

# PragmaDev

## change request

Emmanuel Gaudin

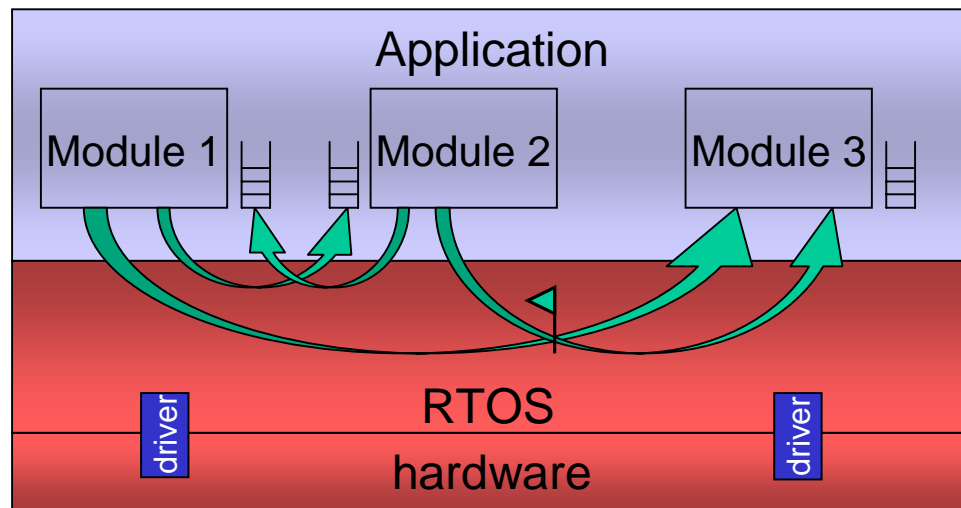
[emmanuel.gaudin@pragmadev.com](mailto:emmanuel.gaudin@pragmadev.com)

# Table of contents

- PragmaDev introduction
- Languages
- SDL-RT
- Tool support
- Market tendency
- Change requests

# PragmaDev

Dedicated to the development of a case tool for the development of real time and embedded software.



- All applications running on a Real Time Operating System
- Decomposed in tasks running concurrently
- Communicating through
  - Messages
  - Interrupts
  - Function calls
  - Semaphores

 Message queue

 Semaphore

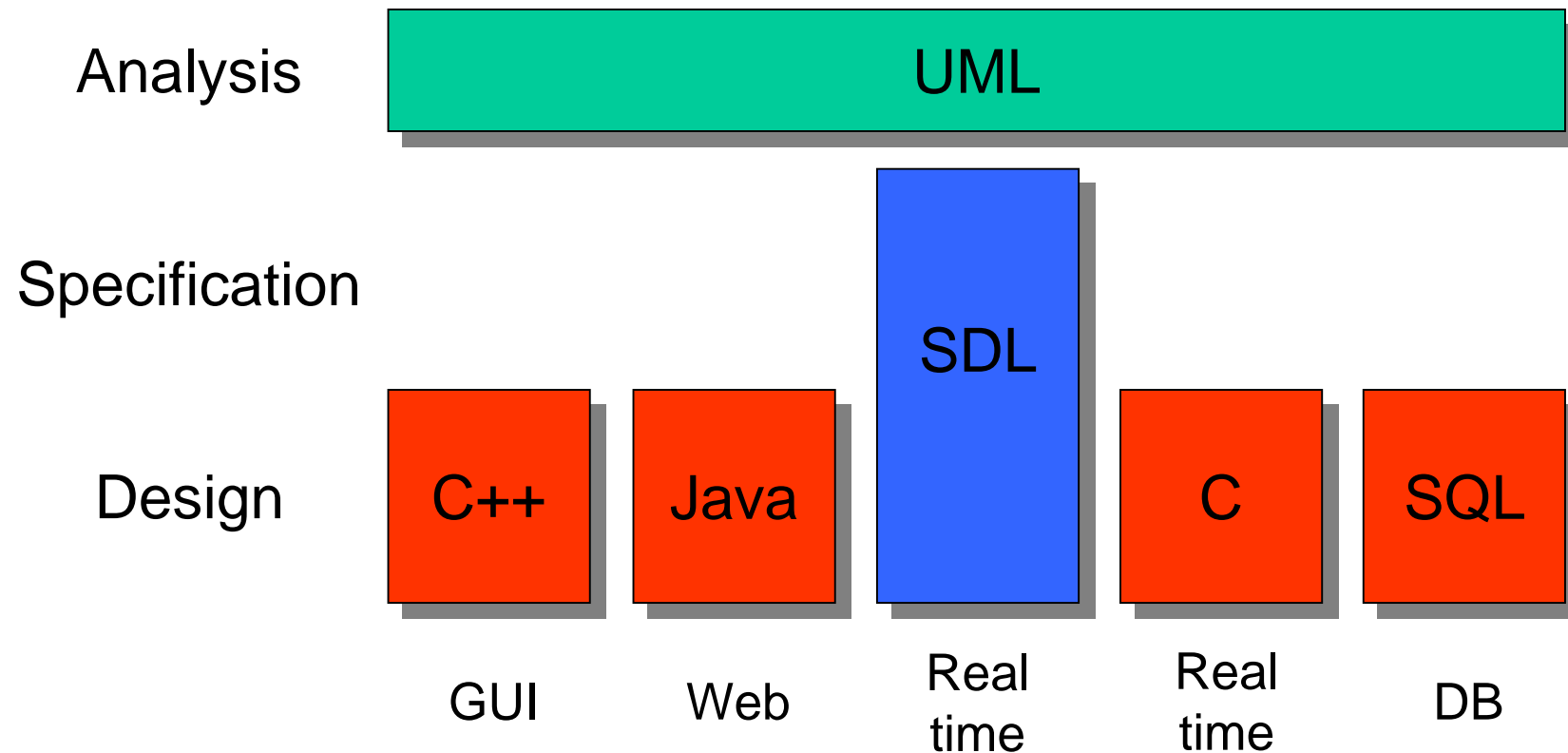
## State of the art

- C language is predominant (75%)
- C++ has been introduced in non real time parts of embedded (40%)
- Assembler (40%)
- Java is experienced in niches (less than 5%)
- 90% of the real time development projects use no graphical tool

# Existing languages

- **SDL** (Specification and Description Language) and **MSC** (Message Sequence Chart) are ITU (International Telecommunication Union) standards.
  - Event oriented,
  - Used by ETSI to standardize telecommunication protocols,
  - Graphical,
  - Formal (complete and non-ambiguous), e.g. allows to fully describe the system,
  - Object oriented,
- **UML** (Unified Modeling Language) standardized by the OMG (Object Management Group).
  - Can be used to represent any type of systems,
  - Graphical,
  - Used at a pretty high level of abstraction,
  - Not formal, e.g. another language is necessary to describe in detail (C, C++, Java, SDL),
  - Very object oriented.

# Languages positioning



## No real time specificity in UML

- UML has no graphical representation for classical real time concepts such as: tasks, semaphores, messages, timers...
- UML is adapted to C++ for **static** data representation.
- Deployment diagram perfect for **distributed** systems.
- In practice UML models are not synchronized with the design.

## Will UML2.0 help ?

- UML 2.0 allows to define domain specific profiles but does not define any
  - Will a real time profile be defined ?
  - Meanwhile UML 2.0 models will probably not be portable from one tool to another and have specific notations



## UML 2.0 trend

- UML 2.0 Sequence diagram has integrated most of the features of the SDL Message Sequence Chart
  - UML 2.0 structural diagram is equivalent to the SDL block diagram
- Interesting things come from SDL

## SDL: the perfect picture

- SDL **graphical abstractions** (architecture, communication, behavior) improve quality, reduce development time, ease maintenance:
  - Development time is globally reduced by **35%**
  - Number of mistakes per 1000 lines is **5 times less** than C code
- SDL being formal, it is possible to **simulate** the system behavior on host with graphical debugging facilities.
- SDL being formal, full **code generation** is possible.
- SDL being **object oriented**, software components are reusable (ETSI telecommunication protocol standards fully use object orientation).

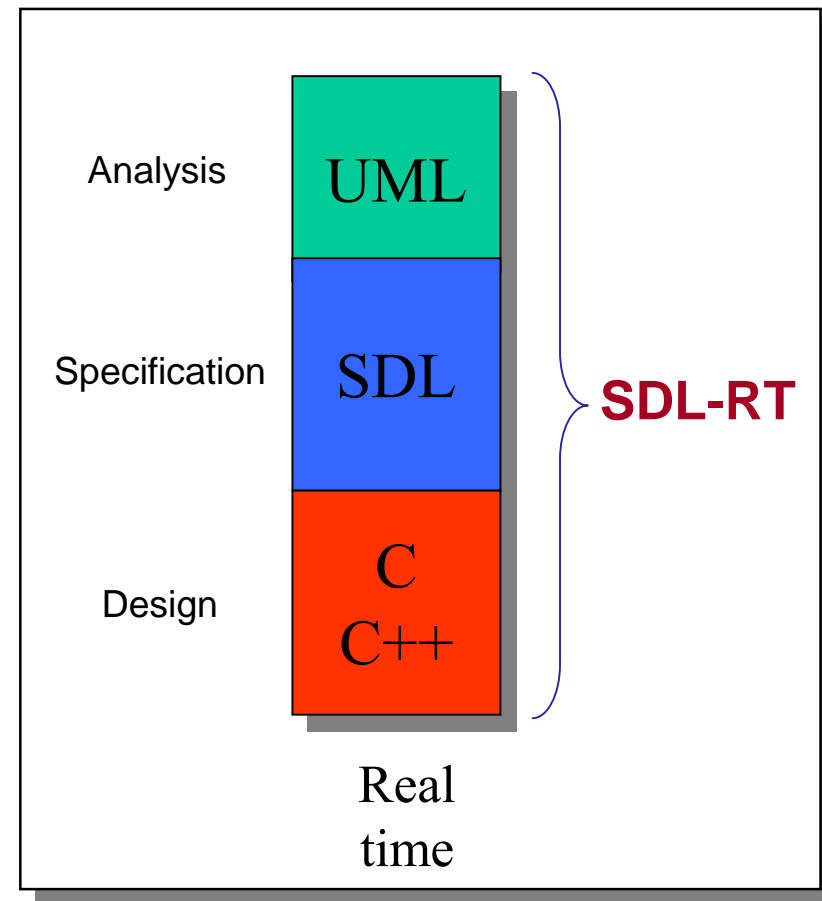
## SDL: the reality

- All existing software modules (RTOS, drivers, legacy code) provide C APIs, not SDL,
  - Some classical real time concepts are not present in SDL such as pointers and semaphores,
  - SDL syntax is not suited for design.
- 
- **Integration with legacy code is difficult,**
  - **Integration with COTS components is tricky (driver or RTOS),**
  - **Developers are frustrated,**
  - **Generated code is not legible,**

# The technical solution: SDL-RT

## SDL-RT is just the habits and usage in the industry when using SDL

- Keep UML diagrams at high level during analysis and requirements
- Keep the SDL graphical abstraction (architecture, communication, behavior).
- Introduce C data types and syntax instead of SDL's.
- Remove SDL concepts having no practical implementation.
- Extend SDL to deal with uncovered real time concepts (interrupts, semaphores).





specification & description language - real time

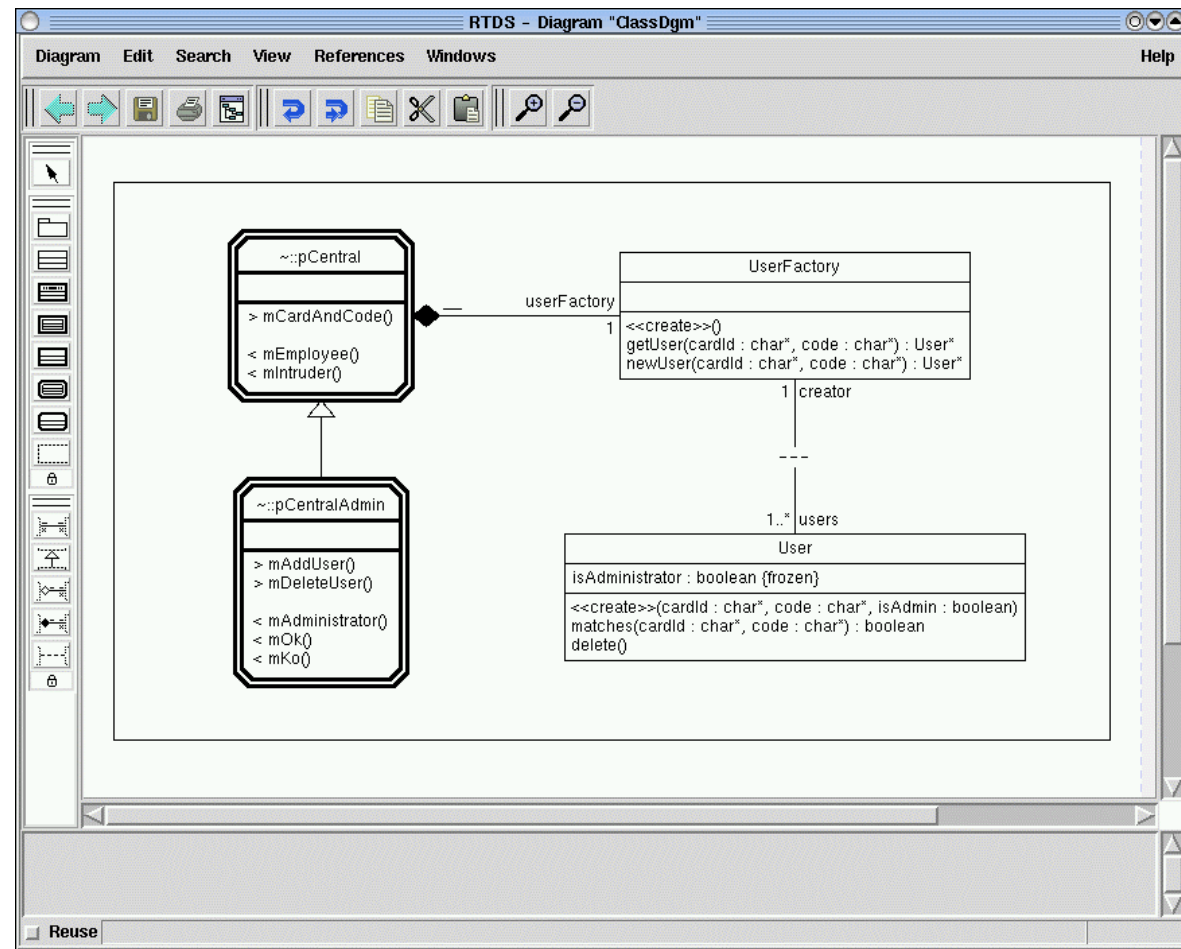
SDL-RT is:

- Available from <http://www.sdl-rt.org> for free,
- Legible,
- Based on a standardized textual format (XML).



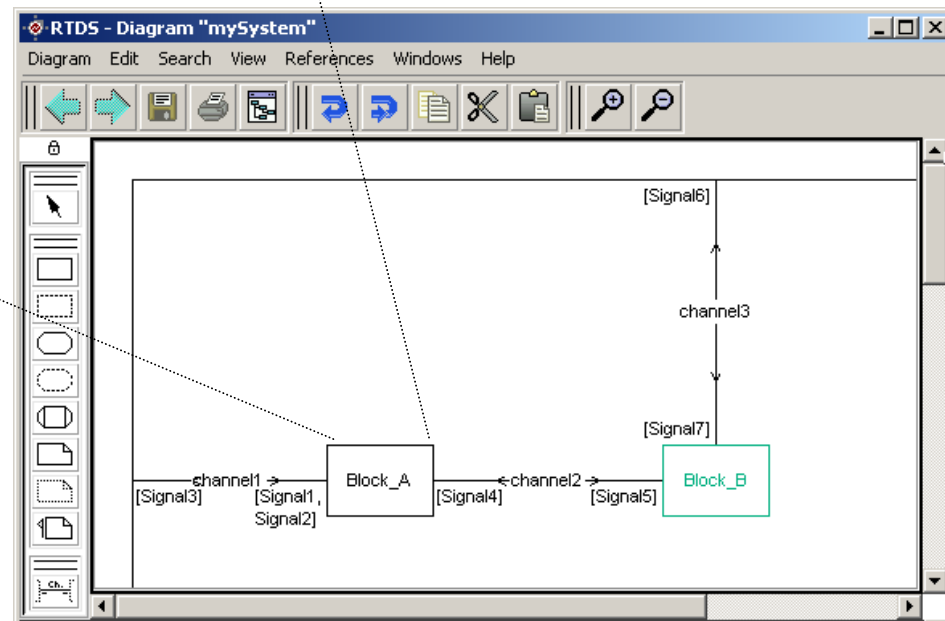
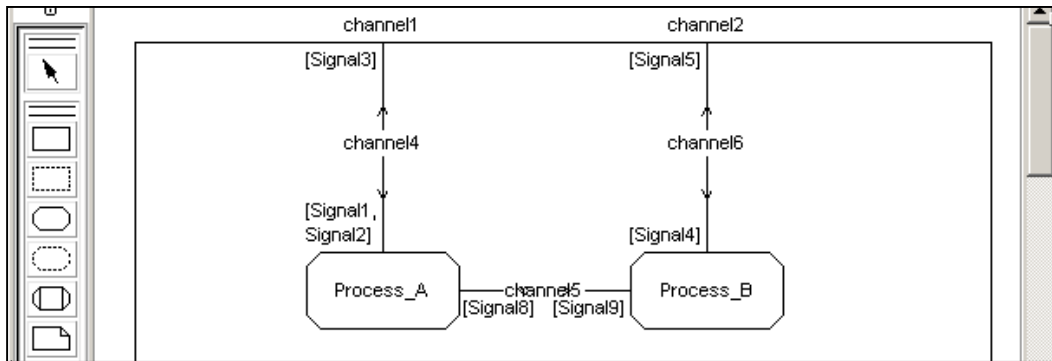
# SDL-RT: 6 views

Relations  
 between static  
 classes (C++)  
 and dynamic  
 classes (SDL)



# SDL-RT: 6 views

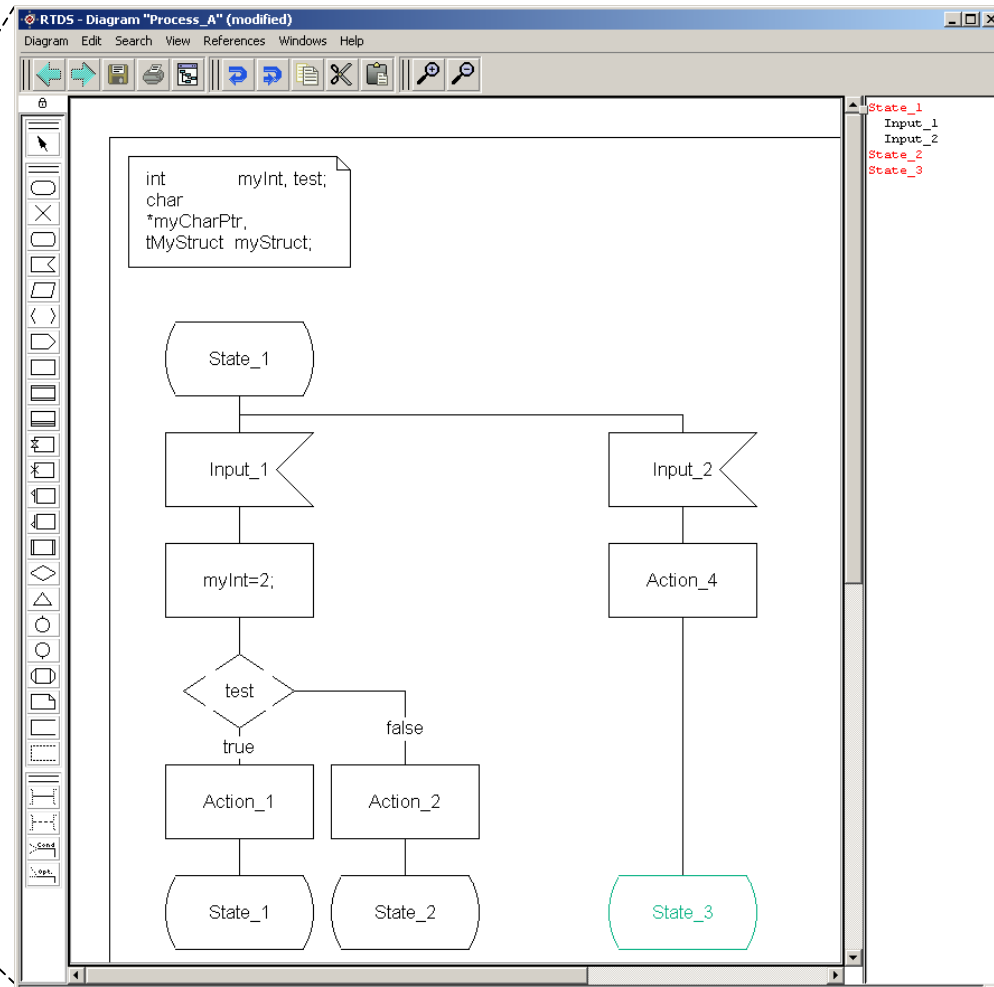
Architecture  
and  
Communication



# SDL-RT: 6 views

Behavior  
and  
Data

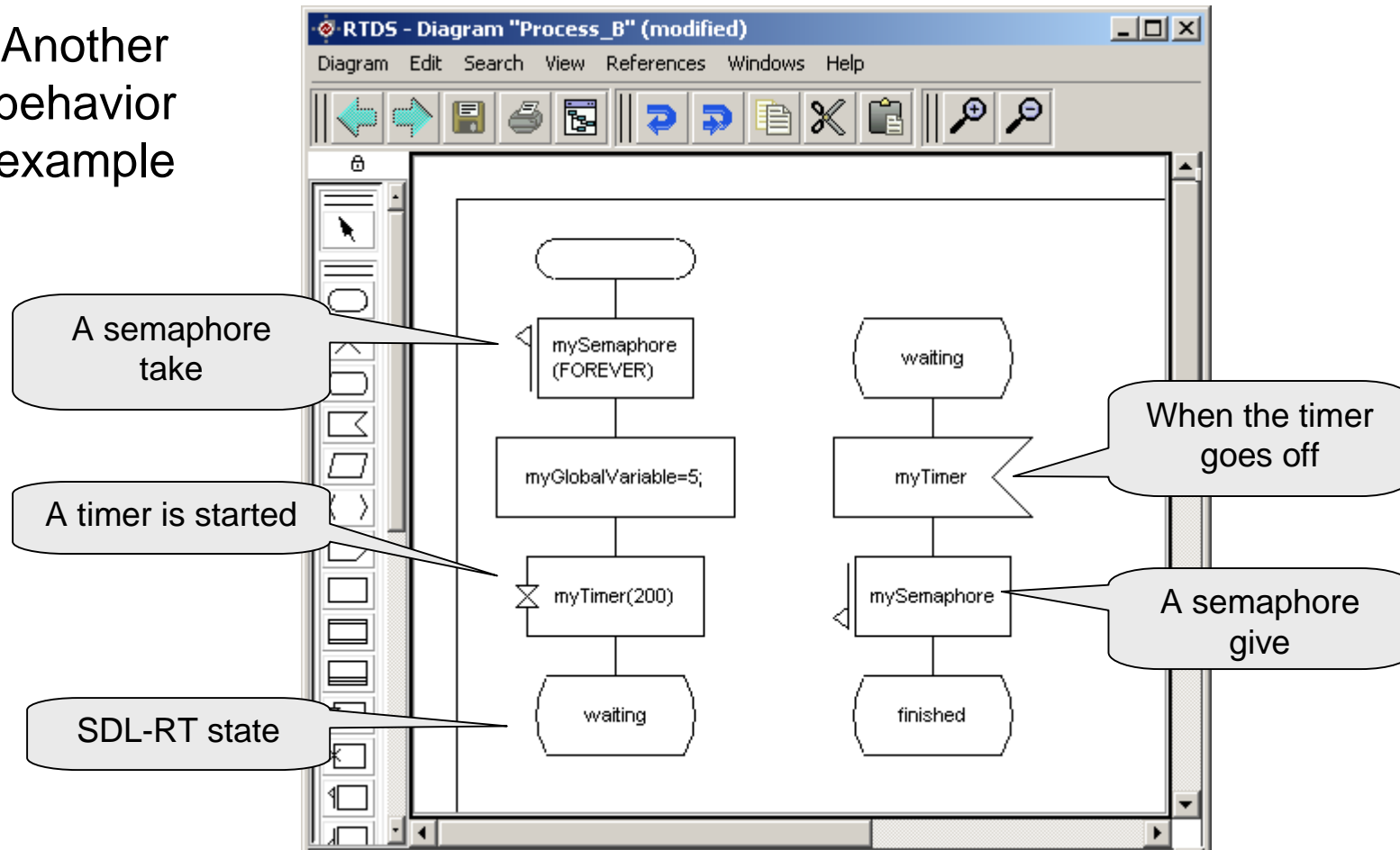
Process A





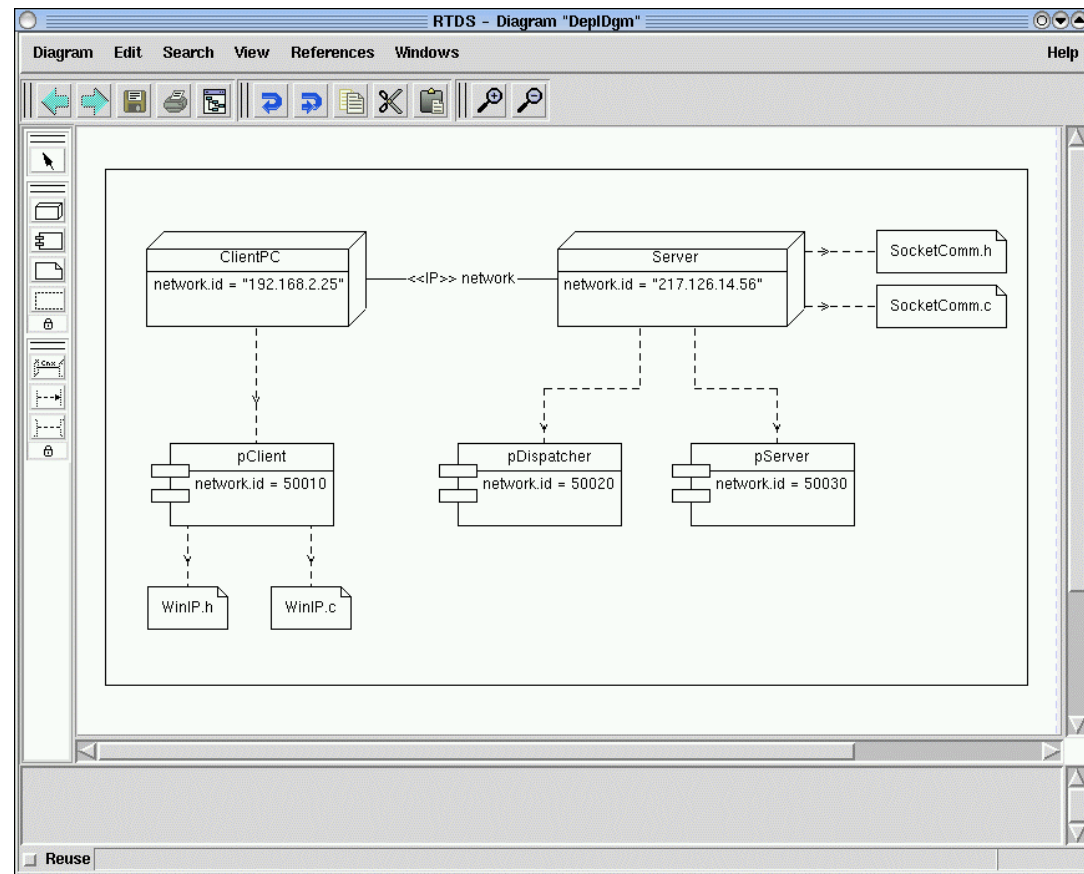
# SDL-RT: 6 views

Another behavior example



# SDL-RT: 6 views

Physical  
deployment



## **SDL-RT: graphical representations**

- **Library of components**
- **System architecture**
- **Interface definitions**
- **Application deployment**
- **Real time concepts**
- **Key points in the design**

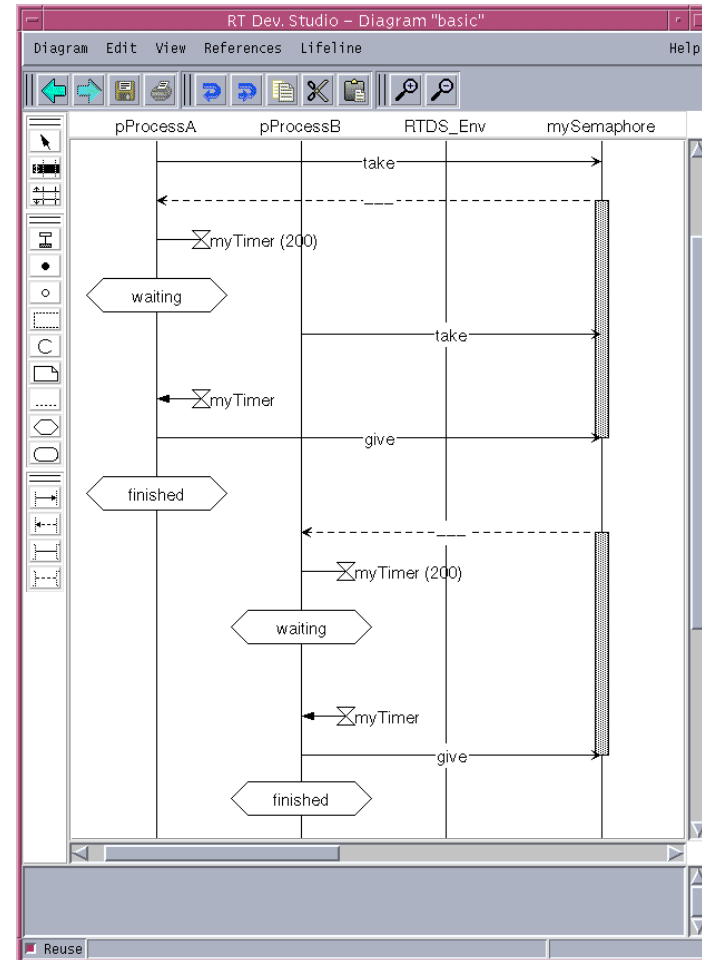
# SDL-RT MSC: dynamic view

## SDL-RT Message Sequence Chart

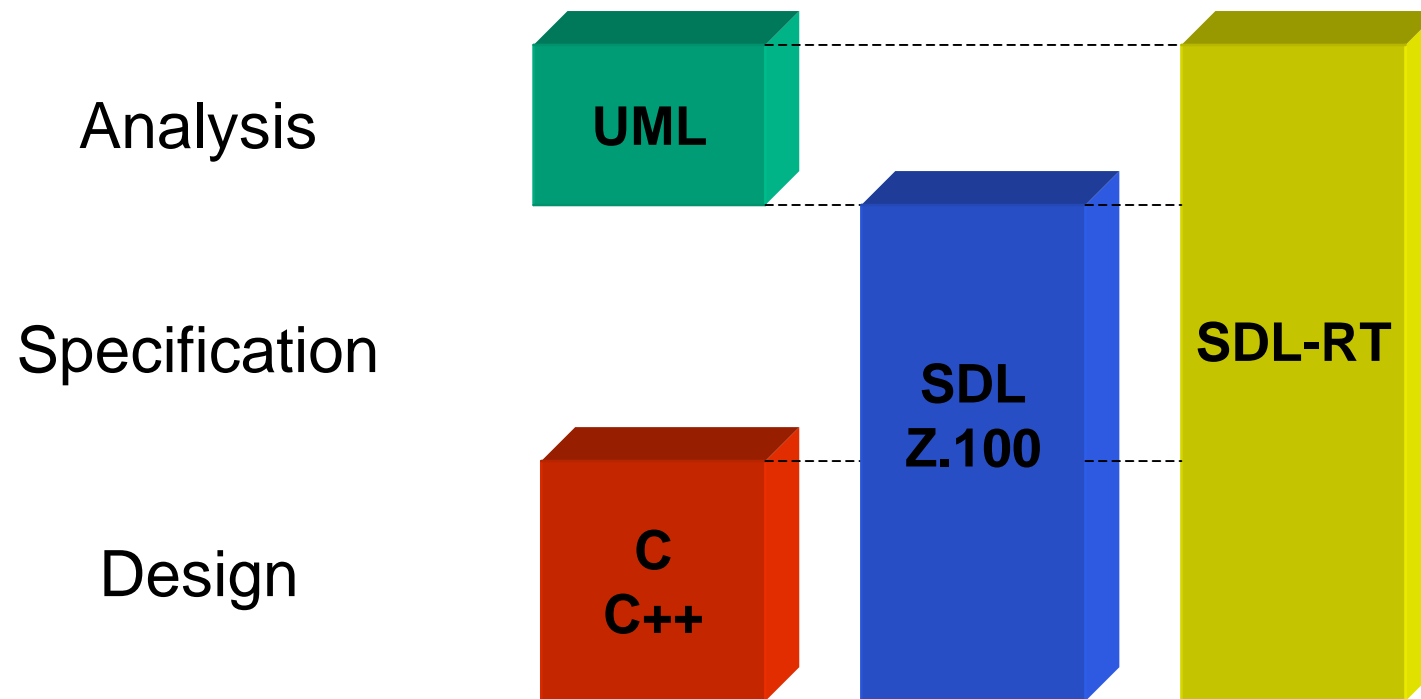
- Vertical lines represent a task, the environment or a semaphore,
- Arrows represent message exchanges, semaphore manipulations or timers.

Can be used:

- As specification
- Execution traces



## RTDS: supported languages



# RTDS: supported languages

## UML

- Editors
- C++ stubs generator



## SDL Z.100

- Editors
- Syntactic et semantics checker
- Simulator



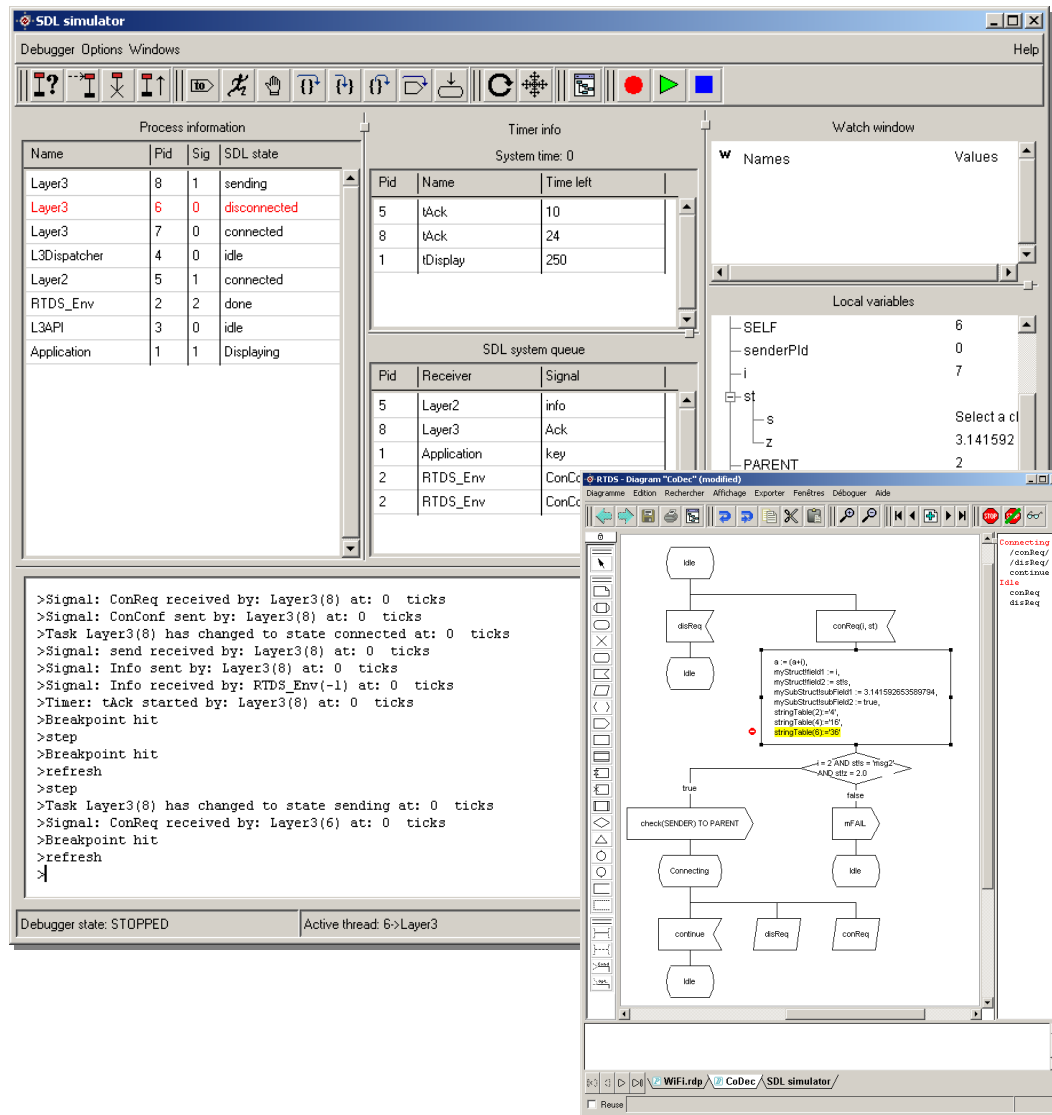
## SDL-RT

- Editors
- Syntax et semantics checker
- Code generator
- Graphical debugger



# SDL Z.100 simulator

An SDL Z.100 graphical debugger

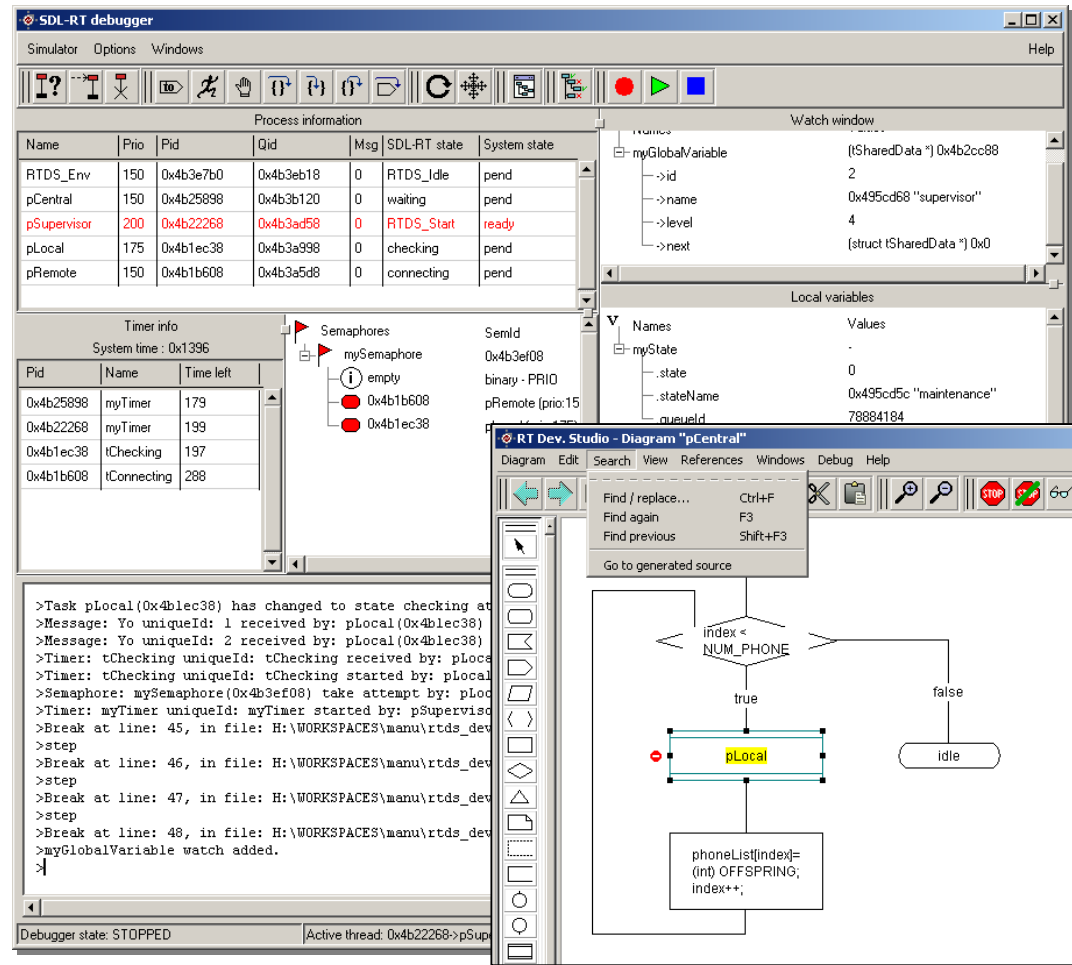


- Breakpoints, stepping, in the SDL diagrams,
- Externally defined or interactive operator calls,
- Dynamic MSC traces,
- Connecting an external tool is possible through a socket.

# Tools: The SDL-RT debugger

Debug at SDL-RT level:

- Breakpoints, stepping, in the SDL/RT diagrams or in the generated C files,
- Dynamic MSC traces,
- Connecting an external tool is possible through a socket.



The screenshot displays the SDL-RT debugger interface with several panels:

- Process information:** A table showing the state of various processes.
 

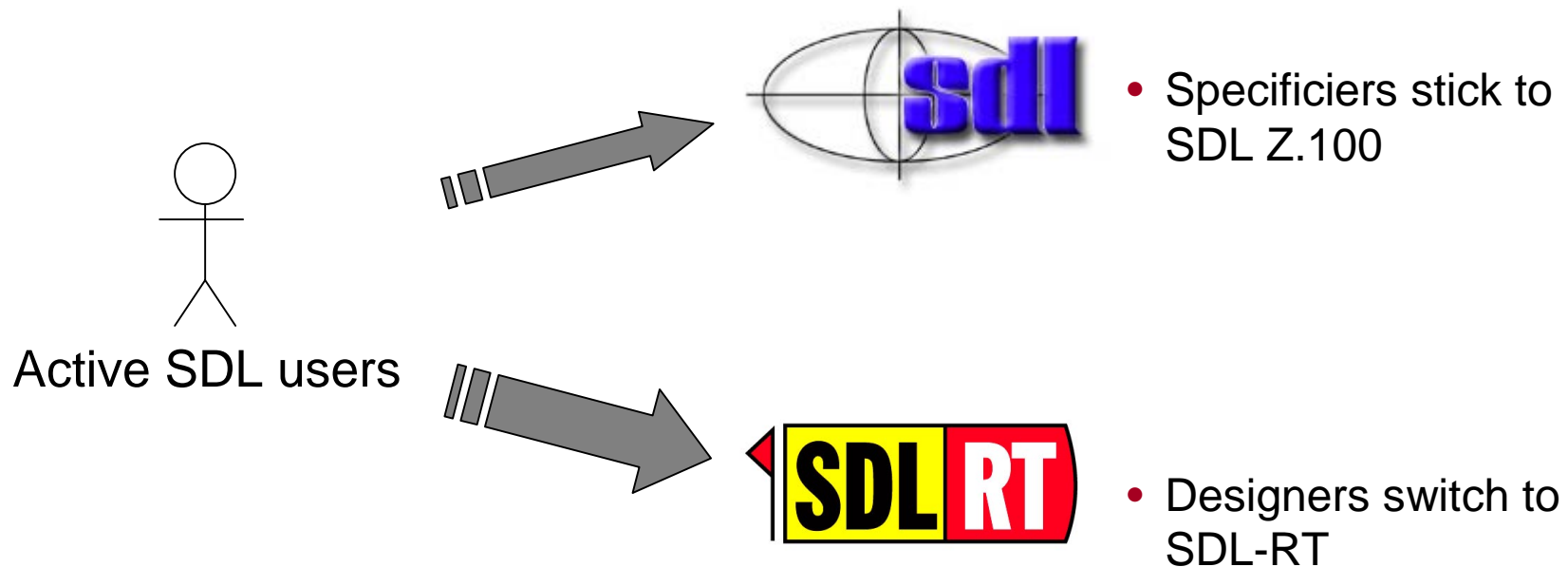
Name	Prio	Pid	Qid	Msg	SDL-RT state	System state
RTDS_Env	150	0x4b3e7b0	0x4b3eb18	0	RTDS_Idle	pend
pCentral	150	0x4b25898	0x4b3b120	0	waiting	pend
pSupervisor	200	0x4b22268	0x4b3ad58	0	RTDS_Start	ready
pLocal	175	0x4b1ec38	0x4b3a998	0	checking	pend
pRemote	150	0x4b1b608	0x4b3a5d8	0	connecting	pend
- Timer info:** A table showing active timers.
 

Pid	Name	Time left
0x4b25898	myTimer	179
0x4b22268	myTimer	199
0x4b1ec38	tChecking	197
0x4b1b608	tConnecting	288
- Semaphores:** A tree view showing semaphore objects like 'mySemaphore' and 'empty'.
- Watch window:** Displays the values of global variables like 'myGlobalVariable'.
- Local variables:** Displays the values of local variables like 'myState'.
- Diagram "pCentral":** A state transition diagram showing states like 'idle' and 'pLocal'. A transition is triggered by the condition 'index < NUM\_PHONE', leading to the 'pLocal' state. Below the diagram, the corresponding C code is shown:
 

```
phoneList[index]= (int) OFFSPRING; index++;
```
- Debugger console:** Shows a log of system events, including task state changes, message reception, timer events, semaphore operations, and breakpoint hits.



## SDL usage trend



**Let's try to have the active SDL users stick to SDL**

## Z.109

- The first Z.109 version used UML extension mechanisms to translate a UML model into an SDL model.
  - That ended up in a UML model that was as rich and specialized as an SDL system.
  - In the end either the UML or the SDL system was useless !
  - And that made UML and SDL competitors...
- Z.109 should focus on the natural complementary aspects of the 2 languages and try to avoid equivalence.
- Z.109 will probably be the only standardized UML profile for telecommunication systems

## Open up to other data types and syntax

- SDL data types are not suited for design
  - No SDL compiler / debugger
  - High level features (assignment, comparison) support requires to generate data manipulation functions or macros
  - Integration problems with legacy code, other modules, RTOS...
  - Missing concepts such as pointers

## Open up to other data types and syntax

- ✓ The best would be to support any other data type and syntax
  - C/C++
  - ADA
  - ...
- ✓ If not suited to open on any data type, C/C++ support is the best opening

## Priorities

- SDL has the concept of signal with priority but does not support priority on process
- RTOS support priority on tasks but not on messages
- Priority is very usefull when designing a telecommunication or a real time system
  
- ✓ Priority on SDL process instances could be an extension; if omitted the SDL system behaves like before.

## Scheduling

- Scheduling policy is undefined in SDL
  - Making validation of SDL systems is pretty difficult
  - Behavior might be different during simulation and on target
- ✓ Scheduling should be definable in order to have a representative simulation of the final system and ease validation
- ✓ Combined with priorities the behavior will be closer to the one found on an operating system

## Semaphores

- Semaphores are one of the key synchronization mechanism in real time systems
- SDL is seen as a telecommunication language restricted to protocols
- ✓ Introducing semaphores would extend SDL usage to any application based on a real time operating system
- ✓ Introducing extension mechanism to add semantic aspects similar to the one found in UML (example: timer freeze)

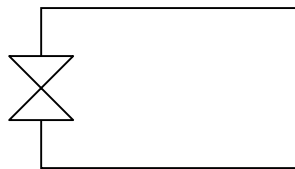
## Define a meta-model for SDL

- The standard seems to gather a lot of concepts without any global organisation
- The standard is not naturally open to extensions
- ✓ Defining a meta-model would help to organise the standard, make it more consistent, and easy to open to extensions.

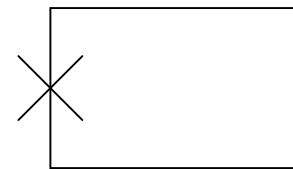


## Timers

- There is no graphical symbol to start or cancel a timer
  - There is a graphical symbol to receive a timer signal
- 
- ✓ Introduce a start timer symbol
  - ✓ Introduce a cancel timer symbol



Start timer suggestion



Cancel timer suggestion

## Improve object orientation

- A specialized agent can not call the super-class transition
- Specializing a transition usually means adding treatment to the inherited one; not replacing it
- ✓ Introduce a super transition call symbol
- ✓ The super class next state can be used

## Simplification suggestions

- ✓ Procedure should not be able to see the variables of the PARENT (not the caller)
- ✓ Remote procedure concept should be removed because it implies discrepancies in the standard:
  - Synchronous call in an asynchronous environment ?
  - Which procedure is called when several instances of the PARENT procedure ?
  - The procedure caller can modify the remote procedure PARENT variables !
- ✓ Are VIRTUAL and REDEFINED syntax usefull ?