



Un environnement de développement C/C++ interactif

Présentation RIA, le 10 décembre 2004  
Thomas Lemaire, Maxime Cottret, Sébastien Bosch  
LAAS-CNRS, groupe RIA

- petit historique : de vizir à jafar
- comparatif calife/jafar
- généralités sur Jafar
- module Jafar
- swig, STL, boost
- gestion des erreurs
- documentation
- tests unitaires

- vizir (ViZion In Robotique)
  - 1981 – 1986
  - projet ARA (bras manipulateur et tapis roulant...)
- calife
  - 1983 – 2005 (?)
  - 20<sup>aine</sup> de thésards
  - Hilare Junior, Adam, Lama, Dala, Karma
- jafar...

	<b><i>Calife</i></b>	<b><i>Jafar</i></b>
langage	C	C/C++
shell	Tcl	Tcl, Python,...
environnement modulaire	oui	oui
squelette de module	non	oui
version de module	non	version.revision
interface langage/shell	iwac	swig
chargement modules	statique (shell)	dynamique (module tcl)
aide en ligne (tcl/C)	oui/non	oui/bof
complétion	oui (built-in)	oui (eltclsh, tclreadline)
arguments par défaut	oui (iwak)	oui (C++)
erreur	variable globale	exceptions
test unitaire	non	oui (boost)
documentation	doc utilisateur, source	doxygen
compatibilité shell GenoM	non	oui
librairies externes	non	STL, openCV, boost (uBlas, unit_test, BGL,...)
installation	checkout	checkout + librairies

- Modulaire
  - commande *jafar-module*
  - modèle de module dans `share/template_module`
  - dépendance entre modules (bof...)
- Interactif
  - shell tcl quelconque, mais pourquoi pas python (pas encore supporté...)
  - `share/initJafar.tcl`
  - utilisation de packages tcl
- Documentation
  - `doc/html/index.html`

- Aspects modulaires :
  - espace de nom C++  
jafar::module::
  - type d'erreurs  
ModuleException
  - package tcl  
module
  - espace de nom tcl  
module::
  - group doxygen  
module
  - librairies  
libmodule.so,.a
- commande *jafar-module*
  - création d'un nouveau module
  - mise à jour

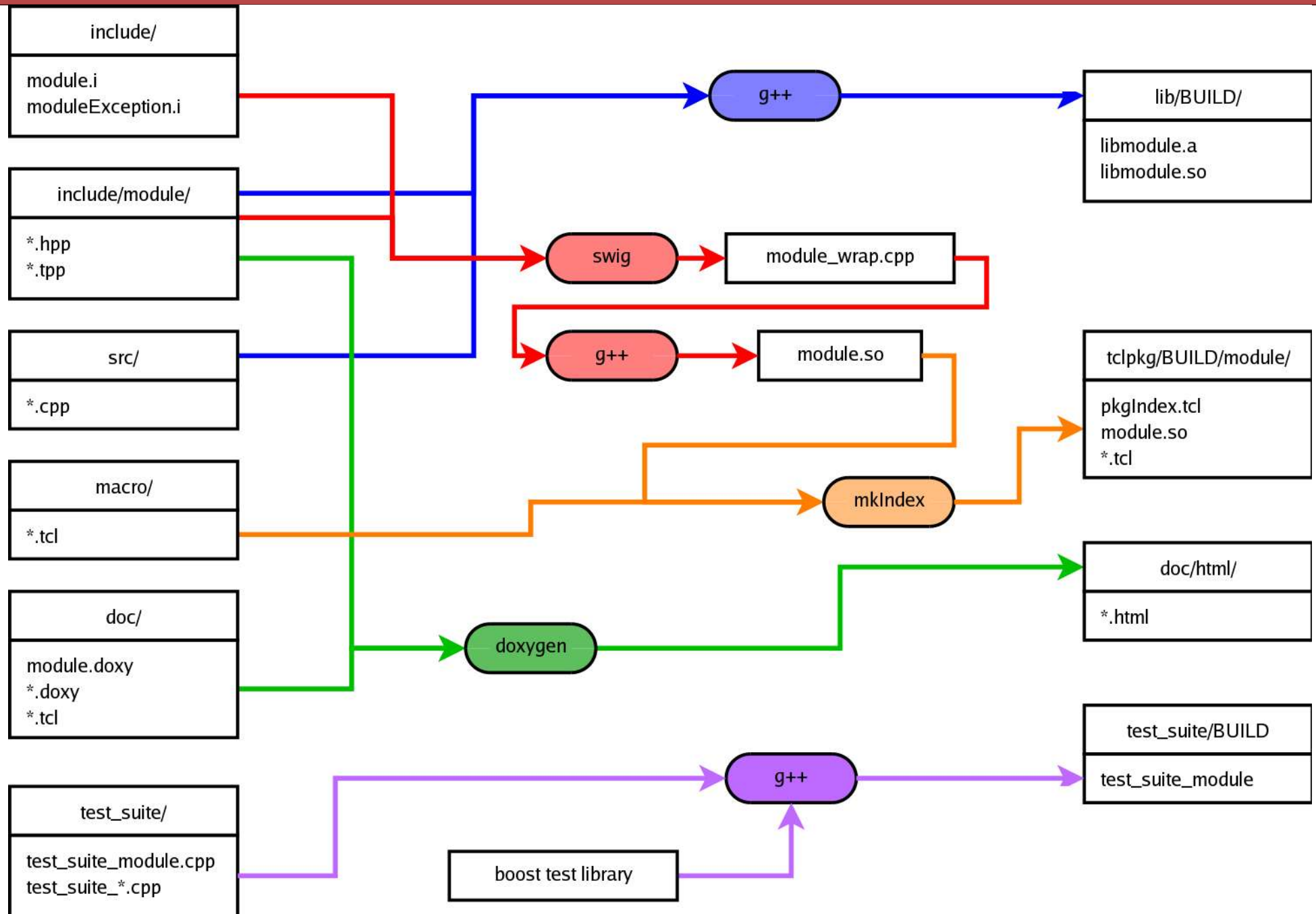
```
pollux[modules] ../bin/jafar-module -c pipo
Creating module pipo
  template dir   : /home/tlemaire/Work/jafar/share/template_module
  destination dir : /home/tlemaire/Work/jafar/modules/pipo
```

```
Directory /home/tlemaire/Work/jafar/modules/pipo...ok
```

```
. ... ok
./macro ... ok
./Makefile ... ok
./configure ... ok
./aclocal.m4 ... ok
./src ... ok
./src/pipoException.cpp ... ok
./Makefile.module.in ... ok
./test_suite ... ok
./test_suite/test_suite_pipo.cpp ... ok
./COPYRIGHT ... ok
./include ... ok
./include/pipoException.i ... ok
./include/pipo.i ... ok
./include/pipo ... ok
./include/pipo/pipoException.hpp ... ok
./include/pipo/pipoTcl.hpp ... ok
./doc ... ok
./doc/pipo.doxy ... ok
./README ... ok
./configure.ac ... ok
./User.make ... ok
./config ... ok
./config/config.guess ... ok
./config/missing ... ok
./config/install-sh ... ok
./config/user.ac ... ok
./config/ltmain.sh ... ok
./config/config.sub ... ok
```

```
Module pipo successfully created !
```

# Module Jafar





- Modification du User.make
- Cibles principales :
  - all : config et module
  - config : lance le script configure avec les options du User.make
  - module : compile et installe les librairies et le module tcl
  - tcl-pkg : installe le module tcl
  - test : compile et lance la test\_suite du module
  - clean
  - acf : régénère le script configure (User.ac)

```
# $Id: User.make 481 2004-11-25 16:08:53Z tlemaire $ #

# module version
MODULE_VERSION = 0
MODULE_REVISION = 1

# configure option
CONF_OPT = --with-boost

# modules dependencies
USE_MODULES = kernel jmath

# to add custom libraries
# consider also modifying .config/User.m4 to check for these libs at
# configure time

# LDFLAGS +=
LIBS += -lkernel -ljmath

CPPFLAGS += $(BOOST_CPPFLAGS) -DBOOST_UBLAS_USE_EXCEPTIONS
CPPFLAGS += -DNDEBUG
# CPPFLAGS += -DJFR_NDEBUG

CXXFLAGS += -g -ggdb -Wall
```

- SWIG = "Simple Wrapping Interface Generator"
  - supporte tcl, python, perl,...
  - C++: arguments par défaut, classe, héritage, template,...
  - récupère la documentation au format doxygen (python : *docstring*, tcl : ??)
- tcl
  - *package* tcl ou interpréteur
  - support des *namespace*
  - libswig: std::string, std::vector,...

# Swig - Example

documentation doxygen

```
/* $Id: helloworld.i 363 2004-10-27 14:01:40Z tlemaire $ */  
/** swig/tcl interface file for module helloworld.  
 *  
 * \file helloworld.i  
 * \ingroup helloworld  
 */
```

entêtes pour compiler  
module\_wrap.cpp

```
%module helloworld  
  
%{  
  
#include <string>  
  
// uncomment to control debug output from tcl  
#include "kernel/jafarDebug.hpp"  
  
#include "helloworld/helloworldTcl.hpp"  
  
#include "helloworld/helloWorld.hpp"  
  
%}
```

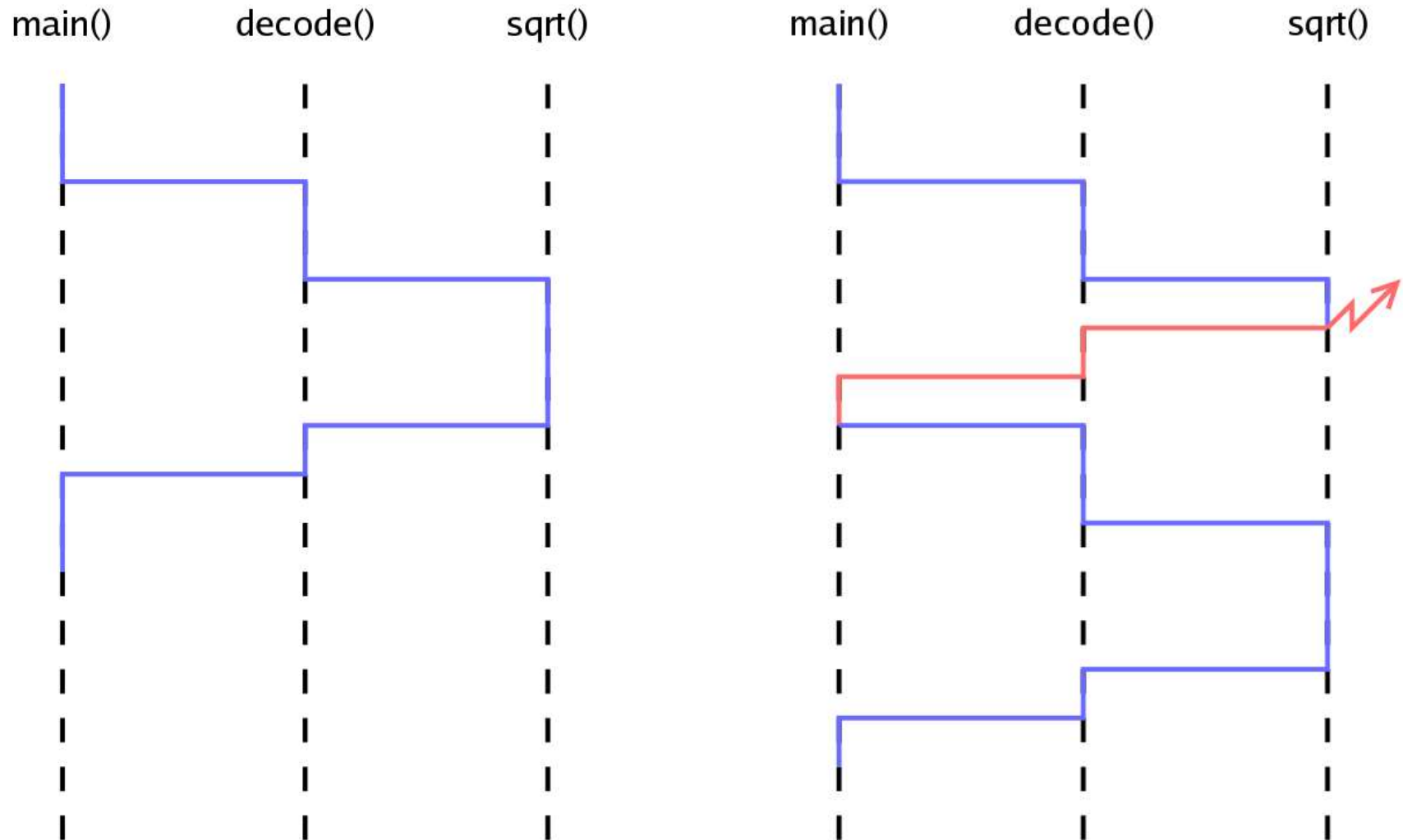
entêtes a wrapper en tcl

```
%include "std_string.i"  
  
%include "helloworldException.i"  
  
// uncomment to control debug output from tcl  
%include "kernel/jafarDebug.hpp"  
  
%include "helloworld/helloworldTcl.hpp"  
%template(print)  
jafar::helloworld::print<jafar::helloworld::HelloWorld>;  
  
%include "helloworld/helloWorld.hpp"
```

- Atouts :
  - *Standard* Template Library
  - intégré à g++
  - performante
  - documentée
- Défauts :
  - lenteur à la compilation (C++, template)
- Exemple :
  - string
  - conteneurs : list, vector, map,...
  - flux E/S

- Déjà utilisée dans Jafar :
  - random, uBlas (module jmath)
  - date\_time
  - unit test
  - smart\_ptr
- Bientôt utilisé :
  - BGL (Boost Graph Library)
- Si vous avez besoin...
  - quaternion, octonion, serialization, tokenizer, tuple,...
  - <http://www.boost.org>

- Repose sur le mécanisme des exceptions :



- Informations communes :
  - ID, nom de module, nom de fichier, numéro de ligne, pile d'appel, message
- Les erreurs standards (classe `JafarException`)
  - programmation par contrat (pré-condition, post-condition, invariant)
  - erreur *run-time*
  - macros : `JFR_PRECOND`,...
- Les erreurs de modules
  - simples, classe `ModuleException`
  - complexes, classe utilisateur dérivant de `ModuleException`



# Exceptions – Example (1/2)

documentation doxygen

types d'exceptions

```
/** Base class for all exceptions defined in the module
 * helloworld.
 *
 * @ingroup helloworld
 */
class HelloWorldException : public jafar::kernel::Exception {
public:

    /** This enumeration defines exceptions id for the module
     * helloworld.
     */
    enum ExceptionId {
        EMPTY_HELLO, /**< The hello string is empty */
        BAD_FORMAT    /**< bad format for string hello */
    };

    /** Constructor. You should not use this constructor directly,
     * prefer macros jfrThrowEx or jfrCreateEx which fill for you
     * parameters \c file_ and \c line_.
     *
     * @param id_ exception id
     * @param message_ message used for debug
     * @param file_ where the exception was thrown
     * @param line_ where the exception was thrown
     */
    HelloWorldException(ExceptionId id_,
                        const std::string& message_,
                        const std::string& file_, int line_) throw();
    virtual ~HelloWorldException() throw();
    ExceptionId getExceptionId() const throw();

protected:
    ExceptionId id;
    static std::string exceptionIdToString(ExceptionId id_) throw();
};
```

# Exceptions – Example (2/2)

héritage

```
/** hello attribute not initialized. This is an over simple
 * example on how to add your own exception.
 *
 * @ingroup helloworld
 */
class HelloworldFormatException : public HelloworldException {
public:

    HelloworldFormatException(const std::string& hello_,
                              const std::string& message_,
                              const std::string& file_, int
line_)

        throw();

    virtual ~HelloworldFormatException() throw();

    const std::string& getHello() const throw();

private:

    /// the proposed hello string
    std::string hello;

};
```

nouveaux  
attributs

- Module indispensable, tous les modules sont «linkés» à *libkernel* :
  - exception de base, erreurs génériques
  - message de debug
  - log de données
  - tic, toc, help, ask,...
- Autres modules disponibles :
  - *image* (Sébastien, Maxime)
  - *jmath*, *filter* (Thomas)

- Documentation générale :
  - guide de l'utilisateur,
  - description d'un algorithme
- Documentation dans les sources (.hpp) :
  - fonctions, classes,...
  - extraite des sources et mise en forme
- Doxygen - [www.doxygen.org](http://www.doxygen.org)
  - syntaxe à la *javadoc*
  - génération d'une documentation uniforme en html, hyperliens,...
  - ou en pdf (latex), rtf,...

# Documentation – Example (1/2)

```
/** HelloWorld class. This class is a simple example.
 *
 * \ingroup helloworld
 */
class HelloWorld {

    private :

        /// contains the string to be displayed
        std::string hello;

        static bool checkHello(const std::string& hello_);

    public :

        /// Default constructor.
        HelloWorld();

        /** Constructor.
         * @param hello_ the initial string
         * @pre \a hello_ is not empty
         */
        HelloWorld(const std::string& hello_);

        /// Destructor.
        ~HelloWorld();

}; // class HelloWorld
```

# Documentation – Example (2/2)

```
/*!
```

```
\addtogroup helloworld Module helloworld
```

```
\version 0.3
```

```
\author thomas.lemaire@laas.fr, maxime.cottret@laas.fr
```

This is a very minimal helloWorld module. It shows how to add a simple class `jafar::helloworld::HelloWorld` and a user defined exception `jafar::helloworld::HelloEmptyException`.

```
\section secHelloworldHistory History
```

- 0.3 (2004-10-04) - Add test\_suite
- 0.2 (2004-09-20) - Add support for exceptions
- 0.1 (2004-09-09) - Initial version

```
\section secHelloworldMacro Macro
```

- `demoHello.tcl` is a very simple macro

```
\section secHelloworldExample Example use of helloWorld module
```

```
\subsection subsecHelloworldOld Using low level swig functions
```

```
\include helloworld_old.tcl
```

```
\subsection subsecHelloworldNew Using an object interface
```

```
\include helloworld_new.tcl
```

- *Boost test library*
- Test d'un algorithme :
  - évaluation de l'algorithme pour certaines entrées, comparaison avec le résultat attendu
- Test d'un module :
  - notion de *test\_suite*
- Utilisation
  - Non régression (modifications de l'algorithme, mise à jour librairie externe,...)
  - Compilation sur une nouvelle plate-forme
  - lancement périodique des tests, rapport générés automatiquement,...

# Tests unitaires – Exemple (1/3)

```
class test_HelloWorld {
public:

    HelloWorld hw;
    HelloWorld hw2;

    test_HelloWorld() : hw(), hw2("Hello world") {}

    void test_init() {
        BOOST_CHECK_EQUAL(hw.getHello().size(), 0);
        BOOST_CHECK_EQUAL(hw2.getHello(), "Hello world");
    }

    void test_setHello() {
        hw.setHello("Hello world");
        BOOST_CHECK_EQUAL(hw.getHello(), "Hello world");

        BOOST_CHECK_THROW(hw.setHello(""), jafar::kernel::JafarException);

        BOOST_CHECK_THROW(hw.setHello("what ever"), HelloWorldFormatException);
    }

    void test_clearHello() {
        hw.clearHello();
        BOOST_CHECK_EQUAL(hw.getHello().size(), 0);
    }
};
```



# Tests unitaires – Exemple (2/3)

```
class test_suite>HelloWorld : public test_suite {  
  
    public:  
  
        test_suite>HelloWorld() : test_suite("test_suite>HelloWorld") {  
            // add member function test cases to a test suite  
            boost::shared_ptr<test>HelloWorld> instance( new test>HelloWorld() );  
  
            test_case* test_case_init          =  
                BOOST_CLASS_TEST_CASE( &test>HelloWorld::test_init, instance );  
            test_case* test_case_setHello      =  
                BOOST_CLASS_TEST_CASE( &test>HelloWorld::test_setHello, instance );  
            test_case* test_case_clearHello    =  
                BOOST_CLASS_TEST_CASE( &test>HelloWorld::test_clearHello, instance );  
  
            test_case_setHello->depends_on( test_case_init );  
            test_case_clearHello->depends_on( test_case_init );  
  
            add(test_case_init);  
            add(test_case_setHello);  
            add(test_case_clearHello);  
        }  
};
```

# Tests unitaires – Exemple (3/3)

```
pollux[helloworld] gmake test
```

```
[...]
```

```
*****  
* Running tests for module helloworld *  
*****
```

```
test_suite/sparc-solaris2.9/test_suite_helloworld  
Running 3 test cases...
```

```
*** No errors detected
```

```
gmake[1]: Leaving directory `/home/tlemaire/Work/jafar/modules/helloworld'
```

- documentation :
  - <file:///usr/local/jafar/doc/html/index.html>
- repository subversion :
  - organisation standard (trunk/, branches/, tags/)
  - répertoire unstable\_modules/
- TP
  - création d'un module *devin*
  - prétexte : un jeu de devinette