

New features of



4.0

Thierry Lecomte

thierry.lecomte@clearsy.com

Extension mechanism



localization

Animation



Code Generation



Rodin₂B Conversion



Automatic refinement



B₄SYN

B₂Ladder

Localization

The screenshot displays the Atelier B software interface. On the left, a project tree shows a component named 'MPU (OK|OK|968|342|64%)' with sub-items like 'Composants', 'Definitions', and 'Librairies'. The main workspace shows a table of components with their verification status:

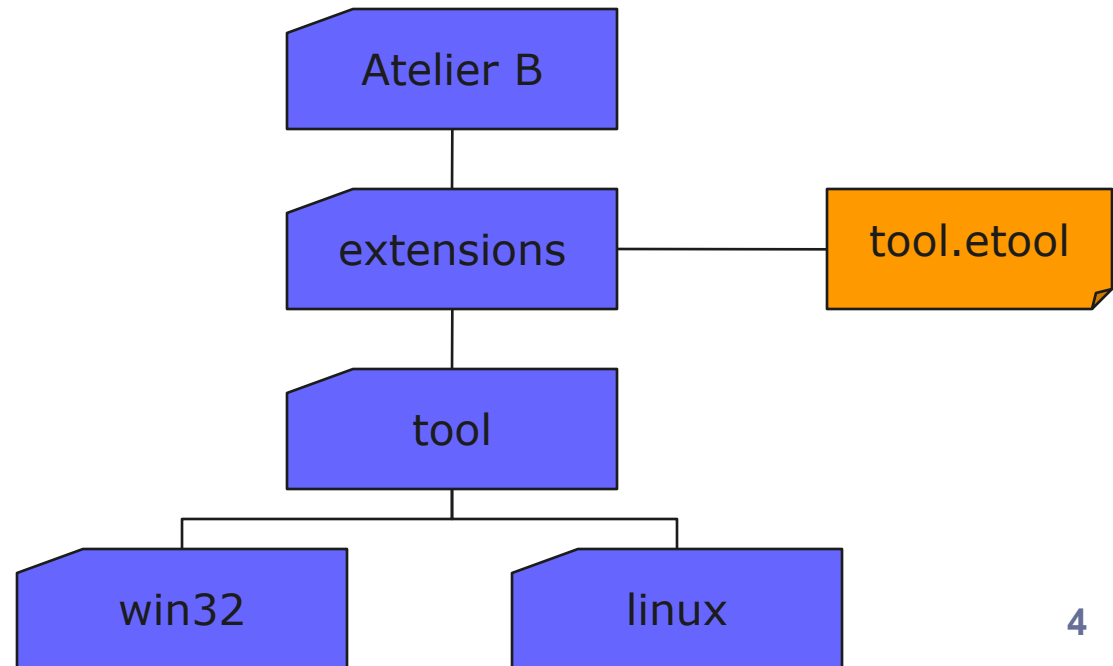
Composant	Typage vér
BASIC_B4SYN	OK
mpu00	OK
mpu01	OK
mpu18	OK
mpu17_values_ctx	OK
mpu17	OK
mpu16	OK
mpu15	OK
mpu14	OK
mpu13	OK
mpu12	OK

On the right, the 'Préférences' (Preferences) dialog is open, showing settings for 'Fenêtre principale', 'Projets', 'Nouveaux composants', and 'Éditeur interne'. The 'Vérification du code' section is expanded, showing 'Perform semantic analysis' and 'Effectuer les contrôles B0' checked. The 'Complétion de code et navigati' section shows 'Commande btags' set to 'C:\Program Files\...'. The 'Spell-Checking' section is highlighted with a red box, showing 'Spell-check comments' checked and 'Default Language' set to 'fr_FR'. A 'Choose dictionary' dialog is also open, showing 'Available dictionaries' with 'fr_FR' selected.

- English/French interface
- Spell-check comments

Extension mechanism

- Ability to run external tools from within the GUI
- At different level: project, component, proof obligation, goal
- xml file (etool extension) in the « extensions » Atelier B directory



Extension mechanism

■ Recognized tags

- ▶ **externalTool**: The enclosing tag of the etool file. Defines an external tool
- ▶ **toolParameter**: Used to define parameters for running the command
- ▶ **temporaryFile**: Used to create temporary files before running the command
- ▶ **command**: The command that should be run
- ▶ **param**: a parameter to the command
- ▶ **componentList**: the list of components that are parts of the project. Used to specify parameters to the command

■ externalTool

- ▶ **name**: the internal name of the tool (required)
- ▶ **category**: the category of the tool. Can be "component", "project", "po" or "goal" (required)
- ▶ **label**: the text that is displayed in the corresponding menu entry (required)
- ▶ **shortcut**: an optional keyboard shortcut for calling the tool
- ▶ **tooltip**: information that is displayed about the tool
- ▶ **icon**: the icon that will be used in the corresponding menu entry. This must be the name of a .png file located in the same directory as the etool file.
- ▶ **outputType**: the type of output of the tool. If this tag is not present, the output is assumed to be text. Valid values are `text` and `html`

■ toolParameter

- ▶ **name (required)**: the name of the parameter. The name correspond to the name of the variable
- ▶ **type (required)**: the type of the parameter. Can be one of "resource", "exefile", "file" and "tool"
- ▶ **default**: a default value for the parameter if no other value is available
- ▶ **optional**: can be "yes" or "no" to indicate that the parameter is optional. If the parameter is not optional, the tool will not be executed in the case where no value is found for the parameter
- ▶ **description**: a textual description of the parameter. This field is used when creating a preference page for configuring the tool

■ etc.

Extension mechanism

■ Predefined variables

Variable	Tool type	Description
<code>#{projectName}</code>	All	The name of the currently selected project, or the project to which the selected component belongs
<code>#{projectBdp}</code>	All	The absolute path to the "bdp" directory of the project
<code>#{extensionsDir}</code>	All	The path to the extension directory
<code>#{componentName}</code>	Component, Po, Goal	The name of the selected component
<code>#{componentPath}</code>	Component, Po, Goal	The absolute path to the selected component file
<code>#{componentDir}</code>	Component, Po, Goal	The absolute path to the directory where the component is located
<code>#{poName}</code>	Po, Goal	The name of the current proof obligation
<code>#{poGoal}</code>	Goal	The current goal (as text)
<code>#{poHypothesis}</code>	Goal	The current list of hypothesis (as text)

Extension mechanism

■ A sample example

```
<externalTool name="openbdp"  
  category="project"  
  label="Open bdp"  
  shortcut="Ctrl+Y"  
  tooltip="Opens the project database directory in an explorer window"  
  icon="openbdp.png"  
>  
  <command>c:\windows\explorer.exe</command>  
  <param>${projectBdp}</param>  
</externalTool>
```

■ ProB

```
<externalTool category="component" name="ProBTclTk" label="Animate with ProB (Tcl/Tk)">  
  <toolParameter name="editor" type="tool" configure="yes"  
  default="c:/program files/prob/ProBWin.exe" />  
  <command>${editor}</command>  
  <param>${componentPath}</param>  
</externalTool>
```

Brama

- Animation of Event-B projects ported from Rodin to Atelier B
 - Qt environment
 - PredicateB library redesigned in C++
 - Flash / 3D engine
 - Common interface with B Motion Studio ?

- To be released Q4 2010

Rodin2B

- Imports Rodin xml files (machines and contexts) into an existing Event-B project in Atelier B
- To be combined with software project wizard
 - Events as software services specification, requiring a scheduler
 - Events to compose
- To be released Q4 2010

Generating Ada code from Event B model

- *Application of aggregation rules to transform a set of events into an algorithm*

```
SELECT P ∧ Q THEN R END  
[]  
SELECT P ∧ not Q THEN S END  
~>  
SELECT P THEN  
  IF Q THEN R ELSE S END  
END
```

Condition:

$P \wedge Q \Rightarrow [R] \text{ not } P$

$P \wedge Q \Rightarrow [S] \text{ not } P$

Generating Ada code from Event B model

```
SELECT P THEN R END  
[]  
SELECT Q THEN S END  
~>  
SELECT P THEN R;S END
```

Condition:
 $P \Rightarrow [R] Q$

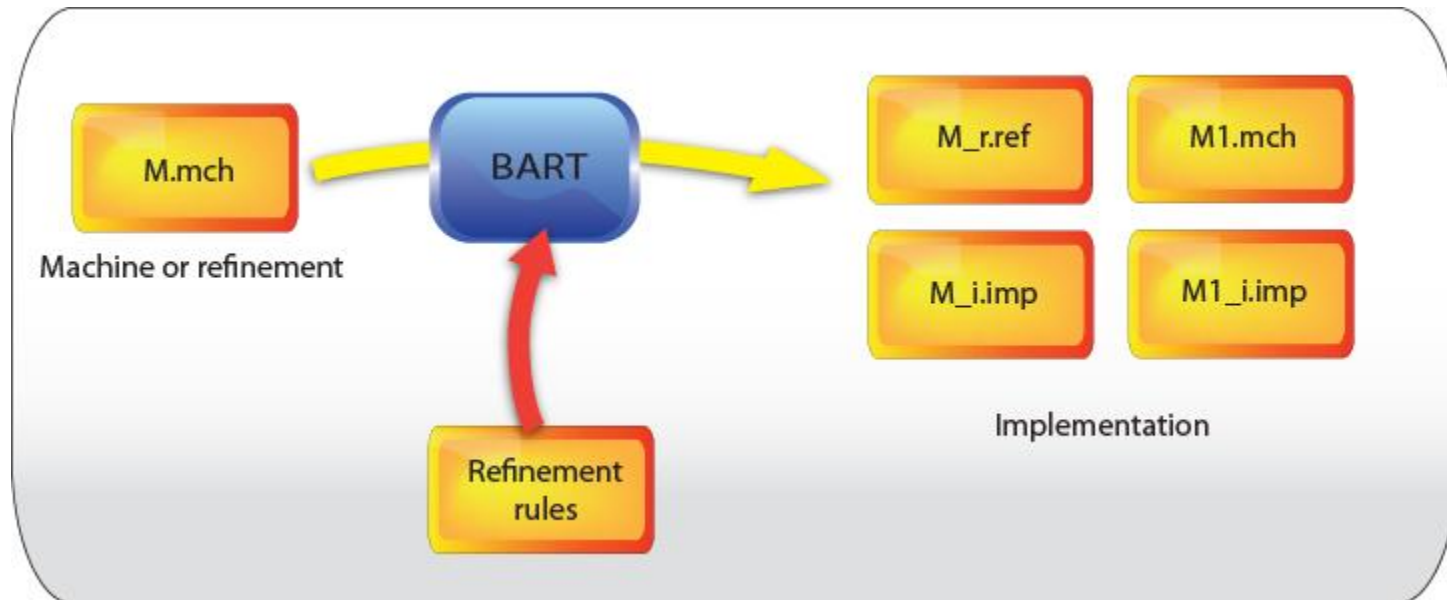
Bart v1.1

- More stable version
 - Bugs removal
 - Now supports trivial examples
 - Improved refinement rules database
- Presented at ABZ'2010 – Bart tutorial
- Addition: Bart GUI User Manual
- Extension to Event-B models in 2011



Setting up methodology and tools (automatic refinement)

- Input: complete set-theoretic model of a software
- Output: refinements and implementations

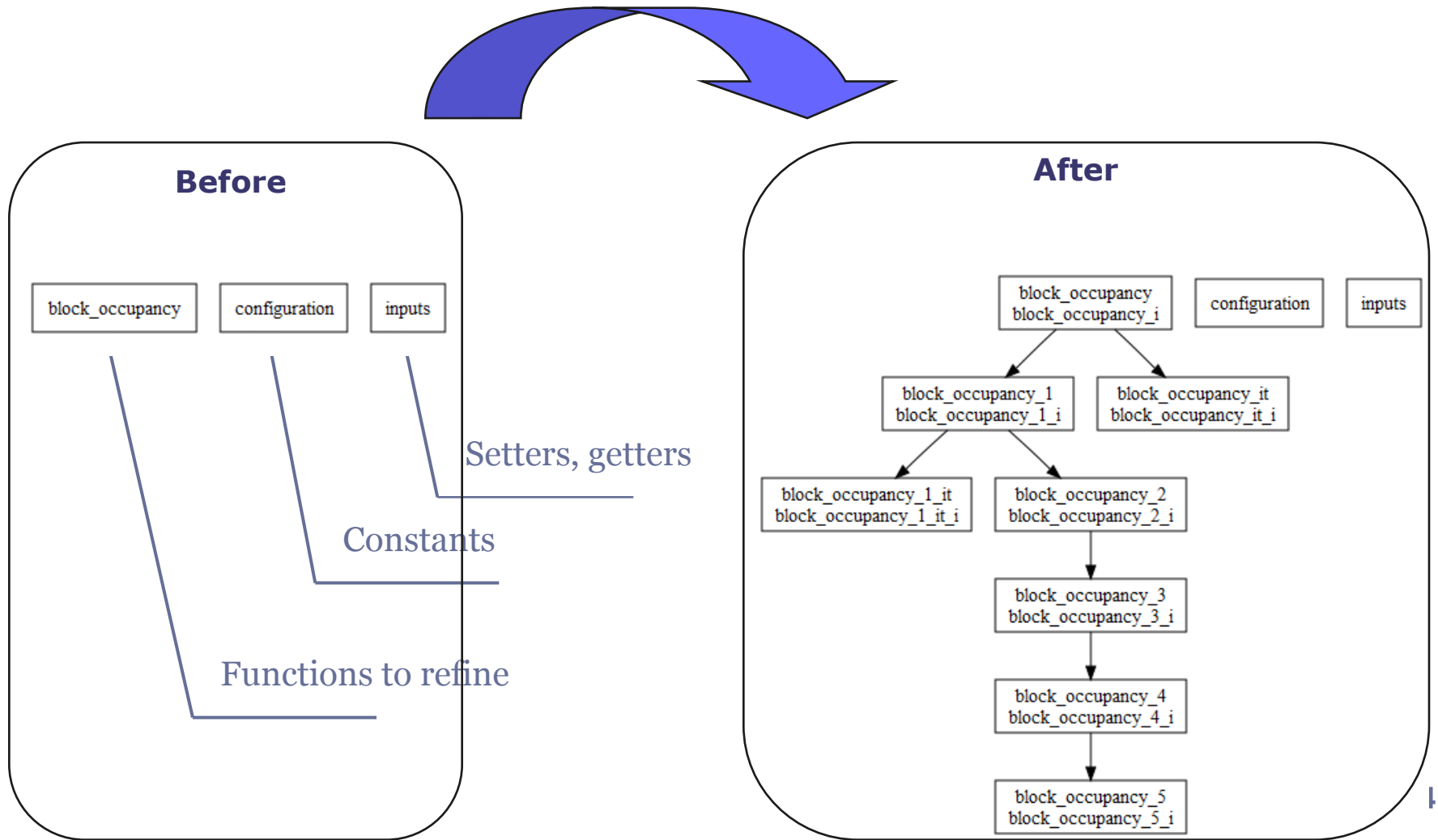


- Refinement engine: applying transformation rules

```
RULE assign_a_bool_subset_b_c_11
REFINES
  @a := bool(@b <: @c-@d)
REFINEMENT
  @a := bool(@b <: @c & @b /\ @d = {})
END;
```

```
RULE assign_a_bool_belongs_b_c_16
REFINES
  @a := bool(@b|->@d : @c*@e)
REFINEMENT
  @a := bool(@b:@c & @d:@e)
END;
```

Setting up methodology and tools (automatic refinement)

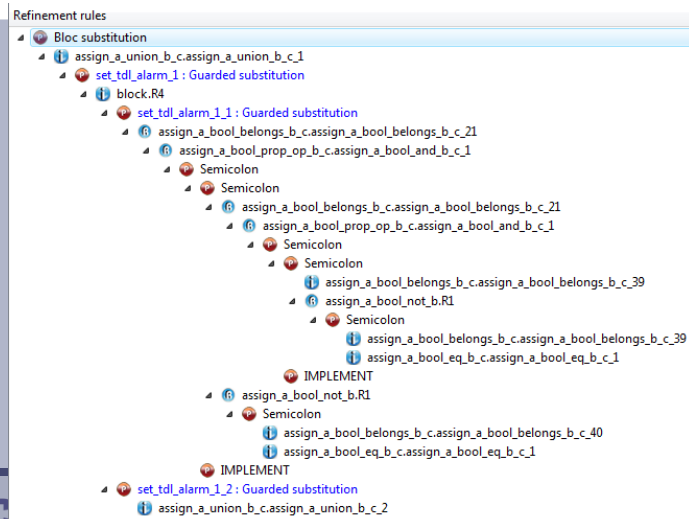


Setting up methodology and tools (automatic refinement)

$tdla := tdla \cup ob - mb - otd$

Refinement tree

Condensed Implementation



```

IMPLEMENT (vg_boucle <-- initier_iteration_t_block) ;
WHILE vg_boucle = TRUE DO
  IMPLEMENT (vg_boucle , l_1 <-- poursuivre_iteration_t_block) ;
  /* ?tdla := tdla \ / ({l_1} /\ ob - mb - otd)? */
  set_tdl_alarm_1([par_in_0_1:=l_1]) =
  [
    BEGIN
      /* ?l_2 := bool(par_in_0_1 : ob - mb - otd)? */
      [par_out_1_1:=l_2] <-- set_tdl_alarm_1_1([par_in_1_1:=par_in_0_1]) =
      [
        BEGIN
          l_5 := ob_i ( par_in_1_1) ;
          l_7 := mb_i ( par_in_1_1) ;
          l_6 := bool(l_7 = FALSE) ;
          l_3 := bool(l_5 = TRUE &
            l_6 = TRUE) ;
          l_8 <-- lire_otd(par_in_1_1) ;
          l_4 := bool(l_8 = FALSE) ;
          par_out_1_1 := bool(l_3 = TRUE &
            l_4 = TRUE)
        END
      ] ;
    IF l_2 = TRUE THEN
      /* ?tdla := tdla \ / {par_in_0_1}? */
      set_tdl_alarm_1_2([par_in_1_1:=par_in_0_1]) =
      [
        BEGIN
          tdl_a_i ( par_in_1_1) := TRUE
        END
      ]
    END
  ]
END
END
INARIANT
vg_boucle = bool(t_block_a_traiter /= {}) &
t_block_a_traiter \ / t_block_traites = t_block &
t_block_a_traiter /\ t_block_traites = {} &
tdla = tdla$0 \ / t_block_traites /\ ob - mb - otd

```

Pattern matching in detail

Substitution
to refine

Hypothesis	Node substitution	Node rule	Matching rules
------------	-------------------	-----------	----------------

```
l_16 := bool(par_in_1_1 : otd)
```

Matching rule

```
RULE assign_a_bool_belongs_b_c_40  
REFINES
```

```
@a := bool(@b : @c)
```

```
WHEN
```

```
DECL_OPERATION(@d <-- @e(@f) |  
PRE  
  @g  
THEN  
  @d := bool(@f : @c)  
END)
```

```
IMPLEMENTATION
```

```
@a <-- @e(@b)
```

```
END
```

Matching getter

```
res <-- lire_otd(p_block) =  
PRE  
  p_block : t_block  
THEN  
  res := bool(p_block : otd)  
END;
```

Refined substitution

```
l_16 <-- lire_otd(par_in_1_1)
```

ComenC v2

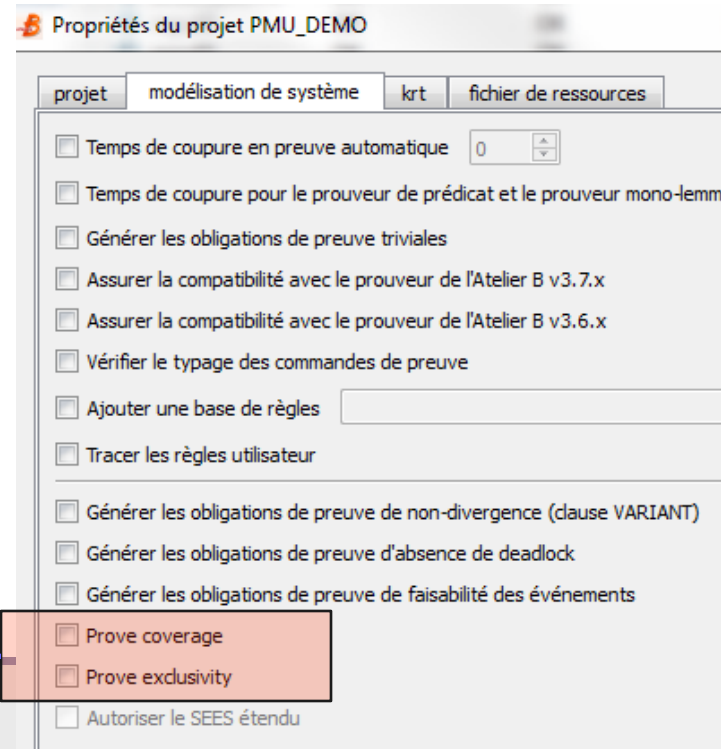
- Existing ComenC C code generator
 - Difficult to use (too many safety critical constraints)
 - Difficult to maintain (Cocktail Compiler Compiler library requires old SunOS)
- Next version is being implemented in C++ on top of the B Compiler
- To be released Q4 2010
- Two independent ongoing C code generator developments based on B Compiler

B4SYN

- Event-B to VHDL translator
- Extra information (b4syn file), including:
 - Circuit signature (inputs, clocks, outputs (synchronous/asynchronous))
 - Combinatorial/synchronous/asynchronous events



- Extra proof obligations
- Will be part of Microelectronics Atelier B



B2Ladder

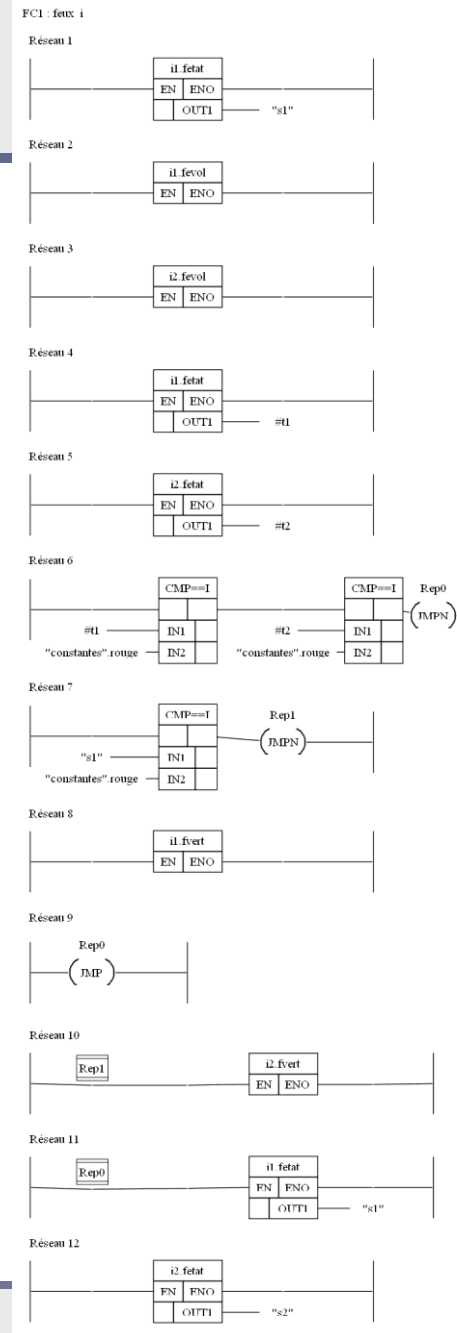
- Event-B to Ladder translator
- Based on the B Compiler
- Specific to SIEMENS S7
- To be extended to:
 - HIMA Ladder
 - LIST programming language
- Combined with a test case generator (IMPLEMENTATION)
- Will be part of PLC-based Atelier B

Traffic light management

```

s1,s2 <-- evol =
VAR t1,t2 IN
  s1 <-- i1.fetat;
  i1.fevol;
  i2.fevol;
  t1 <-- i1.fetat;
  t2 <-- i2.fetat;
  IF t1=rouge & t2=rouge THEN
    IF s1=rouge THEN
      i1.fvert
    ELSE
      i2.fvert
    END
  END;
END;

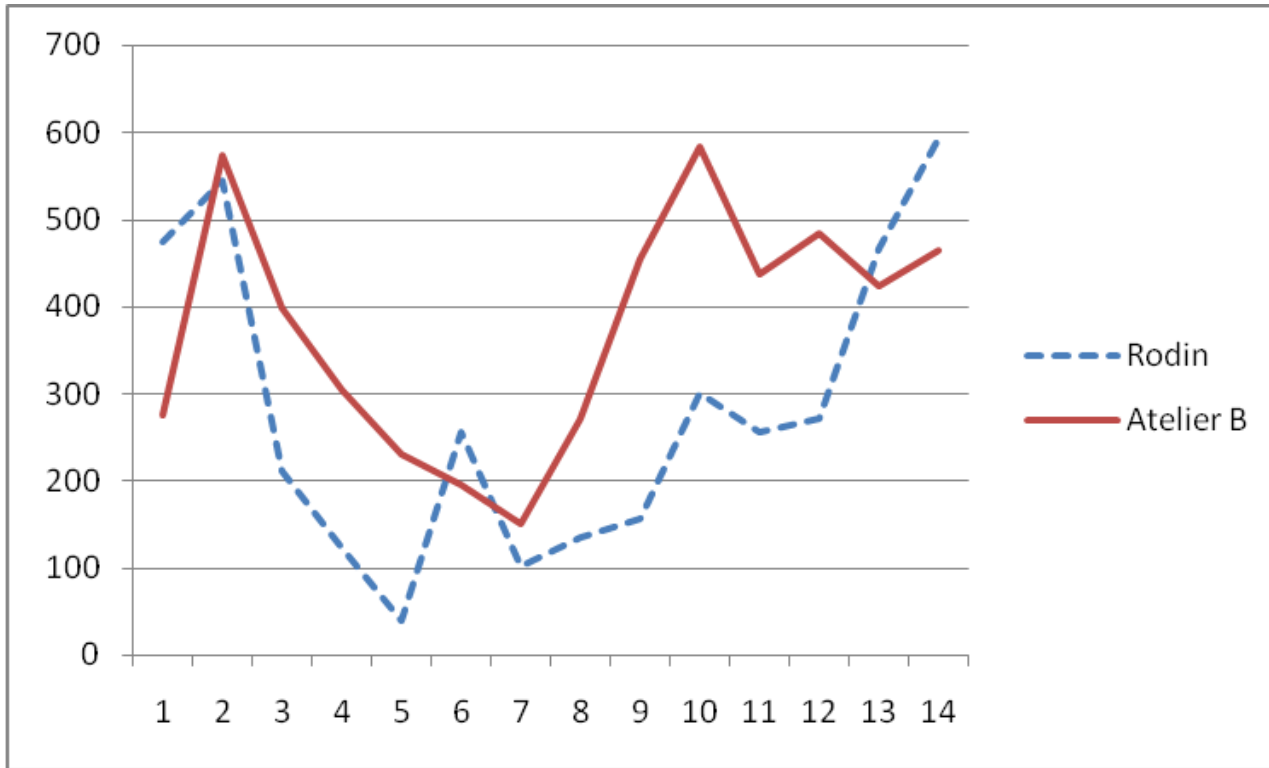
s1<--i1.fetat;
s2<--i2.fetat
END
  
```



- 7 sections: Bart, ComenC, Bcompiler, GUI, Brama, Languages, Modelling
- 8 000+ different visitors / month
- Worldwide contributors (addition, correction, translation)
- Bug status per tool

ComenC edit		Workaround	Action planned
528	When using b2c translator (ComenC), Makefiles files are not generated for the project.		Correction in Atelier B 4.1
573	Error messages generated by ComenC (b2c executable file) are not displayed in the related GUI.		Corrected in Atelier B 4.0.1
583	When trying to translate while the ComenC executable is missing, Atelier B crashes with the following error message: <pre>ASSERT failure in QVector<T>::at: "index out of range", file /usr/local/Trolltech/Qt-4.4.1/include/QtCore/qvector.h, line 323 Abandon</pre>		Correction in Atelier B 4.1
587	Generated headers display an incorrect Atelier B version ("atelierb_current" instead of Atelier B 4.0)		Correction in Atelier B 4.1
613	ComenC translates constants in "const", instead of using #define, leading to multiple references error message when such constants are declared in a context machine seen by more than one machine.	Transform constant declarations const into #define constructs in the generated source code	Correction in Atelier B 4.1
620	The token [] (empty sequence) is not recognized by ComenC	Use {} (empty set) instead	Correction in Atelier B 4.1

Downloads



monthly downloads since January 2009

- Countries
- Germany
 - France
 - Japan
 - UK
 - India
 - China

downloads (samples)

Atelier B 4.0:	3 877
Rodin 1.0:	608
Rodin 1.3:	449

Industry

- First use of Atelier B 4.0 for the development of a safety critical application



ALSTOM

Perspectives

- Tighter integration with ProB
- Generic proof obligation generator
- Support for real numbers and floating point
- Dedicated environments (PLC, microelectronics)

Thank you for your attention



CLEAR SY
System Engineering