Jafar, a C/C++ interactive development environnement

Cyril Roussillon

December 3, 2010

Content of the course

Main Goals

- Sharing: don't reinvent the wheel ?
  - Algorithms,
  - Visualisation,
  - Data access...
- Ease the development of **your software**.
- Provide visibility to software.
- Modular:
  - ease of maintenance
  - increased reuse
  - = faster development, less bugs

Main Goals

- Sharing: don't reinvent the wheel ?
  - Algorithms,
  - Visualisation,
  - Data access...
- Ease the development of **your software**.
- Provide visibility to software.
- Modular:
  - ease of maintenance
  - increased reuse
  - = faster development, less bugs

Main Goals

- Sharing: don't reinvent the wheel ?
  - Algorithms,
  - Visualisation,
  - Data access...
- Ease the development of **your software**.
- Provide visibility to software.
- Modular:
  - ease of maintenance
  - increased reuse
  - = faster development, less bugs

Main Goals

- Sharing: don't reinvent the wheel ?
  - Algorithms,
  - Visualisation,
  - Data access...
- Ease the development of **your software**.
- Provide visibility to software.
- Modular:
  - ease of maintenance
  - increased reuse
  - = faster development, less bugs

Main Goals

- Sharing: don't reinvent the wheel ?
  - Algorithms,
  - Visualisation,
  - Data access...
- Ease the development of **your software**.
- Provide visibility to software.
- Modular:
  - ease of maintenance
  - increased reuse
  - = faster development, less bugs

Main Goals

- Sharing: don't reinvent the wheel ?
  - Algorithms,
  - Visualisation,
  - Data access...
- Ease the development of **your software**.
- Provide visibility to software.
- Modular:
  - ease of maintenance
  - increased reuse
  - = faster development, less bugs

Main Goals

- Sharing: don't reinvent the wheel ?
  - Algorithms,
  - Visualisation,
  - Data access...
- Ease the development of **your software**.
- Provide visibility to software.
- Modular:
  - ease of maintenance
  - increased reuse
  - = faster development, less bugs

Main Goals

- Sharing: don't reinvent the wheel ?
  - Algorithms,
  - Visualisation,
  - Data access...
- Ease the development of **your software**.
- Provide visibility to software.
- Modular:
  - ease of maintenance
  - increased reuse
  - = faster development, less bugs

Main Goals

- Sharing: don't reinvent the wheel ?
  - Algorithms,
  - Visualisation,
  - Data access...
- Ease the development of **your software**.
- Provide visibility to software.
- Modular:
  - ease of maintenance
  - increased reuse
  - = faster development, less bugs

Main Goals

- Sharing: don't reinvent the wheel ?
  - Algorithms,
  - Visualisation,
  - Data access...
- Ease the development of **your software**.
- Provide visibility to software.
- Modular:
  - ease of maintenance
  - increased reuse
  - = faster development, less bugs

Main features

- Support for C/C++,
- A build system: *cmake*,
- Modular environnement,
- Interactive shell: *TCL or Ruby (bindings with Swig)*,
- Errors reporting: *C++ exceptions*,
- Unit testing: *boost*,
- Documentation: *doxygen*.

Main features

- Support for C/C++,
- A build system: *cmake*,
- Modular environnement,
- Interactive shell: *TCL or Ruby (bindings with Swig)*,
- Errors reporting: *C++ exceptions*,
- Unit testing: *boost*,
- Documentation: *doxygen*.

Main features

- Support for C/C++,
- A build system: *cmake*,
- Modular environnement,
- Interactive shell: *TCL or Ruby (bindings with Swig)*,
- Errors reporting: *C++ exceptions*,
- Unit testing: *boost*,
- Documentation: *doxygen*.

Main features

- Support for C/C++,
- A build system: *cmake*,
- Modular environnement,
- Interactive shell: *TCL or Ruby (bindings with Swig)*,
- Errors reporting: *C++ exceptions*,
- Unit testing: *boost*,
- Documentation: *doxygen*.

Main features

- Support for C/C++,
- A build system: *cmake*,
- Modular environnement,
- Interactive shell: *TCL or Ruby (bindings with Swig)*,
- Errors reporting: *C++ exceptions*,
- Unit testing: *boost*,
- Documentation: *doxygen*.

Main features

- Support for C/C++,
- A build system: *cmake*,
- Modular environnement,
- Interactive shell: *TCL or Ruby (bindings with Swig)*,
- Errors reporting: *C++ exceptions*,
- Unit testing: *boost*,
- Documentation: *doxygen*.

Main features

- Support for C/C++,
- A build system: *cmake*,
- Modular environnement,
- Interactive shell: *TCL or Ruby (bindings with Swig)*,
- Errors reporting: *C++ exceptions*,
- Unit testing: *boost*,
- Documentation: *doxygen*.

Main features

- Support for C/C++,
- A build system: *cmake*,
- Modular environnement,
- Interactive shell: *TCL or Ruby (bindings with Swig)*,
- Errors reporting: *C++ exceptions*,
- Unit testing: *boost*,
- Documentation: *doxygen*.

Jafar and Genom

- Very similar: modules, interactive shell...
- They share the library part of the module (no file reading, no display, ...)
- $\Rightarrow$ Integration in Jafar for easy developement (scripts, demos)
- $\Rightarrow$ Integration in Genom for running on robots

Jafar and Genom

- Very similar: modules, interactive shell...
- They share the library part of the module (no file reading, no display, ...)
- ⇒ Integration in Jafar for easy developement (scripts, demos)
- ⇒ Integration in Genom for running on robots

Jafar and Genom

- Very similar: modules, interactive shell...
- They share the library part of the module
  (no file reading, no display, ...)
- $\Rightarrow$ Integration in Jafar for easy developement (scripts, demos)
- $\Rightarrow$ Integration in Genom for running on robots

Jafar and Genom

- Very similar: modules, interactive shell...
- They share the library part of the module
  (no file reading, no display, ...)
- $\Rightarrow$ Integration in Jafar for easy developement (scripts, demos)
- $\Rightarrow$ Integration in Genom for running on robots

Jafar Structure

- Directories
  - bin : various tools
  - modules : all the installed modules
  - doc : the documentation
  - build (or build_debug and build_release)
- Git repository
  - ssh://trac.laas.fr/git/robots/jafar/jafar : the core of jafar
  - ssh://trac.laas.fr/git/robots/jafar/modules : all the modules
- Documentation
  - **Wiki**: https://intranet.laas.fr/intranet/robots/wiki/Jafar
  - **Doxygen**: http://homepages.laas.fr/croussil/doc/jafar
  - **Doxygen**: $JAFAR_DIR/doc/html/index.html
  - **email**: jafar@laas.fr

Jafar Structure

- Directories
  - bin : various tools
  - modules : all the installed modules
  - doc : the documentation
  - build (or build_debug and build_release)
- Git repository
  - ssh://trac.laas.fr/git/robots/jafar/jafar : the core of jafar
  - ssh://trac.laas.fr/git/robots/jafar/modules : all the modules
- Documentation
  - **Wiki**: https://intranet.laas.fr/intranet/robots/wiki/Jafar
  - **Doxygen**: http://homepages.laas.fr/croussil/doc/jafar
  - **Doxygen**: $JAFAR_DIR/doc/html/index.html
  - **email**: jafar@laas.fr

Jafar Structure

- Directories
  - bin : various tools
  - modules : all the installed modules
  - doc : the documentation
  - build (or build_debug and build_release)
- Git repository
  - ssh://trac.laas.fr/git/robots/jafar/jafar : the core of jafar
  - ssh://trac.laas.fr/git/robots/jafar/modules : all the modules
- Documentation
  - **Wiki**: https://intranet.laas.fr/intranet/robots/wiki/Jafar
  - **Doxygen**: http://homepages.laas.fr/croussil/doc/jafar
  - **Doxygen**: $JAFAR_DIR/doc/html/index.html
  - **email**: jafar@laas.fr

Jafar Structure

- Directories
  - bin : various tools
  - modules : all the installed modules
  - doc : the documentation
  - build (or build_debug and build_release)
- Git repository
  - ssh://trac.laas.fr/git/robots/jafar/jafar : the core of jafar
  - ssh://trac.laas.fr/git/robots/jafar/modules : all the modules
- Documentation
  - **Wiki**: https://intranet.laas.fr/intranet/robots/wiki/Jafar
  - **Doxygen**: http://homepages.laas.fr/croussil/doc/jafar
  - **Doxygen**: $JAFAR_DIR/doc/html/index.html
  - **email**: jafar@laas.fr

Jafar Structure

- Directories
  - bin : various tools
  - modules : all the installed modules
  - doc : the documentation
  - build (or build_debug and build_release)
- Git repository
  - ssh://trac.laas.fr/git/robots/jafar/jafar : the core of jafar
  - ssh://trac.laas.fr/git/robots/jafar/modules : all the modules
- Documentation
  - **Wiki**: https://intranet.laas.fr/intranet/robots/wiki/Jafar
  - **Doxygen**: http://homepages.laas.fr/croussil/doc/jafar
  - **Doxygen**: $JAFAR_DIR/doc/html/index.html
  - **email**: jafar@laas.fr

Jafar Structure

- Directories
  - bin : various tools
  - modules : all the installed modules
  - doc : the documentation
  - build (or build_debug and build_release)
- Git repository
  - ssh://trac.laas.fr/git/robots/jafar/jafar : the core of jafar
  - ssh://trac.laas.fr/git/robots/jafar/modules : all the modules
- Documentation
  - **Wiki**: https://intranet.laas.fr/intranet/robots/wiki/Jafar
  - **Doxygen**: http://homepages.laas.fr/croussil/doc/jafar
  - **Doxygen**: $JAFAR_DIR/doc/html/index.html
  - **email**: jafar@laas.fr

Jafar Structure

- Directories
  - bin : various tools
  - modules : all the installed modules
  - doc : the documentation
  - build (or build_debug and build_release)
- Git repository
  - ssh://trac.laas.fr/git/robots/jafar/jafar : the core of jafar
  - ssh://trac.laas.fr/git/robots/jafar/modules : all the modules
- Documentation
  - **Wiki**: https://intranet.laas.fr/intranet/robots/wiki/Jafar
  - **Doxygen**: http://homepages.laas.fr/croussil/doc/jafar
  - **Doxygen**: $JAFAR_DIR/doc/html/index.html
  - **email**: jafar@laas.fr

Jafar Structure

- Directories
  - bin : various tools
  - modules : all the installed modules
  - doc : the documentation
  - build (or build_debug and build_release)
- Git repository
  - ssh://trac.laas.fr/git/robots/jafar/jafar : the core of jafar
  - ssh://trac.laas.fr/git/robots/jafar/modules : all the modules
- Documentation
  - **Wiki**: https://intranet.laas.fr/intranet/robots/wiki/Jafar
  - **Doxygen**: http://homepages.laas.fr/croussil/doc/jafar
  - **Doxygen**: $JAFAR_DIR/doc/html/index.html
  - **email**: jafar@laas.fr

Jafar Structure

- Directories
  - bin : various tools
  - modules : all the installed modules
  - doc : the documentation
  - build (or build_debug and build_release)
- Git repository
  - ssh://trac.laas.fr/git/robots/jafar/jafar : the core of jafar
  - ssh://trac.laas.fr/git/robots/jafar/modules : all the modules
- Documentation
  - **Wiki**: https://intranet.laas.fr/intranet/robots/wiki/Jafar
  - **Doxygen**: http://homepages.laas.fr/croussil/doc/jafar
  - **Doxygen**: $JAFAR_DIR/doc/html/index.html
  - **email**: jafar@laas.fr

Jafar Structure

- Directories
  - bin : various tools
  - modules : all the installed modules
  - doc : the documentation
  - build (or build_debug and build_release)
- Git repository
  - ssh://trac.laas.fr/git/robots/jafar/jafar : the core of jafar
  - ssh://trac.laas.fr/git/robots/jafar/modules : all the modules
- Documentation
  - **Wiki**: https://intranet.laas.fr/intranet/robots/wiki/Jafar
  - **Doxygen**: http://homepages.laas.fr/croussil/doc/jafar
  - **Doxygen**: $JAFAR_DIR/doc/html/index.html
  - **email**: jafar@laas.fr

Jafar Structure

- Directories
  - bin : various tools
  - modules : all the installed modules
  - doc : the documentation
  - build (or build_debug and build_release)
- Git repository
  - ssh://trac.laas.fr/git/robots/jafar/jafar : the core of jafar
  - ssh://trac.laas.fr/git/robots/jafar/modules : all the modules
- Documentation
  - **Wiki**: https://intranet.laas.fr/intranet/robots/wiki/Jafar
  - **Doxygen**: http://homepages.laas.fr/croussil/doc/jafar
  - **Doxygen**: $JAFAR_DIR/doc/html/index.html
  - **email**: jafar@laas.fr

Jafar Structure

- Directories
  - bin : various tools
  - modules : all the installed modules
  - doc : the documentation
  - build (or build_debug and build_release)
- Git repository
  - ssh://trac.laas.fr/git/robots/jafar/jafar : the core of jafar
  - ssh://trac.laas.fr/git/robots/jafar/modules : all the modules
- Documentation
  - **Wiki**: https://intranet.laas.fr/intranet/robots/wiki/Jafar
  - **Doxygen**: http://homepages.laas.fr/croussil/doc/jafar
  - **Doxygen**: $JAFAR_DIR/doc/html/index.html
  - **email**: jafar@laas.fr

Installation

https://intranet.laas.fr/intranet/robots/wiki/Jafar/Installation2

- Install tools dependencies (doxygen, swig, cmake, ...),
- Install external libraries dependencies (boost, opencv, ...),
- Install LAAS libraries dependencies for some modules (t3d, stereopixel, ...),
- Download Jafar (git clone)
- Configure the environment ($JAFAR_DIR, $LD_LIBRARY_PATH, ...)
- Configure the build system (cmake)

Installation

https://intranet.laas.fr/intranet/robots/wiki/Jafar/Installation2

- Install tools dependencies (doxygen, swig, cmake, ...),
- Install external libraries dependencies (boost, opencv, ...),
- Install LAAS libraries dependencies for some modules (t3d, stereopixel, ...),
- Download Jafar (git clone)
- Configure the environment ($JAFAR_DIR, $LD_LIBRARY_PATH, ...)
- Configure the build system (cmake)

Installation

https://intranet.laas.fr/intranet/robots/wiki/Jafar/Installation2

- Install tools dependencies (doxygen, swig, cmake, ...),
- Install external libraries dependencies (boost, opencv, ...),
- Install LAAS libraries dependencies for some modules (t3d, stereopixel, ...),
- Download Jafar (git clone)
- Configure the environment ($JAFAR_DIR, $LD_LIBRARY_PATH, ...)
- Configure the build system (cmake)

Installation

https://intranet.laas.fr/intranet/robots/wiki/Jafar/Installation2

- Install tools dependencies (doxygen, swig, cmake, ...),
- Install external libraries dependencies (boost, opencv, ...),
- Install LAAS libraries dependencies for some modules (t3d, stereopixel, ...),
- Download Jafar (git clone)
- Configure the environment ($JAFAR_DIR, $LD_LIBRARY_PATH, ...)
- Configure the build system (cmake)

Installation

https://intranet.laas.fr/intranet/robots/wiki/Jafar/Installation2

- Install tools dependencies (doxygen, swig, cmake, ...),
- Install external libraries dependencies (boost, opencv, ...),
- Install LAAS libraries dependencies for some modules (t3d, stereopixel, ...),
- Download Jafar (git clone)
- Configure the environment ($JAFAR_DIR, $LD_LIBRARY_PATH, ...)
- Configure the build system (cmake)

Installation

https://intranet.laas.fr/intranet/robots/wiki/Jafar/Installation2

- Install tools dependencies (doxygen, swig, cmake, ...),
- Install external libraries dependencies (boost, opencv, ...),
- Install LAAS libraries dependencies for some modules (t3d, stereopixel, ...),
- Download Jafar (git clone)
- Configure the environment ($JAFAR_DIR, $LD_LIBRARY_PATH, ...)
- Configure the build system (cmake)

The makefiles targets

At Jafar root or in the module:

- `make [all]`
- `make test`
- `make clean`
- `make install`
- `make rebuild_cache` : run cmake again if you added/removed files

At Jafar root only:

- `make doc`

The makefiles targets

At Jafar root or in the module:

- `make [all]`
- `make test`
- `make clean`
- `make install`
- `make rebuild_cache` : run cmake again if you added/removed files

At Jafar root only:

- `make doc`

The makefiles targets

At Jafar root or in the module:

- `make [all]`
- `make test`
- `make clean`
- `make install`
- `make rebuild_cache` : run cmake again if you added/removed files

At Jafar root only:

- `make doc`

The makefiles targets

At Jafar root or in the module:

- `make [all]`
- `make test`
- `make clean`
- `make install`
- `make rebuild_cache` : run cmake again if you added/removed files

At Jafar root only:

- `make doc`

The makefiles targets

At Jafar root or in the module:

- `make [all]`
- `make test`
- `make clean`
- `make install`
- `make rebuild_cache` : run cmake again if you added/removed files

At Jafar root only:

- `make doc`

The makefiles targets

At Jafar root or in the module:

- `make [all]`
- `make test`
- `make clean`
- `make install`
- `make rebuild_cache` : run cmake again if you added/removed files

At Jafar root only:

- `make doc`

Jafar tools

In $JAFAR_DIR/bin:

- `jafar_checkout`: download a module
- `jafar_update`: update jafar and all modules
- `git pull --rebase`: update jafar or a module
- `jafar_status`: display a summary of pending changes and commits
- `jafar_create`: create a new module
- `jafar_add`: add a new module to the repository

Jafar tools

In $JAFAR_DIR/bin:

- `jafar_checkout`: download a module
- `jafar_update`: update jafar and all modules
- `git pull --rebase`: update jafar or a module
- `jafar_status`: display a summary of pending changes and commits
- `jafar_create`: create a new module
- `jafar_add`: add a new module to the repository

Jafar tools

In $JAFAR_DIR/bin:

- `jafar_checkout`: download a module
- `jafar_update`: update jafar and all modules
- `git pull --rebase`: update jafar or a module
- `jafar_status`: display a summary of pending changes and commits
- `jafar_create`: create a new module
- `jafar_add`: add a new module to the repository

Jafar tools

In $JAFAR_DIR/bin:

- `jafar_checkout`: download a module
- `jafar_update`: update jafar and all modules
- `git pull --rebase`: update jafar or a module
- `jafar_status`: display a summary of pending changes and commits
- `jafar_create`: create a new module
- `jafar_add`: add a new module to the repository

Jafar tools

In $JAFAR_DIR/bin:

- `jafar_checkout`: download a module
- `jafar_update`: update jafar and all modules
- `git pull --rebase`: update jafar or a module
- `jafar_status`: display a summary of pending changes and commits
- `jafar_create`: create a new module
- `jafar_add`: add a new module to the repository

Jafar tools

In $JAFAR_DIR/bin:

- `jafar_checkout`: download a module
- `jafar_update`: update jafar and all modules
- `git pull --rebase`: update jafar or a module
- `jafar_status`: display a summary of pending changes and commits
- `jafar_create`: create a new module
- `jafar_add`: add a new module to the repository

Kernel module: Features

- Debug messages,
- Usefull macros,
- Configuration file,
- Timing tools,
- ...

Debug (1/2)

- DataLogger : to log data into a file
- Debug macro : JFR_DEBUG, JFR_VDEBUG, JFR_VVDEBUG, JFR_WARNING, JFR_ERROR

```
1 JFR_DEBUG(u << " + " << v << " = " << (u+v));
2 D:playmodule/file.cpp:709: Robot state after move [13
```

- JFR_ASSERT / JFR_PRED_ERROR : check that a parameter is correct

```
1 int MyFirstClass::div(int u, int v) const
2 {
3   JFR_ASSERT(v != 0, "Can't divide by 0");
4   return u / v;
5 }
```

Debug (1/2)

- DataLogger : to log data into a file
- Debug macro : JFR_DEBUG, JFR_VDEBUG, JFR_VVDEBUG, JFR_WARNING, JFR_ERROR

```
1 JFR_DEBUG(u << " + " << v << " = " << (u+v));
2 D:playmodule/file.cpp:709: Robot state after move [13
```

- JFR_ASSERT / JFR_PRED_ERROR : check that a parameter is correct

```
1 int MyFirstClass::div(int u, int v) const
2 {
3   JFR_ASSERT(v != 0, "Can't divide by 0");
4   return u / v;
5 }
```

Debug (1/2)

- DataLogger : to log data into a file
- Debug macro : JFR_DEBUG, JFR_VDEBUG, JFR_VVDEBUG, JFR_WARNING, JFR_ERROR

```
1 JFR_DEBUG(u << " + " << v << " = " << (u+v));
2 D:playmodule/file.cpp:709: Robot state after move [13
```

- JFR_ASSERT / JFR_PRED_ERROR : check that a parameter is correct

```
1 int MyFirstClass::div(int u, int v) const
2 {
3   JFR_ASSERT(v != 0, "Can't divide by 0");
4   return u / v;
5 }
```

Debug (2/2)

Verbosity level: Off, Trace, Warning, Debug, VerboseDebug, VeryVerboseDebug

```
1 kernel::DebugStream::setDefaultLevel(
2   kernel::DebugStream::Debug)
3 kernel::DebugStream::setLevel("playmodule",
4   kernel::DebugStream::VeryVerboseDebug);
5 kernel::DebugStream::setLevel("playmodule",
6   kernel::DebugStream::Off);
```

Usefull macros

For instance JFR_FOREACH:

```
1  std::vector< CoolObject > coolObjects;
2  JFR_FOREACH( CoolObject& coolObject, coolObjects )
3  {
4    coolObject.soSomethingCool();
5  }
```

Instead of:

```
1  std::vector< CoolObject > coolObjects;
2  for( std::vector< CoolObject >::iterator it
3       = coolObjects.begin();
4    it = coolObjects.end(); ++it )
5  {
6    it->soSomethingCool();
7  }
```

Configuration file (1/3)

Exemple of configuration file:

```
1 MyValue: 10
2 OtherValue: hello
```

Exemple of code to read file:

```
1 KeyValueFile configFile;
2 configFile.readFile( "test.cfg");
3 int val;
4 configFile.getItem( "MyValue", val );
5 std::string val2;
6 configFile.getItem( "OtherValue", val2 );
```

Configuration file (2/3)

KeyValueFileSave: an object which can save its parameters.

```
1  class CoolAlgorithm : public KeyValueFileSave {
2    public:
3      virtual void saveKeyValueFile(
4      jafar::kernel::KeyValueFile& keyValueFile)
5      {
6        keyValueFile.setItem("MyParameter", m_parameter );
7      }
8    public:
9      int m_parameter;
10 };
11
12 CoolAlgorithm coolAlgorithm;
13 coolAlgorithm.save("algo.cfg");
```

Configuration file (3/3)

KeyValueFileLoad: an object which can load its parameters.

```
1  class CoolAlgorithm : public KeyValueFileLoad {
2    public:
3      virtual void loadKeyValueFile(
4      jafar::kernel::KeyValueFile const& keyValueFile)
5      {
6        keyValueFile.getItem("MyParameter", m_parameter );
7      }
8    public:
9      int m_parameter;
10 };
11
12 CoolAlgorithm coolAlgorithm;
13 coolAlgorithm.load("algo.cfg");
```

Timing tools

- Chrono

```
1  Chrono chrono;
2  chrono.start();
3  // Do some extensive computation
4  JFR_DEBUG(chrono.elapsed());
```

- Framerate
- ...

Timing tools

- Chrono

```
1  Chrono chrono;
2  chrono.start();
3  // Do some extensive computation
4  JFR_DEBUG(chrono.elapsed());
```

- Framerate
- ...

Timing tools

- Chrono

```
1  Chrono chrono;
2  chrono.start();
3  // Do some extensive computation
4  JFR_DEBUG(chrono.elapsed());
```

- Framerate
- ...

What are unit tests ?

From wikipedia: *In computer programming, unit testing is a method of testing that verifies the individual units of source code are working properly.*

- Test the behavior of individual functions,
- As much as possible independent tests,
- Automatic.

What are unit tests ?

From wikipedia: *In computer programming, unit testing is a method of testing that verifies the individual units of source code are working properly.*

- Test the behavior of individual functions,
- As much as possible independent tests,
- Automatic.

What are unit tests ?

From wikipedia: *In computer programming, unit testing is a method of testing that verifies the individual units of source code are working properly.*

- Test the behavior of individual functions,
- As much as possible independent tests,
- Automatic.

Why unit tests are important ?

- Make sure your code does what you want it to do,
- Speed up development and optimizations/refactoring,
- Make sure nobody else breaks your feature,
- Tests are documentation.

Why unit tests are important ?

- Make sure your code does what you want it to do,
- Speed up development and optimizations/refactoring,
- Make sure nobody else breaks your feature,
- Tests are documentation.

Why unit tests are important ?

- Make sure your code does what you want it to do,
- Speed up development and optimizations/refactoring,
- Make sure nobody else breaks your feature,
- Tests are documentation.

Why unit tests are important ?

- Make sure your code does what you want it to do,
- Speed up development and optimizations/refactoring,
- Make sure nobody else breaks your feature,
- Tests are documentation.

Write an unit test.

Add a file test_suite/test_MyFirstClass.cpp :

```
1  #include <boost/test/auto_unit_test.hpp>
2  #include <kernel/jafarTestMacro.hpp>
3  #include "playmodule/MyFirstClass.hpp"
4
5  BOOST_AUTO_TEST_CASE( test_MyFirstClass )
6  {
7     MyFirstClass mfc;
8     JFR_CHECK_EQUAL( mfc.add(1, 2), 3 );
9  }
```

Then:

```
1  make test
```

Documentation: a brief introduction to doxygen

Comments syntax:

```
1 /** mutliple lines comment */
2 /// single line comment
3 ///< post-code comment
```

General tags:

- **@ingroup** declare a function to be part of a module
- **@ref** give a reference to an other function/class

Function tags:

- **@param** describe a parameter
- **@return** describe the return parameter

Lets document MyFirstClass

```
1  /**
2   * This is my first class in Jafar. @ref add is
3   * the most important function.
4   * @ingroup playmodule
5   */
6  class MyFirstClass {
7  int data; ///< this is the data
8    public:
9      /**
10      * This function add two numbers.
11      * @param u first number
12      * @param v second number
13      * @return the addition of u with v
14      */
15     int add(int u, int v) const;
16 };
```

A Jafar module is...

- A C/C++ library
  - headers in playmodule/**include**/playmodule/*.hpp
  - sources in playmodule/**src**/*.hpp
- A set of tcl/ruby scripts and/or C/C++ demos
  - scripts in playmodule/**macro**/*.rb or *.tcl
  - demos in playmodule/**demo_suite**/demo_*.cpp
- Documentation
  - in playmodule/**doc**/*.doxy
  - **doxygen comments in headers files**
- Unit tests
  - in playmodule/**test_suite**/test_*.cpp

A Jafar module is...

- A C/C++ library
  - headers in playmodule/**include**/playmodule/*.hpp
  - sources in playmodule/**src**/*.hpp
- A set of tcl/ruby scripts and/or C/C++ demos
  - scripts in playmodule/**macro**/*.rb or *.tcl
  - demos in playmodule/**demo_suite**/demo_*.cpp
- Documentation
  - in playmodule/**doc**/*.doxy
  - **doxygen comments in headers files**
- Unit tests
  - in playmodule/**test_suite**/test_*.cpp

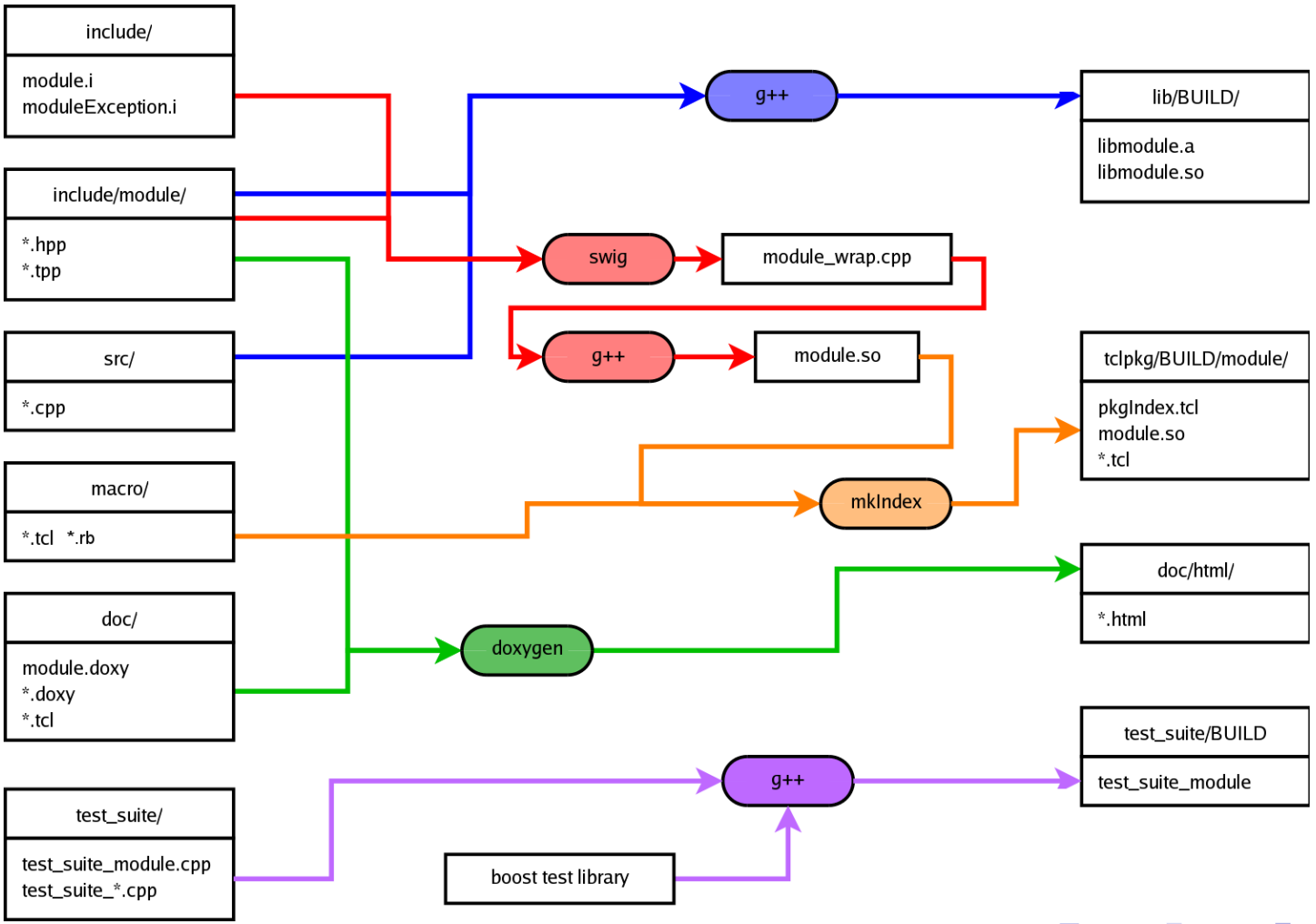
A Jafar module is...

- A C/C++ library
  - headers in playmodule/**include**/playmodule/*.hpp
  - sources in playmodule/**src**/*.hpp
- A set of tcl/ruby scripts and/or C/C++ demos
  - scripts in playmodule/**macro**/*.rb or *.tcl
  - demos in playmodule/**demo_suite**/demo_*.cpp
- Documentation
  - in playmodule/**doc**/*.doxy
  - **doxygen comments in headers files**
- Unit tests
  - in playmodule/**test_suite**/test_*.cpp

A Jafar module is...

- A C/C++ library
  - headers in playmodule/**include**/playmodule/*.hpp
  - sources in playmodule/**src**/*.hpp
- A set of tcl/ruby scripts and/or C/C++ demos
  - scripts in playmodule/**macro**/*.rb or *.tcl
  - demos in playmodule/**demo_suite**/demo_*.cpp
- Documentation
  - in playmodule/**doc**/*.doxy
  - **doxygen comments in headers files**
- Unit tests
  - in playmodule/**test_suite**/test_*.cpp

Directory structure



How to create a module ?

Locally create the module:

```
1 cd ${JAFAR_DIR}/modules
2 ../bin/jafar_create playmodule
```

Commit and push the initial version:

```
1 ../bin/jafar_add playmodule
```

A tour inside the new module

- **CMakeLists.txt** (dependencies)
- **include/playmodule.i** (swig additional wrapping)
- **include/playmoduleException.hpp** (exceptions)
- **src/playmoduleException.cpp** (exceptions)
- **macro/** (macros)
- **test_suite/** (unit tests)
- **demo_suite/** (demos)
- **doc/** (documentation)

A tour inside the new module

- **CMakeLists.txt** (dependencies)
- **include/playmodule.i** (swig additional wrapping)
- **include/playmoduleException.hpp** (exceptions)
- **src/playmoduleException.cpp** (exceptions)
- **macro/** (macros)
- **test_suite/** (unit tests)
- **demo_suite/** (demos)
- **doc/** (documentation)

A tour inside the new module

- **CMakeLists.txt** (dependencies)
- **include/playmodule.i** (swig additional wrapping)
- **include/playmoduleException.hpp** (exceptions)
- **src/playmoduleException.cpp** (exceptions)
- **macro/** (macros)
- **test_suite/** (unit tests)
- **demo_suite/** (demos)
- **doc/** (documentation)

A tour inside the new module

- **CMakeLists.txt** (dependencies)
- **include/playmodule.i** (swig additional wrapping)
- **include/playmoduleException.hpp** (exceptions)
- **src/playmoduleException.cpp** (exceptions)
- **macro/** (macros)
- **test_suite/** (unit tests)
- **demo_suite/** (demos)
- **doc/** (documentation)

A tour inside the new module

- **CMakeLists.txt** (dependencies)
- **include/playmodule.i** (swig additional wrapping)
- **include/playmoduleException.hpp** (exceptions)
- **src/playmoduleException.cpp** (exceptions)
- **macro/** (macros)
- **test_suite/** (unit tests)
- **demo_suite/** (demos)
- **doc/** (documentation)

A tour inside the new module

- **CMakeLists.txt** (dependencies)
- **include/playmodule.i** (swig additional wrapping)
- **include/playmoduleException.hpp** (exceptions)
- **src/playmoduleException.cpp** (exceptions)
- **macro/** (macros)
- **test_suite/** (unit tests)
- **demo_suite/** (demos)
- **doc/** (documentation)

A tour inside the new module

- **CMakeLists.txt** (dependencies)
- **include/playmodule.i** (swig additional wrapping)
- **include/playmoduleException.hpp** (exceptions)
- **src/playmoduleException.cpp** (exceptions)
- **macro/** (macros)
- **test_suite/** (unit tests)
- **demo_suite/** (demos)
- **doc/** (documentation)

A tour inside the new module

- **CMakeLists.txt** (dependencies)
- **include/playmodule.i** (swig additional wrapping)
- **include/playmoduleException.hpp** (exceptions)
- **src/playmoduleException.cpp** (exceptions)
- **macro/** (macros)
- **test_suite/** (unit tests)
- **demo_suite/** (demos)
- **doc/** (documentation)

Compilation

- Compile it

```
1 make
```

- Load it

```
1 require 'jafar/playmodule'
```

- So what is available...

Compilation

- Compile it

```
1 make
```

- Load it

```
1 require 'jafar/playmodule'
```

  - So what is available...

Compilation

- Compile it

```
1 make
```

- Load it

```
1 require 'jafar/playmodule'
```

- So what is available...

A first class: header

```
1  #ifndef _MY_FIRST_CLASS_
2  #define _MY_FIRST_CLASS_
3  namespace jafar {
4    namespace playmodule {
5      class MyFirstClass {
6        public:
7          int add(int u, int v) const;
8      };
9    }
10 }
11 #endif
```

A first class: source

```
1  #include "playmodule/MyFirstClass.hpp"
2
3  using namespace jafar::playmodule;
4
5  int MyFirstClass::add(int u, int v) const
6  {
7    return u + v;
8  }
```

Bind it

Open file *playmodule.i* and add:

```
1 #include "playmodule/MyFirstClass.hpp"
```

And:

```
1 %include "playmodule/MyFirstClass.i"
```

Use it

```
1  require 'jafar/playmodule'
2  obj = Playmodule::MyFirstClass.new
3  obj.add( 1, 2)
```

jmath: Features

- uses BLAS/LAPACK
- matrices and vectors computation
- linear solvers
- least-square optimization

Create vectors and matrix

- Bounded vectors

```
1 jblas::vec2 vec_2;
2 jblas::vec3 vec_3;
3 jblas::vec4 vec_4;
```

- Unbounded vectors
- Bounded matrix
- Unbounded matrix
- Zero, Scalar and Identity matrix

Create vectors and matrix

- Bounded vectors
- Unbounded vectors

```
1  jblas::vec vec_n( 10 );
```

- Bounded matrix
- Unbounded matrix
- Zero, Scalar and Identity matrix

Create vectors and matrix

- Bounded vectors
- Unbounded vectors
- Bounded matrix

```
1 jblas::mat22 mat_22;
2 jblas::mat33 mat_33;
3 jblas::mat44 mat_44;
```

- Unbounded matrix
- Zero, Scalar and Identity matrix

Create vectors and matrix

- Bounded vectors
- Unbounded vectors
- Bounded matrix
- Unbounded matrix

```
1 jblas::mat mat_nn(100,400);
```

- Zero, Scalar and Identity matrix

Create vectors and matrix

- Bounded vectors
- Unbounded vectors
- Bounded matrix
- Unbounded matrix
- Zero, Scalar and Identity matrix

```
1 jblas::mat mat = jblas::zero_mat(5);
2 jblas::mat mat = jblas::identity_mat(5);
3 jblas::mat mat = 5.0 * jblas::scalar_mat(5); // Matr
```

Vectors and matrix operations

- Addition, substraction

```
1  jblas::vec2 v1, v2, v3;
2  v1 = v1 + v2 - v3;
```

- Multiplication
- Transposition
- Inversion
- Dot and cross product
- Determinant

Vectors and matrix operations

- Addition, substraction
- Multiplication

```
1  jblas::mat m1, m2, m3;
2  m3 = ublas::prod( m1, m2 );
3  m3 = ublas::prod( m1,
4  jblas::mat( ublas::prod( m3, m2 ) ) );
```

- Transposition
- Inversion
- Dot and cross product
- Determinant

Vectors and matrix operations

- Addition, substraction
- Multiplication
- Transposition

```
1  jblas::mat m4 = ublas::trans(m3);
2  m4 = ublas::prod( ublas::trans(m2), m1 );
```

- Inversion
- Dot and cross product
- Determinant

Vectors and matrix operations

- Addition, substraction
- Multiplication
- Transposition
- Inversion

```
1 ublasExtra::inv( m4 );
```

- Dot and cross product
- Determinant

Vectors and matrix operations

- Addition, substraction
- Multiplication
- Transposition
- Inversion
- Dot and cross product

```
1 ublas::outer_prod( v1, v2 ); // cross product
2 ublas::inner_prod( v1, v2 ); // dot product
```

- Determinant

Vectors and matrix operations

- Addition, substraction
- Multiplication
- Transposition
- Inversion
- Dot and cross product
- Determinant

```
1 ublasExtra::det( m4 );
```

Symmetric matrix

- Bounded symmetric matrix:

```
1 jblas::sym_mat22 mat_22;
2 jblas::sym_mat33 mat_33;
3 jblas::sym_mat44 mat_44;
```

- Unbounded symmetric matrix:
- Create symmetrix matrix from non symmetric matrix
- Access elements

Symmetric matrix

- Bounded symmetric matrix:
- Unbounded symmetric matrix:

```
1 jblas::sym_mat mat_nn(100);
```

- Create symmetrix matrix from non symmetric matrix
- Access elements

Symmetric matrix

- Bounded symmetric matrix:
- Unbounded symmetric matrix:
- Create symmetrix matrix from non symmetric matrix

```
1  jblas::sym_mat mat_nn(100);
2  jblas::sym_mat smat_nn =
3      ublas::symmetric_adaptor<jblas::mat44,
4          ublas::lower>( mat_nn );
5  jblas::sym_mat smat_nn =
6      ublas::symmetric_adaptor<jblas::mat44,
7          ublas::upper>( mat_nn );
```

- Access elements

Symmetric matrix

- Bounded symmetric matrix:
- Unbounded symmetric matrix:
- Create symmetrix matrix from non symmetric matrix
- Access elements

```
1  jblas::sym_mat22 mat_22;
2  mat_22(0,1) = 10.0;
3  // Warning:
4  mat_22(1,0) = 12.0;
5  JFR_DEBUG( mat_22(1,0) ); // will display 10.0 !
```

Use lapack

- To compute SVD and eigen values
- Warning: use column major with Lapack

```
1 jblas::mat A( 30, 3 );
2 jblas::mat_column_major m_A( A );
3 jblas::vec s(3);
4 jblas::mat_column_major U(30, 3);
5 jblas::mat_column_major VT(3, 3);
6
7 int ierr = lapack::gesdd(m_A,s,U,VT);
```

Use lapack

- To compute SVD and eigen values
- Warning: use column major with Lapack

```
1  jblas::mat A( 30, 3 );
2  jblas::mat_column_major m_A( A );
3  jblas::vec s(3);
4  jblas::mat_column_major U(30, 3);
5  jblas::mat_column_major VT(3, 3);
6
7  int ierr = lapack::gesdd(m_A,s,U,VT);
```

Linear least square

- Find x that minimize $\|A.x - b\|^2$
- LinearLeastSquares
- VariableSizeLinearLeastSquares

Linear least square

- Find x that minimize $\|A.x - b\|^2$
- LinearLeastSquares

```
1  LinearLeastSquares lls;
2  lls.setSize( 3 /* model size */,
3               10 /* number of points */ );
4  jblas::vec valueOfA;
5  double valueOfB;
6  lls.setData( 0 /* index of point */,
7               valueOfA,
8               valueOfB );
9  ...
10 lls.solve();
11 lls.x(); // return the value of x
12 lls.xCov(); // return the covariance
```

- VariableSizeLinearLeastSquares

Linear least square

- Find x that minimize $\|A.x - b\|^2$
- LinearLeastSquares
- VariableSizeLinearLeastSquares

```
1  VariableSizeLinearLeastSquares vsll
2                ( 3 /* model size */ );
3  jblas::vec valueOfA;
4  double valueOfB;
5  vsll.addMeasure( valueOfA, valueOfB );
6  ...
7  vsll.solve();
8  vsll.x(); // return the value of x
```

geom: Features

- T3D: 3D Transformation
- Geometric classes such as Point, Lines, Boxes...

geom: Features

- T3D: 3D Transformation
- Geometric classes such as Point, Lines, Boxes...

T3D: 3D Transformation

- Support for Euler and Quaternion

```
1  jblas::vec transfoX(6);
2  transfoX(0) = x;
3  transfoX(1) = y;
4  transfoX(2) = z;
5  transfoX(3) = yaw;
6  transfoX(4) = pitch;
7  transfoX(5) = roll;
8  geom::T3DEuler transfo( transfoX );
```

- Composition
- Invert
- Transform a vector

T3D: 3D Transformation

- Support for Euler and Quaternion
- Composition

```
1  geom::T3DEuler robotToWorld = something;
2  geom::T3DEuler sensorToRobot = something;
3  geom::T3DEuler sensorToWorld;
4  geom::T3D::compose( sensorToRobot,
5                      robotToWorld,
6                      sensorToWorld);
```

- Invert
- Transform a vector

T3D: 3D Transformation

- Support for Euler and Quaternion
- Composition
- Invert

```
1 geom::T3DEuler robotToWorld = something;
2 geom::T3DEuler worldToRobot;
3 geom::T3D::invert( robotToWorld, worldToRobot );
```

- Transform a vector

T3D: 3D Transformation

- Support for Euler and Quaternion
- Composition
- Invert
- Transform a vector

```
1 geom::T3DEuler sensorToWorld = something;
2 jblas::vec X_sensor;
3 jblas::vec X_world = ublas::prod( sensorToWorld.getM
```

Geometric classes

- Points

```
1  jblas::vec v = coordinates of the point;
2  geom::Point<3> p1(
3                 new Point<3>::EuclideanDriver( v ) );
```

- Lines
- Segments, polylines, planes, facets...
- Operations
- Bounding box

Geometric classes

- Points
- Lines

```
1  jblas::vec v = coordinates of the origin;
2  jblas::vec v = coordinates of the vector director;
3  geom::Line<3> l1(
4                  new Line<3>::EuclideanDriver( v ) );
5  geom::Point<3> p1;
6  geom::Point<3> p2;
7  geom::Line<3> l2(
8      new Line<3>::TwoPointsPointerDriver( &p1, &p2 )
9  geom::Line<3> l2(
10                 new Line<3>::TwoPointsDriver( p1, p2 )
```

- Segments, polylines, planes, facets...
- Operations
- Bounding box

Geometric classes

- Points
- Lines
- Segments, polylines, planes, facets...
- Operations
- Bounding box

Geometric classes

- Points
- Lines
- Segments, polylines, planes, facets...
- Operations

```
1  geom::distance( p1, p2 );
2  geom::distance( p1, l2 );
3  geom::angle( l1, l2 );
```

- Bounding box

Geometric classes

- Points
- Lines
- Segments, polylines, planes, facets...
- Operations
- Bounding box

```
1  geom::Segment segment( {1,1,1}, {-1,-1,-1} );
2  segment.boundingBox(); // Return {1,1,1}, {-1,-1,-1}
```

Geometric classes : VoxelSpace (1/2)

```
1  class Object
2  {
3    public:
4      Object(const geom::Atom<3>& atom_)
5        : m_atom(atom_)
6      {
7      }
8      const geom::Atom<3>& atom() const
9      { return m_atom; }
10   private:
11     const geom::Atom<3>& m_atom;
12 };
```

Geometric classes : VoxelSpace (2/2)

```
1  geom::VoxelSpace<dimension, Object,
2       geom::AtomBoundingBoxGetter<dimension, Object> >
3       voxelSpace;
4  geom::Point<3> pt;
5  Object* obj1 = new Object(pt);
6  voxelSpace.insertObject( obj1 );
7  geom::Line<3> li;
8  Object* obj2 = new Object(li);
9  voxelSpace.insertObject( obj2 );
10 geom::BoundingBox<3> bb( onePoint, oneAnotherPoint );
11 std::list<Object*> objects =
12                          voxelSpace.objectsIn( bb );
```

image: Features

- load/read images
- access to the whole OpenCV API
  an `image::Image` object can be used as a `cv::Mat`

image: Features

- load/read images
- access to the whole OpenCV API
  an `image::Image` object can be used as a `cv::Mat`

Features

- Create an image

```
1 image::Image dx(width, height, CV_8U, JfrImage_CS_GRA
```

- Use a functin from OpenCV

```
1 image::Image myImage;
2 myImage.loadImage("MyFile.png");
3 cvSobel(myImage, dx, 1, 0, 3);
```

Features

- Create an image

```
1 image::Image dx(width, height, CV_8U, JfrImage_CS_GRA
```

- Use a functin from OpenCV

```
1 image::Image myImage;
2 myImage.loadImage("MyFile.png");
3 cvSobel(myImage, dx, 1, 0, 3);
```

camera: Features

Camera models:

- Pinhole, Barreto (omni), Stereo bench
- project, jacobians...

datareader: Features

Ease the use of data respecting the organization of the pelican/tic server

- Read calibration infos
- Read images and preprocess them
- Read position infos

datareader: Features

Ease the use of data respecting the organization of the pelican/tic server

- Read calibration infos
- Read images and preprocess them
- Read position infos

datareader: Features

Ease the use of data respecting the organization of the pelican/tic server

- Read calibration infos
- Read images and preprocess them
- Read position infos

Default configuration

- set the base directory

```
1 Jafar::Datareader::DataReader
2     .setDefaultBasePath("~/laas/data")
```

- set the series name

```
1 Jafar::Datareader::DataReader
2     .setDefaultSeriesName("2010-11-28_grande-salle")
```

- set the serie number

```
1 Jafar::Datareader::DataReader
2     .setDefaultSerieNumber(11)
```

Default configuration

- set the base directory

```
1 Jafar::Datareader::DataReader
2     .setDefaultBasePath("~/laas/data")
```

- set the series name

```
1 Jafar::Datareader::DataReader
2     .setDefaultSeriesName("2010-11-28_grande-salle")
```

- set the serie number

```
1 Jafar::Datareader::DataReader
2     .setDefaultSerieNumber(11)
```

Default configuration

- set the base directory

```
1 Jafar::Datareader::DataReader
2     .setDefaultBasePath("~/laas/data")
```

- set the series name

```
1 Jafar::Datareader::DataReader
2     .setDefaultSeriesName("2010-11-28_grande-salle")
```

- set the serie number

```
1 Jafar::Datareader::DataReader
2     .setDefaultSerieNumber(11)
```

Read data

```
1  dr = Datareader::DataReader.new
2  sr = dr.getStereoReader( 0 )
3  img = sr.left.loadImage( 0 )
```

qdisplay: Features

- Uses QT
- Displays images
- Displays vector graphics overlay

qdisplay: Features

- Uses QT
- Displays images
- Displays vector graphics overlay

qdisplay: Features

- Uses QT
- Displays images
- Displays vector graphics overlay

Use

```
1  dr = Datareader::DataReader.new
2  sr = dr.getStereoReader( 0 )
3  imgL = sr.left.loadImage( 0 )
4  imgR = sr.right.loadImage( 0 )
5
6  viewer = Jafar::Qdisplay::Viewer.new
7  imageviewL = Jafar::Qdisplay::ImageView.new(imgL)
8  viewer.setImageView(imageviewL)
9  imageviewR = Jafar::Qdisplay::ImageView.new(imgR)
10 viewer.setImageView(imageviewR, 1, 0)
11
12 shape = Qdisplay::Shape.new(Qdisplay::Shape::ShapeRectan
13   10, 10, 5, 5)
14 shape.setColor(0,255,0)
15 shape.setLabel("Hello World!")
16 imageviewL.addShape(shape)
```

gdhe: Features

- Easy interface to be a client of GDHE
- GDHE objects are wrapped to C++ objects

Example:

```
1  gdhe::Client client;
2  client.connect("localhost");
3
4  gdhe::Ellipsoid *ell = new gdhe::Ellipsoid(x,xCov,3.);
5  ell->setLabel("12");
6  ell->setColor(255,0,0);
7  ell->setLabelColor(255,0,0);
8
9  client.addObject(ell);
```

gdhe: Features

- Easy interface to be a client of GDHE
- GDHE objects are wrapped to C++ objects

Example:

```
1  gdhe::Client client;
2  client.connect("localhost");
3
4  gdhe::Ellipsoid *ell = new gdhe::Ellipsoid(x,xCov,3.);
5  ell->setLabel("12");
6  ell->setColor(255,0,0);
7  ell->setLabelColor(255,0,0);
8
9  client.addObject(ell);
```

Algorithms modules

Lower or higher level:

- Image Processing: correl, fdetect (harris, sift, surf, star), gfm, klt, dseg, jstereopixel...
- Optimization/Estimation: filter, jbn, localizer, bundle, ddf, oracle ...
- Localization/Modeling: slam, vme, dtm, ...

And yours !

Conclusion

Jafar helps you:

- Provides framework and tools,
- Provides algorithms,
- Provides visibility to your software.

But you also have to help Jafar:

- before implementing, check if it already exists,
- put the stuff you create in the right module (functional separation),
- adopt standard coding: wiki://Jafar/Development/Rules
  or more generally:
  http://www.possibility.com/Cpp/CppCodingStandard.html
- document your work
- write unit tests

Conclusion

Jafar helps you:

- Provides framework and tools,
- Provides algorithms,
- Provides visibility to your software.

But you also have to help Jafar:

- before implementing, check if it already exists,
- put the stuff you create in the right module (functional separation),
- adopt standard coding: wiki://Jafar/Development/Rules
  or more generally:
  http://www.possibility.com/Cpp/CppCodingStandard.html
- document your work
- write unit tests

Conclusion

Jafar helps you:

- Provides framework and tools,
- Provides algorithms,
- Provides visibility to your software.

But you also have to help Jafar:

- before implementing, check if it already exists,
- put the stuff you create in the right module (functional separation),
- adopt standard coding: wiki://Jafar/Development/Rules
  or more generally:
  http://www.possibility.com/Cpp/CppCodingStandard.html
- document your work
- write unit tests

Conclusion

Jafar helps you:

- Provides framework and tools,
- Provides algorithms,
- Provides visibility to your software.

But you also have to help Jafar:

- before implementing, check if it already exists,
- put the stuff you create in the right module (functional separation),
- adopt standard coding: wiki://Jafar/Development/Rules
  or more generally:
  http://www.possibility.com/Cpp/CppCodingStandard.html
- document your work
- write unit tests

Conclusion

Jafar helps you:

- Provides framework and tools,
- Provides algorithms,
- Provides visibility to your software.

But you also have to help Jafar:

- before implementing, check if it already exists,
- put the stuff you create in the right module (functional separation),
- adopt standard coding: wiki://Jafar/Development/Rules
  or more generally:
  http://www.possibility.com/Cpp/CppCodingStandard.html
- document your work
- write unit tests

Conclusion

Jafar helps you:

- Provides framework and tools,
- Provides algorithms,
- Provides visibility to your software.

But you also have to help Jafar:

- before implementing, check if it already exists,
- put the stuff you create in the right module (functional separation),
- adopt standard coding: wiki://Jafar/Development/Rules
  or more generally:
  http://www.possibility.com/Cpp/CppCodingStandard.html
- document your work
- write unit tests

Conclusion

Jafar helps you:

- Provides framework and tools,
- Provides algorithms,
- Provides visibility to your software.

But you also have to help Jafar:

- before implementing, check if it already exists,
- put the stuff you create in the right module (functional separation),
- adopt standard coding: wiki://Jafar/Development/Rules
  or more generally:
  http://www.possibility.com/Cpp/CppCodingStandard.html
- document your work
- write unit tests

Conclusion

Jafar helps you:

- Provides framework and tools,
- Provides algorithms,
- Provides visibility to your software.

But you also have to help Jafar:

- before implementing, check if it already exists,
- put the stuff you create in the right module (functional separation),
- adopt standard coding: wiki://Jafar/Development/Rules
  or more generally:
  http://www.possibility.com/Cpp/CppCodingStandard.html
- document your work
- write unit tests