

# QT-CAML, milestone 1 : a binding to GCC-XML

The QT-CAML team

April 16, 2008

## 1 Introduction

QT [Trolltech] is a cross-platform rich client development framework, which is one of the main success of the open source community. Indeed, QT has been chosen by 4500 companies (Adobe, Cadence, Skype, Google, ...) because of its well-known reliability, comprehensiveness and quality of conception.

For the moment, the OBJECTIVE CAML community cannot take benefit of this framework because no binding has been released so far. We are aware of the technical difficulties that make this binding hard to develop (see 2.1). To tackle these problems one by one, we drew a *gradual* roadmap (see 2.2) which will lead to a **semi-automatic binding generator for Qt in Objective Caml**. Each milestone is a self-contained project that will improve the interoperability between OBJECTIVE CAML and C++. This proposal corresponds to the first milestone of the QT-CAML project : **a binding to GCC-XML** [King], an interface to the front-end of GCC.

## 2 Project

### 2.1 Technical difficulties

In this section, we briefly describe the main technical difficulties someone faces to develop a deep binding between a C++ library and OBJECTIVE CAML.

**Two object systems** C++ is an object-oriented programming language. A realistic binding to a C++ library should use the object-oriented features of OBJECTIVE CAML to offer a canonical interface to the C++ API. Furthermore, in many C++ libraries (and QT is one of them), the client must inherit from a given class to integrate a service in its application. This idiom must be usable from OBJECTIVE CAML too : in other words, an OBJECTIVE CAML class must be able to inherit from a C++ class. The stub to be written is quite complex because late binding must be handled correctly. Furthermore, C++ handles *adhoc* polymorphism whereas OBJECTIVE CAML does not: some automatic and smart renaming of method names have to be done. Clearly, a hand-crafted is not conceivable: we must write a **reliable stub generator**.

**Two runtime systems** C++ uses explicit data deallocation whereas OBJECTIVE CAML uses a garbage collector. The ownership of data structures must be tracked between these two worlds both to prevent the garbage collector to release a chunk held by C++ and to forbid C++ to free a custom block held by OBJECTIVE CAML. The deallocation policy of QT is well-defined but the behavior of

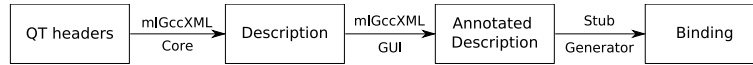


Figure 1: Design of the QT-CAML project.

classes is not homogeneous in the whole hierarchy. Thus, it is necessary to **annotate the API** to determine what deallocation strategy must be followed locally by the stub.

**Need for a scalable solution** QT has a very impressive number of services and the frequency of its releases is high. In order to minimize the maintenance and porting work, we have to make the hand-written code as small as possible. Thus, an **automatic analysis** of QT headers, an **automatic merge** of our annotations to these headers and an **automatic generation** of the stub are necessary.

## 2.2 Overview of the Qt-Caml project

The design of the QT-CAML project is meant to face separately the problems described in the previous section. The overall structure is illustrated in the figure 2.2.

QT headers are parsed using GCC-XML [King]. The core of MLGCCXML is intended to provide an abstraction to the raw XML file containing the C++ declarations. A Graphical User Interface (GUI) will enable the annotation process of the QT API. The resulting annotated description will be processed by a stub generator to obtain the final binding.

## 2.3 Task 1: mlGccXml- Core

The core of MLGCCXML will provide a way to navigate into the QT hierarchies in the spirit of pygccxml [Yakovenko], except that it will be based on OBJECTIVE CAML pattern matching instead of the visitor design pattern. The core MLGCCXML will also integrate a cache system to avoid the expensive parsing of the big amount of XML representing QT API. Finally, the core MLGCCXML will offer a way to merge external annotations and QT declarations. This merging process must be robust with respect to the evolution of the QT API.

Given these wishes, we have decomposed this task into several sub-tasks :

1. Define a subset of the GCC-XML format adapted to our purpose ;
2. Design the core API to navigate into C++ declarations ;
3. Implement the parser and the navigation API ;
4. Implement the caching system ;
5. Design a generic format for external annotations ;
6. Implement a merging algorithm between the external annotations and the C++ declarations.

This development will be validated by a hand-written test suite (composed of unit tests and of realistic tests on QT API).

## 2.4 Task 2: mlGccXml– GUI

The role of the GUI is to facilitate the annotation process. To achieve that purpose, the GUI will give a hierarchical view of the QT headers and will superpose the external annotations on them. This interface will also permit to review QT code in order to determine quickly the annotations associated to a particular C++ class.

Here is the sub-tasks decomposition of this work :

1. Design a graphical application to navigate into C++ declarations ;
2. Implement this application ;
3. Extend the design to annotate the hierarchies ;
4. Implement the annotation widgets.

## 2.5 Task 3: a proof-of-concept

Given the remaining time, the last task will consist in an application of the previously described tools to build a binding of (a small fragment of) QT. This last development will be decomposed into the following sub-tasks :

1. Implement a proof-of-concept stub generator ;
2. Use it to generate a binding ;
3. Use this binding to develop a QT application in OBJECTIVE CAML;
4. Exhibit the technical difficulties into a test suite.

## 2.6 Deliverables

The development will be done in a transparent way using an open source development framework. The TRAC system is publicly available at :

<http://qt-caml.crapulion.org>

## 2.7 Deliverable 1: mlGccXml– Core

The students will deliver at the end of the summer:

- a technical report explaining the design choices ;
- a documented OBJECTIVE CAML library named MLGCCXML;
- a test suite and all their development history.

The software will run under Linux and MS/Windows using the version 3.10 of OBJECTIVE CAML. It will be released under the BSD license. The development version of GCC-XML will be required.

## 2.8 Deliverable 2: mlGccXml- GUI

The students will deliver at the end of the summer:

- a documented graphical user interface ;
- a partially annotated description of the QT API.

The software will run under Linux and MS/Windows using the version 3.10 of OBJECTIVE CAML. It will be released under the BSD license. The version 2.10.1 of labgtk will be required.

## 2.9 Deliverable 3: a proof-of-concept

The students will deliver at the end of the summer:

- a binding to a very small fragment of the QT API ;
- a technical report and a test-suite explaining the technical difficulties related to this binding and what kind of annotations helps solving them.

The software will run under Linux and MS/Windows using the version 3.10 of OBJECTIVE CAML. The version 4.3 of QT will be required. This piece of code is not meant to be released.

# 3 Participants

## 3.1 Mentor

Yann Régis-Gianas is a postdoc at INRIA Saclay-Île de France. He did his PhD in the Gallium project at INRIA Paris-Rocquencourt.

**email** : yann.regis-gianas@inria.fr  
**address** : INRIA Saclay-Île de France,  
Parc Orsay Université,  
4 rue J. Monod,  
91893 Orsay Cedex France.

## 3.2 Students

Guillem Rieu is an undergraduate student in computer science, currently in third year at Université Paris-Diderot (Paris 7), with strong interest in functional programming.

**email** : guillemr@gmx.net  
**address** : 9 bis rue de la Paix,  
94300 Vincennes France.

Vincent Bernardoff is an undergraduate student in physics (L3 level, third year) at Université Paris-Diderot, with strong interest in computer science and some experience with OCaml, C++ and Qt.

**email** : vbernard@fdn.fr  
**address** : 31 avenue des Gobelins,  
75013 Paris France.

### 3.3 Other members of the Qt-Caml team

Pierre-Yves Strub is a Ph.D. student at Polytechnique / INRIA Saclay-Île de France.

**email** : strub@lix.polytechnique.fr  
**address** : Laboratoire d'Informatique (LIX)  
École Polytechnique  
91128 Palaiseau Cedex France

Vincent Balat is an assistant professor at Université Paris-Diderot (Paris 7).

**email** : vincent.balat@pps.jussieu.fr  
**address** : Labo Preuves Programmes Systèmes  
Université Denis Diderot  
Case 7014  
2 Place Jussieu  
75251 PARIS Cedex 05

## 4 Conclusion

To conclude, we believe that this first milestone towards a robust binding to QT in OBJECTIVE CAML will contribute in two important directions to make development in OBJECTIVE CAML easier. First, the long-term goal of a binding to the QT library will help the development of robust applications and will improve the attractivity of OBJECTIVE CAML. Second, we are convinced that the general tools to make C++ and OBJECTIVE CAML live together can be crucial arguments to move from C++ to OBJECTIVE CAML.

## References

Brad King. Gcc-xml. <http://www.gccxml.org/HTML/Index.html>.

Trolltech. Qt. <http://trolltech.com/products/qt>.

Roman Yakovenko. pygccxml. <http://www.language-binding.net/pygccxml/pygccxml.html>.