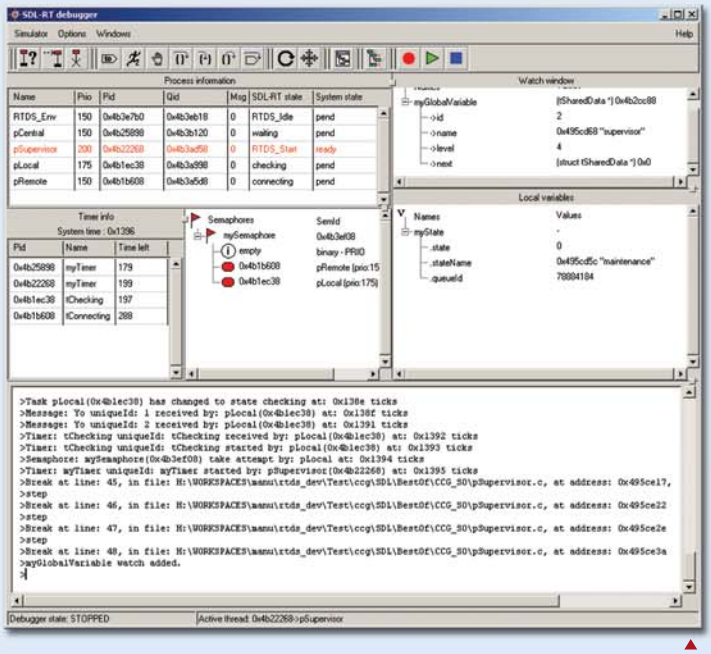
Distributed systems

The overall SDL-RT system can be distributed on several nodes, components, and executables. Distribution is specified with the UML **deployment** diagram and code generation will automatically generate the necessary information to handle communication between nodes, components, and executables.

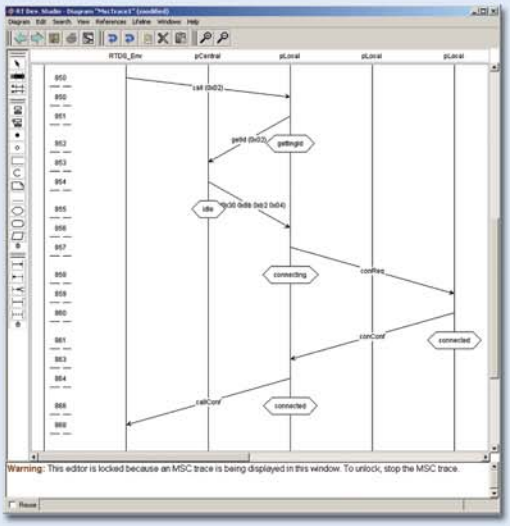
Process pClient running on ClientPC node communicates with process pDispatcher running on Server node via IP. Nodes and components attributes define the IP address and port number.

SDL-RT debugger

SDL-RT description can be simulated with an RTOS simulator or debugged on target. Debugging is based on an underlying classical C debugger or cross-debugger but all the information is displayed **graphically** at SDL-RT level. Three levels of debug allow to run the system: until the next SDL-RT event, step by step graphically, or step by step in the generated C code. A live MSC trace allows to keep track of the past events.



A simple debug session



A live debug trace with system time information

Open environment

c Automatic documentation generation

SDL-RT description is so clear it is often useless to add explanations around it. Still RTDS offers several ways to integrate SDL-RT description in a word processor.

HTML documentation generation

The whole project including UML diagrams, SDL diagrams, C and C++ code can be exported in HTML format. Navigation links between the diagrams are kept to easily browse the system.

Publish / Subscribe

The Publish / Subscribe feature allows to export whole or a part of a diagram, and to update the exported graphics whenever it is needed in a single operation. Documentation in the word processor is then automatically updated.

d Customisation

It is a top priority feature in RTDS to let the user control and customize what the tool does. In the end, there are no nasty surprises when it comes to memory size, performance, or behaviour on target.

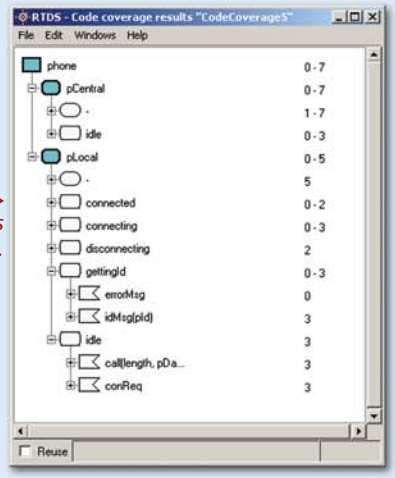
Macro and brick based code generation

RTOS adaptation is done through C macros making any RTOS integration possible. Generated code is also based on customisable C bricks making the hard coded part of the generated code minimal. The C code written in the SDL diagrams is left 'as is' making the generated code very legible. This guarantees target debugging at C level will not give you headaches even if done without the tool. Last but not least, it is possible to navigate back and forth between the generated code and the SDL-RT graphical source.

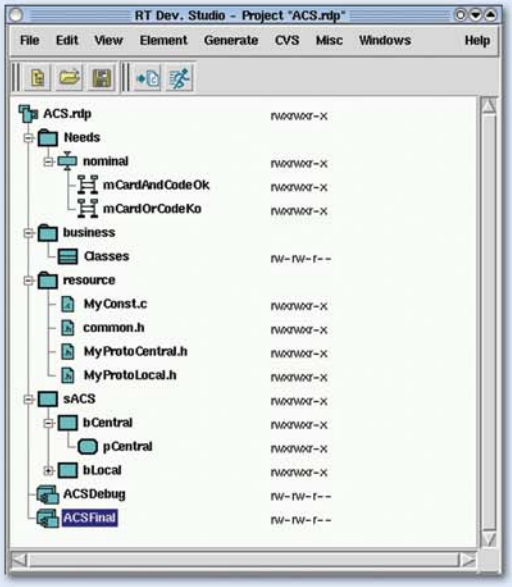
e Requirements validation

RTDS includes all major features to validate and test a system :

- requirements verification,
- non regression testing,
- automatic test generation,
- code coverage.



Untested code is graphically identified.



f Open and integrated environment

RTDS provides several ways to integrate third party tools, and RTDS can be integrated in a larger development environment.

Project manager

The Project manager is the hub of the tool. It gathers all the files parts of the project, makes the link with the editors, generates the code, and starts the debugger. The Project manager supports user-defined external file types and associated external tools are automatically started when opened. Each element being a separate textual file, integration with version control tools is straightforward and transparent. User-defined menus can be added to the Project manager to call external tools such as configuration management tools.

◄ *The Project manager provides a legible view of the project*

g Make the right choice

Cheapest tool on the market

Real Time Developer Studio is among the cheapest tools on the market. That is because we are a highly qualified small development team working with state of the art technology willing to spread the technology to a mass market.

No hidden cost

Our business model is 100% product based. It is not of our intention to try to sneak in any training, coaching, or consulting once you have bought the product.

Instant kick off

SDL-RT approach is so natural that an experienced real time developer does not need any training to get up to speed. Still, if the team has no experience in SDL nor in UML PragmaDev can provide trainings, coaching and consulting.

No technical risk

At any time, it is possible to step back without losing what has been done because you can either re-use the storage format with any XML parser or the very legible generated C code.

Partnership

PragmaDev is a product oriented company that has established a network of partners to offer all kinds of services around its product such as :

- Training,
- Tool customisation,
- Consulting,
- Developments.

It guarantees quality since all partners are specialists in their respective area. Nevertheless a unique speaker among the partners is the interface with the customer and coordinates the teams. This organization offers great flexibility allowing it to handle very large projects with highly skilled consultants in their respective areas.

Openness

Real Time Developer Studio is an open tool on several aspects :

- SDL-RT is free and fully documented,
- Storage format is based on XML textual standard,
- PragmaDev is an active partner of major actors in the real time and embedded world and technically integrates partner products.

Licensing and supported platforms

- Real Time Developer Studio runs on Windows, Solaris, and Linux platforms.
- Real Time Developer Studio uses floating FlexLm licenses that can be distributed over a network.
- C code used when generating code to support SDL-RT concepts is delivered as source code with no royalties.

UNIFIED MODELING LANGUAGE

SDL RT

Contact

PragmaDev
18, rue des Tournelles
75004 Paris France
Tel : +33 1 42 74 15 38
Fax : +33 1 42 74 15 58
http://www.pragmadev.com
mail : info@pragmadev.com

1 SDL-RT : Specification and Description Language Real Time can be found on http://www.sdl-rt.org
2 SDL : Specification and Description Language is standardized by International Telecommunication Union.
3 UML : Unified Modeling Language is standardized by Object Management Group.

RCS PARIS B 437 847 510 - design glg