MESURE Project

User's Guide

Emission Date	**November** 2007
Project name	MESURE
Document type	Technical report
Ref & Version	F3.5-1.0
Classification	Public
Number of pages	**33**

DOCUMENTATION LICENSE AGREEMENT

READ THE TERMS OF THIS AGREEMENT CAREFULLY BEFORE OPENING THE DOCUMENTATION MEDIA PACKAGE, OR DOWNLOADING THE DOCUMENTATION. BY OPENING THE DOCUMENTATION MEDIA PACKAGE, OR DOWNLOADING THE DOCUMENTATION YOU AGREE TO THE TERMS OF THIS AGREEMENT. IF YOU ARE ACCESSING THE DOCUMENTATION ELECTRONICALLY, INDICATE YOUR ACCEPTANCE OF THESE TERMS BY SELECTING THE "I AGREE" BUTTON AT THE END OF THIS AGREEMENT. IF YOU DO NOT AGREE TO ALL THESE TERMS, PROMPTLY RETURN THE UNUSED DOCUMENTATION OR, IF THE DOCUMENTATION IS ACCESSED ELECTRONICALLY, SELECT THE "I DECLINE" BUTTON AT THE END OF THIS AGREEMENT.

1. DEFINITIONS

Derivative Work: any work based upon the Documentation or upon the Documentation and other pre-existing works, such as translation, adaptation, modification, transformation, integration, etc.

Documentation: any report produced for the MESURE project.

Execution Date: date of receipt of the Documentation by Licensee.

License: this documentation license agreement.

Licensee: You

Licensors: MESURE partners (CNAM/Cedric, POPS CNRS/INRIA/USTL and Trusted Logic S.A.).

2. License

(i) Subject to the terms and conditions of this License, Licensors hereby grant Licensee, to the extent of Licensors' intellectual proprietary rights in the Documentation, a worldwide, royalty-free, non-exclusive license to exercise the rights in the Documentation as stated below:
   a. To consult the Documentation.
   b. To modify the Documentation to create Derivative Works.
   c. To copy and distribute the Documentation and Derivative Works created by Licensee.

(ii) The rights granted in Article 2(i) may be exercised in all media and formats whether now known or hereafter devised.

3. Restrictions

(i) The rights granted under Article 2(i) are for non commercial use only. Licensee may not exercise any of the rights granted under Article 2(i) in any manner that is primarily intended for or directed toward commercial advantage or private monetary compensation.

(ii) Licensee may distribute the Documentation and Derivative Works only under the terms of this License. Licensee must include a copy of this License, or a valid Uniform Resource Location for this License, with every copy of the Documentation or Derivative Work distributed by Licensee. Licensee may not offer or impose any terms on the Documentation or Derivative Works that alter or restrict the terms of this License or the recipient's exercise of the rights granted hereunder. Licensee must keep intact, inter alia, the disclaimer of warranties.

(iii) Licensee must notify Licensors of any Derivative Work created by Licensee and make such Derivative Work available to Licensors and any third party under the terms of this License.

4. INTELLECTUAL PROPERTY RIGHTS

(i) The Documentation and all associated intellectual property rights shall remain the exclusive property of Licensors. Licensors remains free to use, modify, integrate, distribute and license the Documentation and Derivative Work for any purpose, including commercial, and license the Documentation and Derivative Work to third parties under any license agreement. Licensors does not commit to provide any maintenance, error correction or support on the Documentation or to make future versions of the Documentation or Derivative Work created by Licensors available to Licensee or any third party.

(ii) Licensee may not remove or obscure any copyright notice or trademark notice of Licensors or third parties contained in the Documentation.

(iii) Licensee is granted no right or title in the Documentation or Derivative Work other than the rights set forth in Article 2 of this License.

5. Disclaimer of Warranty

The Documentation is provided to Licensee "as is". Except to the extent that these disclaimers are held to be legally invalid, Licensors make no representations or warranties of any kind concerning the Documentation, express, implied, statutory or otherwise, including, without limitation, warranties of accuracy, completeness, merchantability, fitness for a particular purpose, non-infringement, or the absence of latent or other defects, or the presence of errors, whether or not discoverable.

6. LIABILITY

(i) To the extent not prohibited by law, in no event will Licensors or their licensors be liable to Licensee for any lost revenue, profit or data, or for special, indirect, consequential, incidental or punitive or exemplary damages, however caused regardless of the theory of liability, arising out or related to the use of or inability to use the Documentation, even if Licensors have been advised of the possibility of such damages.

(ii) The limitations of damages and liability set forth in this License are fundamental elements of this License. The parties acknowledge and agree that neither would be able to perform hereunder on an economic basis without such limitations.

7. INFRINGEMENT

(i) In the event Licensee becomes aware of any claim or assertion that the Documentation might infringe a third party's intellectual property rights, Licensee shall immediately inform Licensors of such claim or assertion. Upon Licensors' request, Licensee shall then immediately cease using the Documentation and comply with Article 8(iii) of this License.

(ii) Should the Documentation be declared infringing in court, or in Licensors' reasonable opinion, be likely to become the subject of a valid claim of infringement of any intellectual property rights associated with the Documentation, Licensors shall, at their sole option, either (i) modify the Documentation so that it is no longer infringing, or (ii) notify Licensee in writing that the first option is not reasonably possible and terminate this License without any delay or compensation.

8. Term and Termination

(i) The term of this License shall begin on the Execution Date and expire with the expiration of the applicable copyright on the Documentation.

(ii) Licensors may terminate this License as set forth in Article 7 or upon any breach of this License by Licensee.

(iii) Upon termination of this License, Licensee shall cease any further use of the Documentation, return to Licensors all copies of the Documentation in any form in Licensee's possession or control, including storage media like e.g. floppy disks, CD-ROMS,

etc and permanently de-install or destroy the Documentation installed on any Licensee computer or storage device. Licensee shall certify in writing upon first request from Licensors that these measures have been carried out.

(iv) Articles 1, 4, 5, 6, 8, 9, 10, 11 and 12 shall survive termination or expiration of this License.

9. Severability

If any term or provision of this License should be declared invalid or unenforceable by a court of competent jurisdiction or by operation of law, the provision is to that extent to be deemed omitted, and the remaining provisions shall not be affected in any way. The invalid term or provision shall be replaced by such valid term or provision as comes closest to the intention underlying the invalid term or provision.

10.Waiver

The delay or failure of either party to exercise in any respect any rights or remedies provided for in this License shall not be deemed a waiver of that right or remedy nor shall any single or partial exercise of any right or remedy preclude the further exercise of that right or remedy or of any other right or remedy under this License.

11.Governing Law

The governing law is the law of France. All disputes arising out of or in connection with this License, including any question regarding its existence, validity or termination, shall be finally settled by arbitration under the Rules of arbitration of the "International Chamber of Commerce" by one arbitrator in accordance with the said Rules. Arbitration shall take place in Paris, France. The language to be used in the arbitration proceeding shall be French.

12.Notices

Any notice under this License shall be delivered either by letter to the address of the party concerned or by e-mail to its e-mail address. For inquiries, please contact:

- CNAM, 292 rue Saint Martin, 75141 Paris Cedex 3, France,
- or INRIA Futurs, POPS research group, Bât. M3, Cité Scientifique, 59655 Villeneuve d'Ascq Cedex, France,
- or Trusted Logic, 5 rue du Bailliage, 78000 Versailles, France.

Table of Contents

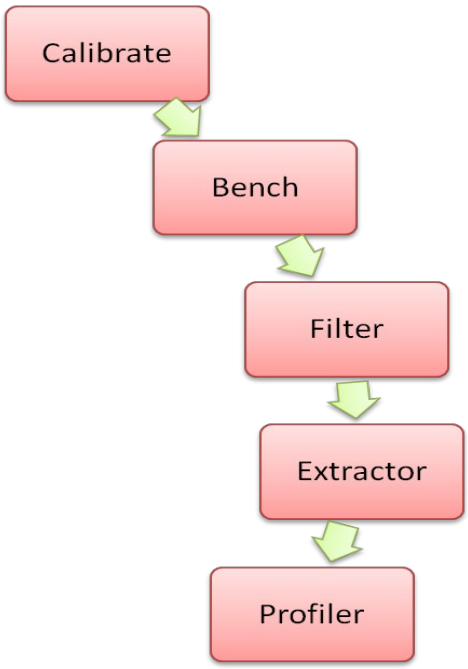
1 Introduction to the MESURE modules

This chapter introduces all the modules available in the MESURE project.

1.1 The Modules in Sequence

1.1.1 Description

The following diagram illustrates the sequence of execution for the available modules. More precisely, XML files are used as input and output files for modularity, and the output files produced by a given module serve as input files for the following modules.



The calibration module

The bench module

The filtering module

The extraction module

The profiling module

1.1.2 Configuration

This section details the structure of the XML configuration files used as input files by these different modules.

1.1.2.1 The Module Configuration Files

A "ToolConfig.xml" file is available in each "config/<module_name>" folder of the MESURE project and defines several sets of options for the different modules.

This file is read when specifying one of the following options for these modules:

```
    <module_name> -c <name>
    : use the specified set of options
OR <module_name> -config <name>
    : use the specified set of options
OR <module_name> -cd
    : use the default set of options
    (shortcut for manager -c default)
OR <module_name> -configDefault
    : use the default set of options
    (shortcut for <module_name> -config default)
```

As example, an excerpt of the configuration file used by the calibration module and the bench module is given below:

```
<Manager>
  <!-- Default configuration. Should always be defined. -->
  <config name="default" description="The default set of options.">
    <help/>
    <warning/>
    <log>log.txt</log>
    <verbose/>
    <verboseLevel>10</verboseLevel>
    <verboseFilter>no</verboseFilter>
    <input>config/ManagerConfig.xml</input>
    <xmloutput>results.xml</xmloutput>
    <textoutput>results.csv</textoutput>
  </config>

  <!-- My configuration -->
  <config name="my_config" description="My config">
    <config>default</config>
    <verboseLevel>2</verboseLevel>
  </config>
</Manager>
```

Thus, the following command line may be used to launch the bench module with the "my_config" set of options:

```
manager -c my_config
```

1.1.2.2 The Manager Configuration File

A "ManagerConfig.xml" file is available in the "config/manager" folder of the MESURE project. The associated XML Schema Definition (XSD) is the "src/tools/manager/config/ManagerConfig.xsd" file.

This file may be used as input file for the calibration module, the bench module (if a calibration is not required) and the profiling module (if a calibration is not required). The calibration module has for purpose to "transform" this file, and to fill it with the most appropriate configuration.

An excerpt of this file is given below:

```
<managerConfig>
  <cad>Gemplus USB Smart Card Reader 0</cad>
  <cardConfig>../cards/CardConfig.xml</cardConfig>
  <tests>
    <test
    testConfig="benchs/jcapi/javacard_framework/ownerpin/TestConfig.xml">
      <testCase name="check_ref" x="11" y="5" minLoop="2"
        maxLoop="181" startLoop="9" precision="2.0">
        <calibrateMetric Xmin="11" Ymin="5" tmin="8.891092262E8"
          a="8.891092262E8" b="0.0"/>
      </testCase>
      ...
    </test>
    ...
  </tests>
</managerConfig>
```

Where:

- `<cad>` specifies the name of the PC/SC CAD to be used. The list of the CADs installed on your computer, and their exact names, are given in the Windows registry, under "HKEY_LOCAL_MACHINE/SOFTWARE/Microsoft/Cryptography/Calais/Readers".

By default, the JSR 268 library is used to communicate with the card (through the `lib.cad.JSR268CAD` class), but it is possible to specify any other library. In order to do that, you first need to develop a class extending the `lib.cad.CAD` abstract class and implementing the expected methods; this class will serve as gateway between the MESURE project and the communication library. Then, you need to reference this class in the configuration file using the format `<package_name><class_name>:<options>`. For example, the following values are valid:

```
<cad>lib.cad.JSR268CAD:Gemplus USB Smart Card Reader 0</cad>
 OR
<cad>lib.cad.MyCAD:</cad>, if no option are necessary
```

- `<timeProvider>` specifies the name of the chronometer to be used. By default, the `System.nanoTime` method is used to measure the execution times (through the `lib.chrono.SystemTimeNanosChronometer` class), but it is possible to specify any other chronometer. In order to do that, you first need to develop a class implementing the `lib.chrono.Chronometer` interface. Then, you need to reference this class in the configuration file using the format `<package_name><class_name>`. For example, the following values are valid:

```
<timeProvider>lib.chrono.SystemTimeNanosChronometer</timeProvider>
to use the System.nanoTime method (by default, if no chronometer is specified)
 OR
<timeProvider>lib.chrono.SystemTimeMillisChronometer</timeProvider>
to use the System.currentTimeMillis method instead
 OR
<timeProvider>lib.chrono.MyChronometer</timeProvider>
```

Note that this tag is useless if you want to use the default chronometer, or if the `CAD` class used to communicate with the card (and whose name is specified with the `<cad>` tag) implements the `Chronometer` interface (*e.g.* a class implemented to communicate with a precision reader).

- `<cardConfig>` specifies the path of the card configuration file.
- `<test>` specifies the path of a test configuration file to be executed.
- `<testcase>` specifies the test cases to be executed for this test, and the parameters to be applied. Thus, `x` fixes the number of on-card executions (*i.e.* when processing the APDU command associated to this test case, the test applet invokes `x` times the piece of code to be measured; see loop size L in § 1.3.1), and `y` fixes the number of off-card executions (*i.e.* the APDU command corresponding to this test case is executed `y` times). Other parameters are useful for calibration: `minLoop`, `maxLoop`, and `statLoop` fix respectively a minimal value, a maximal value, and a starting value for the `x` parameter when searching for the best value, and `precision` sets the maximal expected ratio between the mean and the standard deviation, which can be used during a calibration. The exact value of the precision index is

$$precision \leqslant \log\left(\frac{mean}{standard\ \ deviation}\right)$$

– `<calibratedMetric>` specifies some of the parameters encountered after a calibration of the test. Those should not interfere during a normal use of the manager tool, but we choose to leave those for a potential future use of the application. `Xmin` is an optional parameter which is the minimal on-card loop size for the benchmark to be valid. This should be, in general, similar to the parameter `x` of the `<testcase>`. `Ymin` is also an optional parameter which defines the minimal valid off-card loop size for the test. This should be, in general, equal to the `y` parameter of the `<testcase>`. `tmin` defines the minimal valid time measurement for the test. Assuming that we are dealing with a case where the time `t` grows linearly with the on-card loop size `l`, we should have an equation of the form :

$$t = (l \times a) + b$$

then `a` and `b` are parameters deduced during the calibration through a linear analysis. Those might eventually be useful to extrapolate the behavior of an applet.

1.1.2.3 The Card Configuration File

A "CardConfig.xml" file is available in the "config/cards" folder of the MESURE project. The associated XML Schema Definition (XSD) is the "src/lib/loader/config/CardConfig.xsd" file.

This file is referenced by The Manager Configuration File and contains useful information about the tested card. An excerpt of this file is given below:

```
<cardConfig>
  <cardManagerAid>0xA0:00:00:00:03:00:00:00</cardManagerAid>

  <mutualAuthenticate>
    <div>none</div>
    <scp>1</scp>
    <secLevel>1</secLevel>
    <kenc>0x40:0x41:0x42:0x43:0x44:0x45:0x46:0x47:0x48:0x49:0x4A:0x4B:0x4
C:0x4D:0x4E:0x4F</kenc>
    <kmac>0x40:0x41:0x42:0x43:0x44:0x45:0x46:0x47:0x48:0x49:0x4A:0x4B:0x4
C:0x4D:0x4E:0x4F</kmac>
    <keyVersion>0</keyVersion>
    <keyIndex>0</keyIndex>
  </mutualAuthenticate>

  <load>
    <lc>127</lc>
    <nonVolatileCode>false</nonVolatileCode>
  </load>
</cardConfig>
```

Where:

– `<cardManagerAid>` specifies the AID of the Card Manager.

– `<div>` specifies if the provided key shall be diversified, or not. More precisely, "none" indicates that no diversification is required (the `<kenc>` and `<kmac>` tags give the diversifed keys to be used for mutual authentication), "visa1" indicates that the VISA 1 diversification method will be applied on the specified the master key (the <kmc> tag gives the master key to be diversified; <kenc> and <kmac> are useless), and "visa2" indicates that the VISA 2 diversification method will be applied on the specified master key.

The VISA 1 and VISA 2 diversification methods both use 3DES_ECB with a 16-byte master key (KMC) and 16 bytes of diversification data (D). The data diversification method is as follows:

D = Dleft || Dright

KD = KDleft || KDright

KDleft = 3DES(Dleft,KMC)

KDright = 3DES(Dright,KMC)

The diversification data for VISA 1 are given below:

Key	**Diversification Data (D)**
KENC	FF FF || card serial number (8 bytes from the CPLC data) || 01 00 00 00 00 00
KMAC	00 00 || card serial number || 02 00 00 00 00 00

The diversification data for VISA 2 are given below:

Key	**Diversification Data (D)** xx xx are the last (rightmost) two bytes of the Card Manager AID.
KENC	xx xx || IC serial number (4 bytes from the CPLC data) || F0 01 || xx xx || IC serial number
KMAC	xx xx || IC serial number || F0 02 || xx xx || IC serial number

- `<scp>` specifies the Secure Channel Protocol (SCP). "1" and "2" are the only valid values.
- `<secLevel>` specifies the security level. "0" indicates that no security is required for the commands, "1" indicates that integrity shall be ensured (a MAC is added to the commands), and "2" indicates that the integrity and the confidentiality shall be ensured (a MAC is added to the commands and the commands are enciphered).
- `<kenc>` specifies the 16-byte key to be used for authentication and ciphering.
- `<kmac>` specifies the 16-byte key to be used for MAC computation.
- `<keyVersion>` specifies the version of the key to be used for mutual authentication.
- `<keyIndex>` specifies the index of the key to be used for mutual authentication.
- `<lc>` specifies the maximum length for the LOAD commands.
- `<nonVolatileCode>` specifies if the non volatile code space limit shall be set or not for the INSTALL [for LOAD] commands.

1.1.2.4 The Test Configuration File

A "TestConfig.xml" file is available in each test package of the "src/bench" folder of the MESURE project. The associated XML Schema Definition (XSD) is the "src/bench/lib/config/TestConfig.xsd" file.

These files are referenced by The Manager Configuration File and contain useful information about the tests to be performed. An excerpt of this file is given below:

```
<testConfig>
  <package>
    <capfile>benchs/jcapi/javacard_framework/ownerpin/javacard/ownerpin.c
ap</capfile>
  </package>
  <script>scripts.jcapi.javacard_framework.ownerpin.OwnerPINScript</scrip
t>
</testConfig>
```

- `<capfile>` specifies the path of the CAP file associated to this test and that shall be loaded on the card.

– `<script>` specifies the path of the script associated to this test and that shall be executed on the computer (it defines the commands that shall be sent to the card).

1.1.2.5 The Profiler Configuration File

A "coefs.xml" file is available in the "config/profiler" folder of the MESURE project. The associated XML Schema Definition (XSD) is the "src/lib/xml/profiler_result/coefs.xsd" file.

This file is used as input file for the profiling module and contains some coefficients used when calculating the mark for the tested card. An excerpt of this file is given below:

```
<Domains>
  <Domain name="Transport">
    <Bytecodes totalocc="34179">
      <Bytecode name="ACONST_NULL" nbocc="1" avgref="10244.13779703775"/>
      ...
    </Bytecodes>
    <Methods totalocc="52">
      <Method name="javacard.framework.AID.partialEquals" nbocc="1"/>
      ...
    </Methods>
  </Domain>

  <Domain name="Identity">
  ...
  </Domain>

  <Domain name="Banking">
  ...
  </Domain>
</Domains>
```

Thus, for each domain (the domain of the transport applications, identity applications or banking applications), and for each main bytecode and API method (see report F2.1 Functionalities):

– `nbocc` specifies the average number of occurrences of this bytecode or API method in the code of the applications in this domain.

– `avg` describes the mean execution time for the given tested card.

– `avgref` specifies the mean value of the performance used as reference for the given test.

– `impactcoef` this value indicates for each application domain the proportion of the bytecode considering all the bytecodes for this domain. This is based on the `nbocc` number of each of the bytecodes within the domain.

$$impactcoef_j = \frac{nbocc_j}{\sum_{i=1}^{n} nbocc_i}$$

If we suppose that $j \in [1..n]$ for n the number of test performed.

1.1.2.6 Java Binding

Note that the Castor library has been used to generate automatically Java files from XSD files.

The entry point in this library is the `org.exolab.castor.builder.SourceGenerator` class, and you need to invoke its `main` method with the following arguments:

```
-f -dest src -i <XSD_file> -types j2 -package <package_name>
```

For example, to generate the Java files from the `coefs.xsd` file, you will have to specify the following arguments:

```
-f -dest src -i src/lib/xml/profiler_result/coefs.xsd -types j2 -package
lib.xml.profiler_result
```

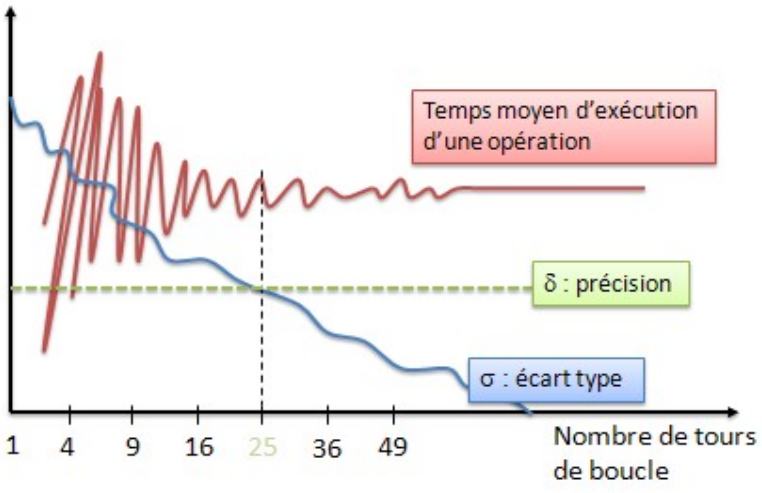
Here is an exhaustive list of the XSD files defined in the MESURE project:

- src/tools/manager/config/ManagerConfig.xsd. This file is the schema for The Manager Configuration File passed as input file to the calibration module (and to the bench module and the profiling module if a calibration is not required), and for the output file produced by the calibration module.
- src/lib/loader/config/CardConfig.xsd. This file is the schema for the .
- src/bench/lib/config/TestConfig.xsd. This file is the schema for The Test Configuration File.
- src/lib/xml/profiler_result/coefs.xsd. This file is the schema for the The Profiler Configuration File passed as input file to the profiling module.
- src/lib/xml/extractor_result/ExtractorResult.xsd. This file is the schema for the output file produced by the extraction module, and passed as input file to the profiling module.
- src/lib/xml/profiler_result/fusion.xsd. This file is the schema for the database produced by the profiling module.
- src/lib/xml/test_result/TestResult.xsd. This file is the schema for the output file produced by the bench module and by the filtering module, and passed as input file to the profiling module.

1.2 The Calibration Module

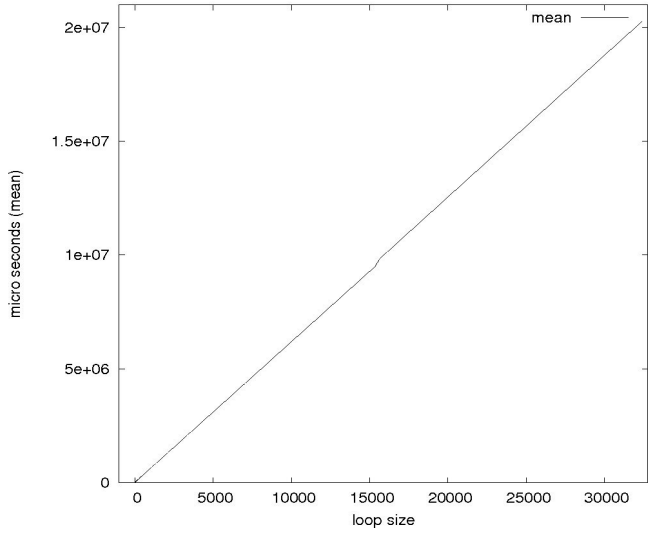
1.2.1 Description

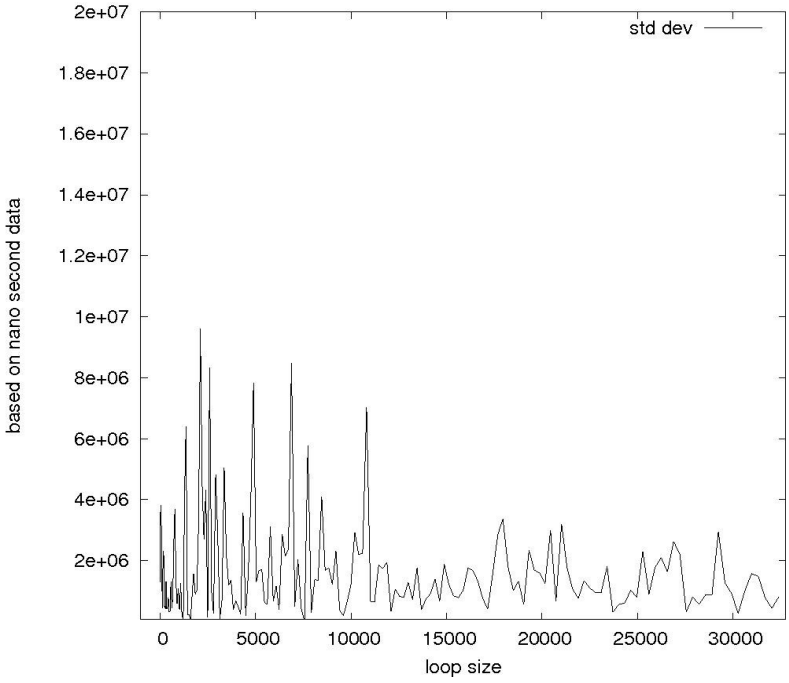
The calibration module computes the optimal parameters (such as the number of loops) needed to obtain measurements of a given precision.



For example, it has been observed that the measurements tend to a certain degree of stability as the loop size increases, and the minimal loop size required to reach this stability depends on several factors, such as the Card Acceptance Device (CAD), its driver, the computer OS, the computer CPU, or the card itself, etc.

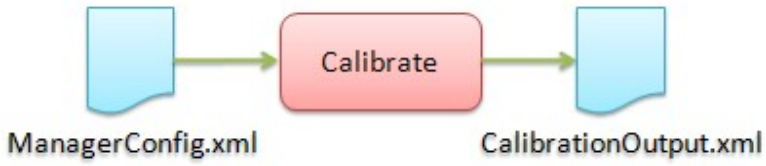
Here are some real-life examples of the evolution of the mean as the on-card loop size grows and the evolution of the standard deviation as the loop size grows for a given test. The linear curve shows the variation of the mean given 30 measurements for each available on-card loop size. The second curve shows the repective standard deviation for the same measurements.





Benchmarking the various bytecodes and API entries takes time. However, it is necessary to be precise enough when measuring those execution times. Furthermore, the end user of such a benchmark should be allowed to focus on a few key elements with a higher degree of precision. It is therefore necessary to devise a tool that let us decide what are the most appropriate parameters for the measurements.

1.2.2 Usage




```
Usage: manager <options> -cal install_path
    OR manager <options> -setCalibrate install_path
    OR manager -c <name>
       : use the specified set of options
    OR manager -config <name>
       : use the specified set of options
    OR manager -cd
       : use the default set of options
       (shortcut for manager -c default)
    OR manager -configdefault
       : use the default set of options
       (shortcut for manager -config default)
Where options include:
 -dc or -displayConfig:display the list of available names for the
                        -c and -config options
 -h  or -help         :print out this message.
 -i  or -input        :specify the manager configuration file.
                       If not specified, the "ManagerConfig.xml" file in
                       "config/manager" folder is used as input file.
 -l  or -log          :specify the log file.
                       If not specified, ths standard output is used to
                       display the calibration history.
 -to or -csvOutput    :specify the text file containing the results.
                       If not specified, the "tmp\results.csv" file is
                       created to store the calibration results.
 -v  or -verbose      :enable verbose history.
 -vf or -verboseFilter:[filtername:filterlevel]
                       set a filter for the history.
                       filterlevel must be an int, or "no" (no filter),
                       or "all" (filter all).
 -vl or -verboseLevel :[filterlevel]
                       set a filter level for the history.
                       filterlevel must be an int, or "no" (no filter),
                       or "all" (filter all).
 -w  or -warning      :[filterlevel]
                       shortcut for -vf [WARNING:filterlevel]
 -xo or -xmlOutput    :specify the XML file containing the results.
                       If not specified, a "tmp\results.xml" file is
                       created to store the calibration results.

install_path          :specify the installation path.
                       If not specified, this path is set to ".."
```

1.2.3 Implementation

The entry point for the calibration module is the `tools.manager.Main` class. In order to run this class from your Eclipse workspace you have to set the working directory to be the "bin" folder of the MESURE project, and to specify the program arguments as detailed before (*e.g.* `-cal -xo ../config/manager/CalibrationOutput.xml`).

1.3 The Bench Module

1.3.1 Description

For a number of cycles, defined by the calibration module, the bench module performs the measurements for:

– The Virtual Machine (VM) bytecodes.

– The Application Programming Interface (API) methods.

– The Runtime Environment (RE) mechanisms, such as the transactions, etc.

– The memory usage.

Each test consists of two parts: the script part and the applet part. The script is written in Java language: each test script extends the `scripts.templates.TemplateScript` class, and each test case of such a script extends the `scripts.templates.TestCase`. The applet is written in Java Card language: each test applet extends the `bench.lib.templates.TemplateApplet` class; each test case extends the `bench.lib.templates.TestCase` class, and implements the following methods:

- A `setUp` method responsible to perform any operation (*e.g.* allocations) needed before running the test case.
- A `run` method used to run the test case.
- A `cleanUp` method invoked after running the test case to perform any useful cleanup.

The purpose of the bench module is to determine as accurately as possible the execution time of the code contained in the `run` method.

There exists a significant and non-predictable elapse of time between the beginning of the measurement, characterized by the starting of the chronometer, and the execution of this method. Indeed, when launching a test case, the execution goes through several software and hardware layers, down to the card's hardware and up to the card's VM. This non-predictability is mainly dependent on hardware characteristics of the benchmark environment (such as the CAD, the computer's hardware, etc), the OS level interferences, services and also on the computer's VM. To minimize the effects of these interferences, we need to isolate the execution time of the code, while ensuring that this execution time is long enough to be measured.

It requires on the one hand a test applet structure with a loop, which will execute the code for a substantial amount of time. On the other hand, it requires to compute the execution time of the code upon which the code of interest is dependent. For example, if the purpose of a test case is to measure the execution time for the `sadd` bytecode, then a prerequisite is to measure the execution time for the bytecodes in charge of loading its operands onto the stack (two `sspush`).

The illustration below gives an example of test case (on the left) and the associated test case whose execution time has to be subtracted ((on the right, called the reference test case).

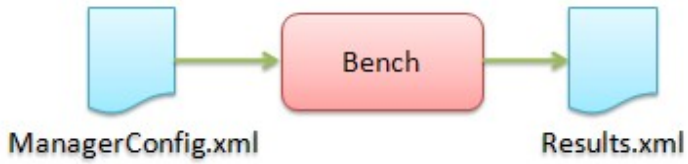
```
process()
{
   i=0
   while i<=L DO
   {
      run()
      i = i+1
   }
}

run()
{
   (n + 1){sspush num}
   (n){op}
}
```

```
process()
{
   i=0
   while i<=L DO
   {
      run()
      i = i+1
   }
}

run()
{
    (n + 1){sspush num}
}
```

1.3.2 Usage




```
Usage: manager <options> install_path
    OR manager -c <name>
       : use the specified set of options
    OR manager -config <name>
       : use the specified set of options
    OR manager -cd
       : use the default set of options
       (shortcut for manager -c default)
    OR manager -configDefault
       : use the default set of options
       (shortcut for manager -config default)
Where options include:
 -dc or -displayConfig:display the list of available names for the
                        -c and -config options
 -h  or -help         :print out this message.
 -i  or -input        :specify the manager configuration file.
                       If not specified, the "ManagerConfig.xml" file in
                       "config/manager" folder is used as input file.
 -l  or -log          :specify the log file.
                       If not specified, the standard output is used to
                       display the execution history.
 -to or -csvOutput    :specify the text file containing the results.
                       If not specified, the "tmp\results.csv" file is
                       created to store the execution results.
 -v  or -verbose      :enable verbose history.
 -vf or -verboseFilter:[filtername:filterlevel]
                       set a filter for the history.
                       filterlevel must be an int, or "no" (no filter),
                       or "all" (filter all).
 -vl or -verboseLevel :[filterlevel]
                       set a filter level for the history.
                       filterlevel must be an int, or "no" (no filter),
                       or "all" (filter all).
 -w  or -warning      :[filterlevel]
                       shortcut for -vf [WARNING:filterlevel]
```

```
-xo or -xmlOutput     :specify the XML file containing the results.
                        If not specified, a "tmp\results.xml" file is
                        created to store the execution results.

install_path           :specify the installation path.
                        If not specified, this path is set to ".."
```

1.3.3 Implementation

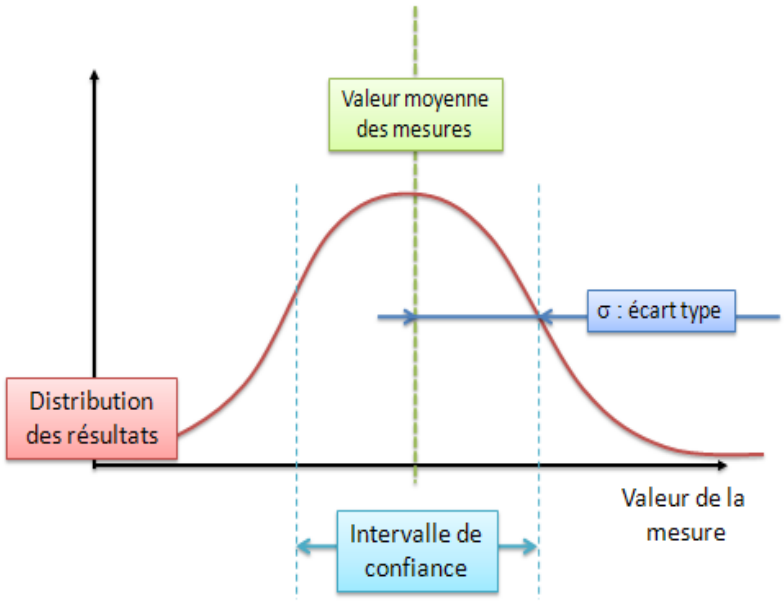
The entry point for the bench module is the `tools.manager.Main` class. In order to run this class from your Eclipse workspace you have to set the working directory to be the "bin" folder of the MESURE project, and to specify the program arguments as detailed before (*e.g.* `-vl 10 -input ../config/manager/CalibrationOutput.xml`).

1.4 The Filtering Module

1.4.1 Description

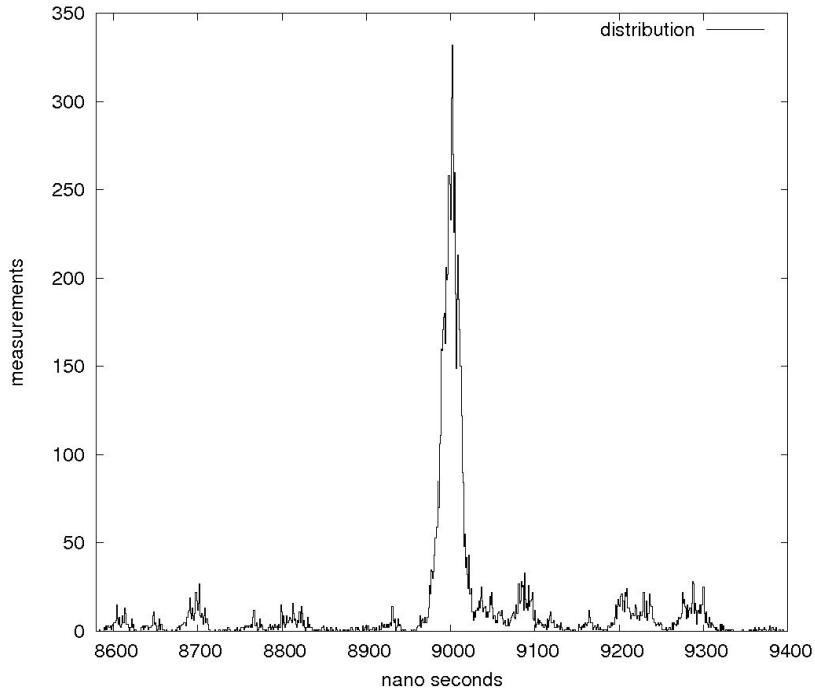
Experimental errors perturb the raw measurements. These perturbations lead to imprecision in the measured values, making it difficult to interpret the results. In the context of the MESURE project, the perturbations come from the card, and/or the CAD in which this platform is inserted, and/or the computer in which this CAD is plugged (measurement tools, OS, hardware).

As a consequence, each test case is executed several times.



The default behavior of the filtering module is to get the set of measurements for a given test case, to remove a given percentage (default values are 50% for the reference test cases, 20% for the standard test cases) of these measurements in order to improve drastically their stability when a "noisy" system is used, and then to output the filtered set of measurements.

Here is an example of a real-life distribution curve for a given bytecode on an actual platform for a total of 10000 measurements. The mean value for those measurements is 9013ns.



1.4.2 Usage


```
Usage: filter <options> -f install_path
    OR filter <options> -filter install_path
    OR filter -c <name>
       : use the specified set of options
    OR filter -config <name>
       : use the specified set of options
    OR filter -cd
       : use the default set of options
       (shortcut for manager -c default)
    OR filter -configDefault
       : use the default set of options
       (shortcut for manager -config default)
Where options include:
 -dc   or -displayConfig  :display the list of available names for the
                           -c and -config options
 -h    or -help           :print out this message.
 -i    or -input          :specify the XML input file
                           (i.e. the XML output file produced by the bench
                           module).
                           If not specified, the "tmp/results.xml" file is
                           used as input file.
 -l    or -log            :specify the log file.
                           If not specified, the standard output is used
                           to display the filetring history.
 -o    or -output         :specify the XML output file
                           If not specified, a "filetred.xml" file is
                           created in the "tmp" folder.
```

```
 -rcp or -refCasePercent :specify the percentage of reference test cases
                          to be filtered.
 -tcp or -testCasePercent:specify the percentage of standard test cases
                          to be filtered.
 -v   or -verbose        :enable verbose history.
 -vf  or -verboseFilter  :[filtername:filterlevel]
                          set a filter for the history.
                          filterlevel must be an int, or "no" (no
                          filter), or "all" (filter all).
 -vl  or -verboseLevel   :[filterlevel]
                          set a filter level for the history.
                          filterlevel must be an int, or "no" (no
                          filter), or "all" (filter all).
 -w   or -warning        :[filterlevel]
                          shortcut for -vf [WARNING:filterlevel]

install_path             :specify the installation path.
                          If not specified, this path is set to ".."
```

1.4.3 Implementation

The entry point for the filtering module is the `tools.filter.Main` class. In order to run this class from your Eclipse workspace you have to set the working directory to be the "bin" folder of the MESURE project, and to specify the program arguments as detailed before.

1.5 The Extraction Module

1.5.1 Description

The extraction module is used to isolate the execution time of the features of interest among the mass of raw measurements that we gathered so far.

More precisely, this module performs the following calculation:

$$\bar{M}(op) = \frac{\bar{m}_L(op) - \bar{m}_L(emptyloop)}{L} - \sum_{i=1}^{n} \bar{M}(op_i)$$

where:

- M(op) is is the execution time for a given part of code in a test case (a bytecode or an API method).

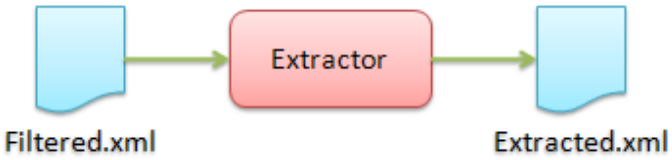
- m_L(op) is the global (*i.e.* for the loop size L) execution time for this same part of code, but including interferences during the measurement, coming from the card, and/or the CAD, and/or the computer.

- m_L(emptyloop) is the global execution time if the code of the test case is empty (*i.e.* the `run` method does nothing).

Other operations represent for example auxiliary parts of code needed to execute the code of interest.

If a test case is executed several times, the execution time taken into consideration in the formula is the mean computed over a significant number of executions.

1.5.2 Usage




```
Usage: extractor <options> -r install_path
       extractor <options> -resolve install_path
    OR extractor -c <name>
       : use the specified set of options
    OR extractor -config <name>
       : use the specified set of options
    OR extractor -cd
       : use the default set of options
       (shortcut for manager -c default)
    OR extractor -configDefault
       : use the default set of options
       (shortcut for manager -config default)
Where options include:
 -dc   or -displayConfig  :display the list of available names for the
                           -c and -config options
 -h    or -help           :print out this message.
 -i    or -input          :specify the XML input file
                           (i.e. the XML output file produced by the
                           extraction module).
                           If not specified, the "tmp/filtered.xml" file
                           is used as input file.
 -l    or -log            :specify the log file.
                           If not specified, the standard output is used
                           to display the extraction history.
 -o    or -output         :specify the XML output file
                           If not specified, a "extracted.xml" file is
                           created in the "tmp" folder.
 -p    or -print          :print the knowledge base used for the
                           extraction.
 -v    or -verbose        :enable verbose history.
 -vf   or -verboseFilter  :[filtername:filterlevel]
                           set a filter for the history.
                           filterlevel must be an int, or "no" (no
                           filter), or "all" (filter all).
 -vl   or -verboseLevel   :[filterlevel]
                           set a filter level for the history.
                           filterlevel must be an int, or "no" (no
                           filter), or "all" (filter all).
 -w    or -warning        :[filterlevel]
                           shortcut for -vf [WARNING:filterlevel]

install_path              :specify the installation path.
                           If not specified, this path is set to ".."
```

1.5.3 Implementation

The entry point for the filtering module is the `tools.extractor.Main` class. In order to run this class from your Eclipse workspace you have to set the working directory to be the "bin" folder of the MESURE project, and to specify the program arguments as detailed before.

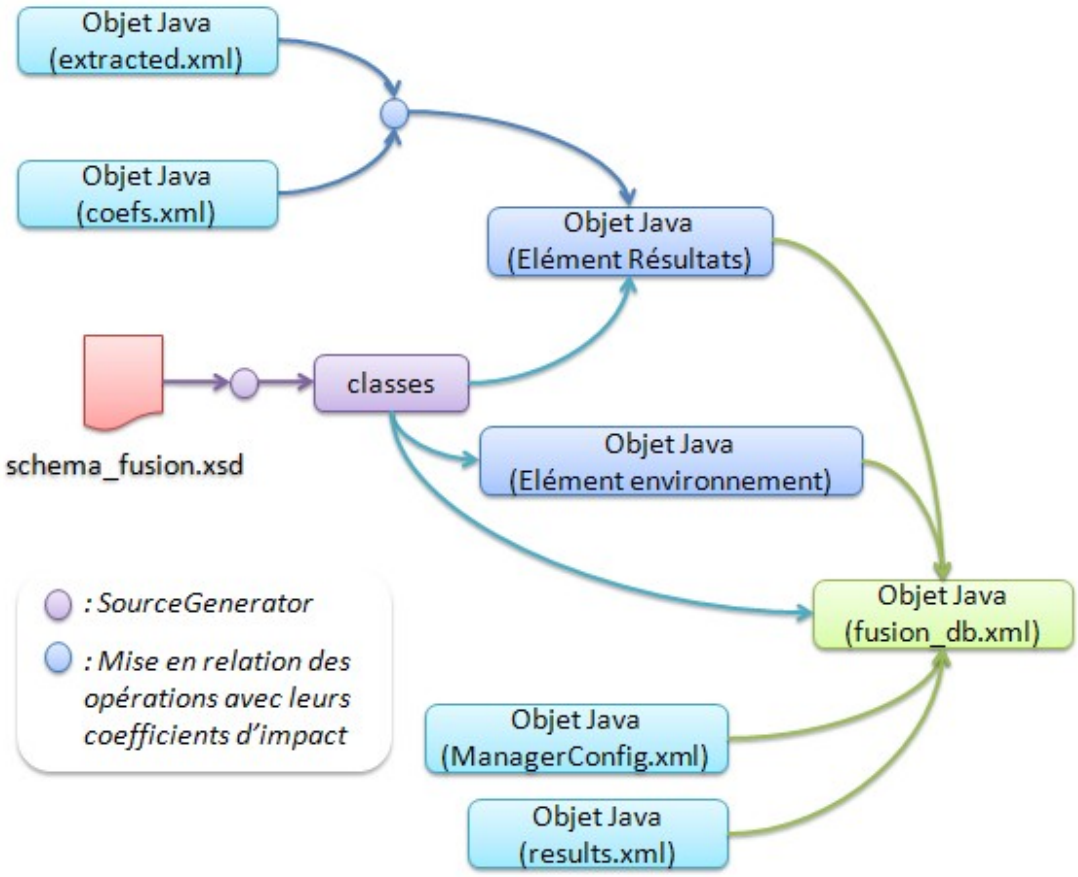
1.6 The Profiling Module

1.6.1 Description

The profiling module is used to produce a mark that represents the performances of the tested card. This mark may differ, according to the considered application domain (banking, transport, identity).

The way we give a mark to the performance is described in detail in **[F2.2MET]**

1.6.2 Usage

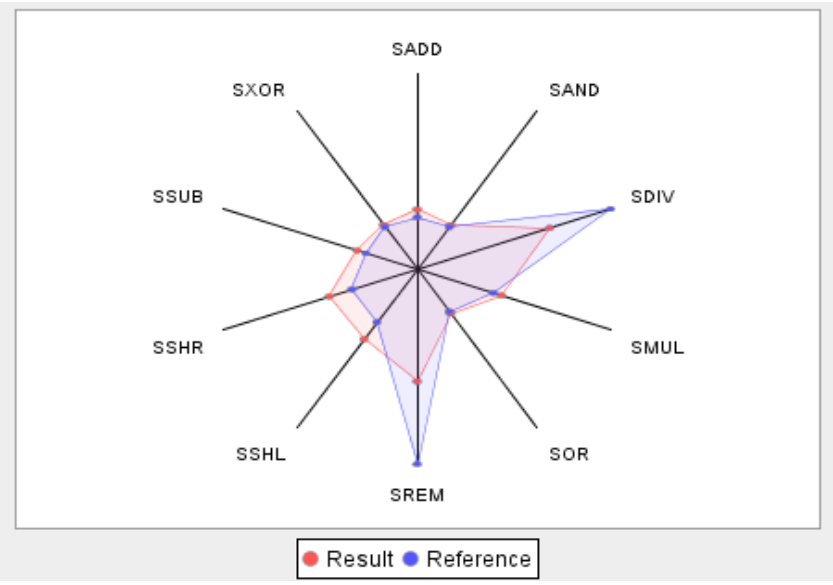



```
Usage: profiler <options> install_path
    OR profiler -c <name>
       : use the specified set of options
    OR profiler -config <name>
       : use the specified set of options
    OR profiler -cd
       : use the default set of options
       (shortcut for manager -c default)
    OR profiler -configDefault
       : use the default set of options
       (shortcut for manager -config default)
Where options include:
 -db  or -database       :specify the database.
                          If not specified, the "tmp/fusion_db.xml" file
                          is created from the input files.
 -dc  or -displayConfig  :display the list of available names for the
                          -c and -config options
```

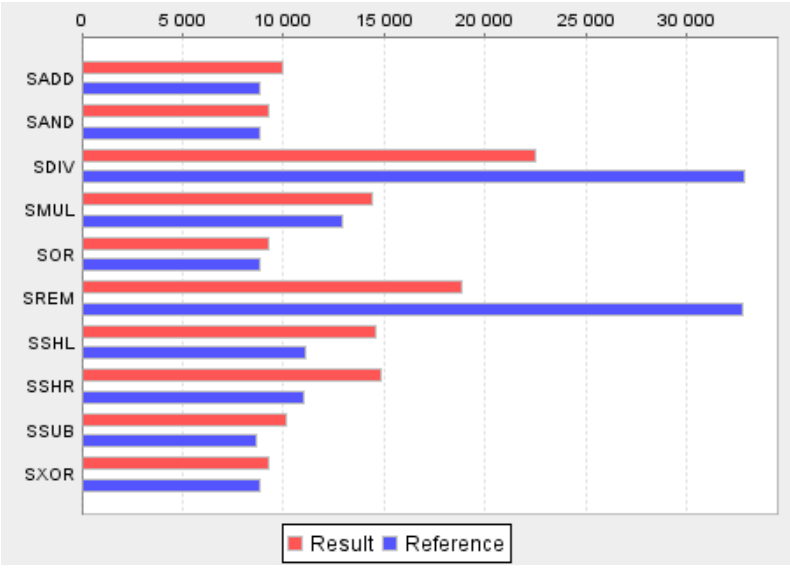
```
-dcs or -domainStyle    :specify the kind of chart to be used to
                         differentiate the application domains.
                         If not specified, a "Bar" chart is built.
-h   or -help           :print out this message.
-i   or -inputs         :[raw_results][extracted_results]
                         [manager_config][coefficients]
                         specify the XML input files.
                         If 0 is specified, the default file is used.
-lg   or -lang          :specify the language for the created HTML file.
                         If not specified, the HTML file is written in
                         english.
-l   or -log            :specify the log file.
                         If not specified, the standard output is used
                         to display the profiling history.
-mcs or -marksStyle     :specify the kind of chart to be used to
                         differentiate the marks.
                         If not specified, a "Spider" chart is built.
-o   or -output         :specify the output file.
                         If not specified, an HTML file is created in
                         the "tmp" folder.
-ss  or -stylesheet     :specify the stylesheet for the created HTML
                         file.
                         If not specified, the " profiler.xsl" file of
                         the "config/profiler" folder is used.
-v   or -verbose        :enable verbose history.
-vf  or -verboseFilter  :[filtername:filterlevel]
                         set a filter for the history.
                         filterlevel must be an int, or "no" (no
                         filter), or "all" (filter all).
-vl  or -verboseLevel   :[filterlevel]
                         set a filter level for the history.
                         filterlevel must be an int, or "no" (no
                         filter), or "all" (filter all).
-w   or -warning        :[filterlevel]
                         shortcut for -vf [WARNING:filterlevel]

install_path            :specify the installation path.
                         If not specified, this path is set to ".."
```

An example of spider chart that the produced HTML file may contain:



The same results illustrated with a bar chart:



1.6.3 Implementation

The entry point for the profiling module is the `tools.profiler.Main` class. In order to run this class from your Eclipse workspace you have to set the working directory to be the "bin" folder of the MESURE project, and to specify the program arguments as detailed before (*e.g.* `-vl 10 -i ../tmp/results.xml ../tmp/extracted.xml ../config/manager/CalibrationOutput.xml ../config/profiler/coefs.xml`).

2 Getting started

This section explains how to get and work with the MESURE project.

2.1 Getting a working environment

2.1.1 Quick Requirements

The following softwares are required:

- The Java Development Kit (JDK), version 6 update 1 or later (useful to get the JSR 268 smart card I/O API).
- The JavaCard Development Kit (JCDK), version 2.1.2 (useful to get the export files for the Java Card API v2.1) AND version 2.2.2 (useful to get a reliable converter and some Ant tasks).
- The Eclipse Software Development Kit (SDK), version 3.1 or later.
- The subclipse plug-in for Eclipse (useful for SVN).

2.1.2 Setting up the JCDK

In order to build the CAP files for the test applets of the MESURE project, you need to download the JCDK v2.1.2 AND to download and install the JCDK v2.2.2. These development kits are provided at the following URL:

http://java.sun.com/products/javacard/dev_kit.html

In order to convert easily the test applets, the content of the "api21_export_files" folder of the JCDK v2.1.2 is to be put in the "tools/converter" folder of the MESURE project. In parallel, the following archives, found in the "lib" folder of the JCDK v2.2.2, have to be put in the same folder: "converter.jar", "offcardverifier.jar" and "jctasks.jar". Then, it is possible to convert a test applet by running the Ant "build.xml" file defined in the applet package, or to convert a set of test applets by running the Ant "build.xml" files contained in the root folders.

2.1.3 Setting up the JDK

The MESURE project currently needs the JDK 6 update 1 up and running. You can find and download it from the following URL:

Http://java.sun.com/javase/downloads/index.jsp

2.1.4 Setting up the Eclipse SDK

The MESURE projects has been developed using the Eclipse environment, which can be dowloaded from the following URL:

http://www.eclipse.org/downloads/

2.1.4.1 Setting up the Subclipse Plug-In

In order to checkout the sources from the MESURE repository, we invite you to use the "subclipse" plug-in, which is a subversion eclipse client that can be found at the following URL:

http://subclipse.tigris.org/install.html

2.1.4.2 Checking out from the MESURE repository

In order to get a working copy of MESURE into your eclipse workspace, please follow the following steps:

- Import from SVN
  - File->Import->"Checkout Projects from SVN"
- Check "Create a new repository location" and input the location:
  - https://scm.gforge.inria.fr/svn/mesureprv
    - for contributors only
  - https://scm.gforge.inria.fr/svn/mesure
    - for anyone
- Select the HEAD folder
- Checkout as a project configured using the New Project Wizard
- Select "Java Project", input a name
  - Here your Default JRE should be the JDK 1.6.0; if not, you should probably select/install it.
- Finish

The project is now check outed. You probably get some auto-compilation errors. Let's fix them:

- We have to add the Java Card library to the project in order to compile the Java Card source files:
  - Project->Properties->Java build Path->Librairies->Add External JARs
  - Add the "api21.jar" file from your JCDK v2.1.2 "lib" installation folder
- Then, we have to add third-party libraries to the project in order to compile the Java source files:
  - Project->Properties->Java build Path->Librairies->Add JARs
  - Add all the JAR files provided in the "lib" folder of the MESURE project. More precisely, this folder contains the following libraries:

    - jfreechart-1.0.5.jar: JFreeChart is a Java library that makes easy to display charts. See http://www.jfree.org/jfreechart/index.html.

    - jcommon-1.0.9.jar: JFreeChart requires the JCommon library.

    - castor-1.0.3.jar: Castor is a Java library that provides Java-to-XML binding. See http://www.castor.org.

    - commons-logging-1.1.jar: Castor requires the Jakarta Commons Loggin library.

    - xerces-J_1.4.0.jar: Castor requires the Xerces XML library.

The project should now be built and error-free. You may skip the command line checkout and go to the tutorial.

2.1.4.3 Checking out using the command line (for anyone)

The MESURE project is currently hosted on the INRIA gforge server, which provides the subversion source-control tool used to incrementally upgrade the sources during the development phase.

For the next step you need a Subversion client. Subversion can be found at the following URL:

http://subversion.tigris.org/

The repository is located at:

https://scm.gforge.inria.fr/svn/mesure

No authentification is needed to get a working copy, and the following command line should succeed:

```
$ svn co https://scm.gforge.inria.fr/svn/mesure
```

2.1.4.4 Checking out using the command line (for contributors)

The repository is here:

https://scm.gforge.inria.fr/svn/mesureprv

Checkouting sources is possible using the following command line:

```
$ svn co https://scm.gforge.inria.fr/svn/mesureprv
```

A valid login and password are requested.

Note that the INRIA gforge allows 2 ways of authentification in order to checkout the source files: the first (easiet) is using webdav (just illustrated). The other one requires to set an openRSA key. Please go to the http://gforge.inria.fr website for further information.

2.2 A Quick Tutorial

The purpose of this section is to help you to run the different modules of the MESURE project for the first time. For each module, the expected command line and the expected results are illustrated.

2.2.1 Configuring

```
config/cards/CardConfig.xml
```

Uncomment the configuration corresponding to the smart card you are about to test, or edit one specific to your needs. Typically, the entry should look like the one presented in The Card Configuration File.

```
config/manager/ManagerConfig.xml
```

Edit the configuration file according to what you want to test. Typically, the entry should look like the one presented in The Manager Configuration File. You don't need to be very specific with the test, but you should at least have one entry test, be it with no parameter, such as :

```
<test testConfig="benchs/jcre/engine/dup/TestConfig.xml"/>
```

2.2.2 Calibrating the tests

2.2.2.1 Typical invocation

```
~/mesure/bin$ ./calibrate.sh -vl 3
Processing arguments...
[MANAGER] > Reading the template
/home/julien/workspace/Mesure/classes/benchs/lib/templates/javacard/templ
ates.cap...
[MANAGER] > Loading the template
/home/julien/workspace/Mesure/classes/benchs/lib/templates/javacard/templ
ates.cap...
[MANAGER] > The template was loaded in 1.800385942E10 ns

[MANAGER] > Executing 1 tests...
[MANAGER] > Getting the CAP file from
resources:benchs/jcre/engine/jump/javacard/jump.cap
[MANAGER] > Reading the CAP file
/home/julien/workspace/Mesure/classes/benchs/jcre/engine/jump/javacard/ju
mp.cap...
[MANAGER] > Loading the CAP file
/home/julien/workspace/Mesure/classes/benchs/jcre/engine/jump/javacard/ju
mp.cap...
[MANAGER] > The CAP file
/home/julien/workspace/Mesure/classes/benchs/jcre/engine/jump/javacard/ju
mp.cap was loaded in 1.233801438E9 ns

[MANAGER] > Installing the contained applets...
[MANAGER] > Executing the script scripts.jcre.engine.jump.JumpScript (3
test cases)...
[MANAGER] > Preparing...
[MANAGER] > Calibrating GOTO REF...
[CALIBRATE] > Measuring the execution time for X=50 and Y=10...
[TESTCASE] Running 10 iterations of GOTO_REF
[TESTCASE] > Average measure for 2500 executions on-card [GOTO_REF]:
1.0801865648E9 ns
[TESTCASE] > Average measure for 1 execution on-card [GOTO_REF]:
432074.62591999996 ns

[CALIBRATE] > Average measure for X=50 and Y=10: 1.0801865648E9
[CALIBRATE] > Measuring the execution time for X=25 and Y=10...
[TESTCASE] Running 10 iterations of GOTO_REF
[TESTCASE] > Average measure for 625 executions on-card [GOTO_REF]:
2.76273137E8 ns
[TESTCASE] > Average measure for 1 execution on-card [GOTO_REF]:
442037.0192 ns

[CALIBRATE] > Average measure for X=25 and Y=10: 2.76273137E8
[CALIBRATE] > Measuring the execution time for X=37 and Y=10...
[TESTCASE] Running 10 iterations of GOTO_REF
[TESTCASE] > Average measure for 1369 executions on-card [GOTO_REF]:
6.005370232E8 ns
[TESTCASE] > Average measure for 1 execution on-card [GOTO_REF]:
438668.3880204529 ns

[CALIBRATE] > Average measure for X=37 and Y=10: 6.005370232E8
[CALIBRATE] > Measuring the execution time for X=43 and Y=10...
[TESTCASE] Running 10 iterations of GOTO_REF
[TESTCASE] > Average measure for 1849 executions on-card [GOTO_REF]:
8.045369996E8 ns
[TESTCASE] > Average measure for 1 execution on-card [GOTO_REF]:
435120.0646836128 ns

[CALIBRATE] > Average measure for X=43 and Y=10: 8.045369996E8
[CALIBRATE] > Measuring the execution time for X=46 and Y=10...
[TESTCASE] Running 10 iterations of GOTO_REF
```

```
[TESTCASE] > Average measure for 2116 executions on-card [GOTO_REF]:
9.136453054E8 ns
[TESTCASE] > Average measure for 1 execution on-card [GOTO_REF]:
431779.44489603024 ns

[CALIBRATE] > Average measure for X=46 and Y=10: 9.136453054E8
[CALIBRATE] > Measuring the execution time for X=48 and Y=10...
[TESTCASE] Running 10 iterations of GOTO REF
[TESTCASE] > Average measure for 2304 executions on-card [GOTO_REF]:
9.965607084E8 ns
[TESTCASE] > Average measure for 1 execution on-card [GOTO_REF]:
432535.0296875 ns

[CALIBRATE] > Average measure for X=48 and Y=10: 9.965607084E8
[CALIBRATE] > Measuring the execution time for X=49 and Y=10...
[TESTCASE] Running 10 iterations of GOTO REF
[TESTCASE] > Average measure for 2401 executions on-card [GOTO_REF]:
1.0445890061E9 ns
[TESTCASE] > Average measure for 1 execution on-card [GOTO_REF]:
435064.14248229907 ns

[CALIBRATE] > Average measure for X=49 and Y=10: 1.0445890061E9
[MANAGER] > Terminating...
[MANAGER] > Deleting the CAP file
/home/julien/workspace/Mesure/classes/benchs/jcre/engine/jump/javacard/ju
mp.cap...
[MANAGER] > Deleting the template
/home/julien/workspace/Mesure/classes/benchs/lib/templates/javacard/templ
ates.cap...
~/mesure/bin$
```

2.2.2.2 Comments

We specified on this last command the verbose level (here, -vl 3). On that level, we can witness the different steps followed by the program to perform a calibration on GOTO and GOTO_W bytecodes. The test used to calibrate these bytecodes is called GOTO_REF.

We first load the test applet on the card. Then, we start measuring the performance of the card for this test. As the calibration goes on, we can witness the evolution of the parameter X, that is the square root of the number of loops performed on the card. Finally, we terminate by deleting the test applet from the card. The result configuration file is ../config/manager/CalibrateOutput.xml and contains all the GOTO tests with a loopsize (X) of 49, that is 2401 on-card iterations.

2.2.3 Running the tests

2.2.3.1 Typical invocation

```
~/mesure/bin$ ./manager.sh -vl 3 -i ../config/manager/CalibrateOutput.xml
Processing arguments...
[MANAGER] > Reading the template
/home/julien/workspace/Mesure/classes/benchs/lib/templates/javacard/templ
ates.cap...
[MANAGER] > Loading the template
/home/julien/workspace/Mesure/classes/benchs/lib/templates/javacard/templ
ates.cap...
[MANAGER] > The template was loaded in 1.800487547E10 ns

[MANAGER] > Executing 1 tests...
[MANAGER] > Getting the CAP file from
resources:benchs/jcre/engine/jump/javacard/jump.cap
[MANAGER] > Reading the CAP file
/home/julien/workspace/Mesure/classes/benchs/jcre/engine/jump/javacard/ju
mp.cap...
```

```
[MANAGER] > Loading the CAP file
/home/julien/workspace/Mesure/classes/benchs/jcre/engine/jump/javacard/ju
mp.cap...
[MANAGER] > The CAP file
/home/julien/workspace/Mesure/classes/benchs/jcre/engine/jump/javacard/ju
mp.cap was loaded in 1.176434385E9 ns

[MANAGER] > Installing the contained applets...
[MANAGER] > Executing the script scripts.jcre.engine.jump.JumpScript (3
test cases)...
[MANAGER] > Preparing...
[TESTCASE] Running 10 iterations of GOTO_REF
[TESTCASE] > Average measure for 2401 executions on-card [GOTO_REF]:
1.040516618E9 ns
[TESTCASE] > Average measure for 1 execution on-card [GOTO_REF]:
433368.0208246564 ns

[TESTCASE] Running 10 iterations of GOTO
[TESTCASE] > Average measure for 2401 executions on-card [GOTO]:
1.0814114691E9 ns
[TESTCASE] > Average measure for 1 execution on-card [GOTO]:
450400.44527280296 ns

[TESTCASE] Running 10 iterations of GOTO W
[TESTCASE] > Average measure for 2401 executions on-card [GOTO_W]:
1.1045677854E9 ns
[TESTCASE] > Average measure for 1 execution on-card [GOTO_W]:
460044.89187838405 ns

[MANAGER] > Terminating...
[MANAGER] > Deleting the CAP file
/home/julien/workspace/Mesure/classes/benchs/jcre/engine/jump/javacard/ju
mp.cap...
[MANAGER] > Deleting the template
/home/julien/workspace/Mesure/classes/benchs/lib/templates/javacard/templ
ates.cap...
~/mesure/bin$
```

2.2.3.2 Comments

We specified on the invocation line that we needed a verbose level of 3 (-vl 3) to be able to look at what is going on. We also used the -i option to specify the default calibration output file as the configuration file for the benchmark. As a result, the module loaded the test applet onto the card, measured 10 times the three relevant tests, including the reference test, and deleted the test applet from the card. The XML result file generated is by default ../tmp/results.xml and indicates the different time measurements performed.

2.2.4 Filtering the obtained measurements

2.2.4.1 Typical invocation

```
~/mesure/bin$ ./filter.sh
Processing arguments...
[WARNING] filter: no option specified, I fall back to default
configuration (-cd).
[FILTER] Saving to XML file : ../tmp/filtered.xml
~/mesure/bin$
```

2.2.4.2 Comments

This basic use of the filtering module eliminates roughly two out of ten measurements for each test within the generated result.xml file to produce an output file with the filtered measurements only.

2.2.5 Extracting interesting measurements from raw measurements

2.2.5.1 Typical invocation

```
~/mesure/bin$ ./extractor.sh
Processing arguments...
[WARNING] extractor: no option specified, I fall back to default
configuration (-cd).

[EXTRACTOR] ---- Knowledge Base (avg/stderror)-----
[EXTRACTOR] Empty
[EXTRACTOR] Resolving...
[EXTRACTOR] Loading reference cases...
[EXTRACTOR] Warning, test case without reference case :GOTO_REF
[EXTRACTOR] statistics for "GOTO_REF" avg=1.040024383875E9 ns
stderror=5706791.916236135
[EXTRACTOR] Loading test cases...
[EXTRACTOR] populateTestCase : splitNameArray.length=1;
splitNameArray[0]=GOTO
[EXTRACTOR] populateTestCase : benchedUnits=x
[EXTRACTOR] Loading Test case : [GOTO] (x)
[EXTRACTOR] REF : GOTO_REF
[EXTRACTOR] refAvgBase = 1.040024383875E9 ns
[EXTRACTOR] TEST: "[GOTO]" avg=16997.038421491045 ns
stderror=215.36507177086057
[EXTRACTOR] populateTestCase : splitNameArray.length=1;
splitNameArray[0]=GOTO_W
[EXTRACTOR] populateTestCase : benchedUnits=x
[EXTRACTOR] Loading Test case : [GOTO_W] (x)
[EXTRACTOR] REF : GOTO REF
[EXTRACTOR] refAvgBase = 1.040024383875E9 ns
[EXTRACTOR] TEST: "[GOTO W]" avg=26885.71412952936 ns
stderror=28.321219242085988
[SOLVER] Solver : name=GOTO BenchedUnit=x
[SOLVER] Solver : name=GOTO_W BenchedUnit=x
[SOLVER] solve1 : sizeSolvedPre-1 ; sizeSolvedPost0
GOTO_W***CstNode 26885.71412952936
Id  Id x
Id1 Id x
xxx 26885.71412952936
[SOLVER] GOTO_W=26885.71412952936
GOTO***CstNode 16997.038421491045
Id  Id x
Id1 Id x
xxx 16997.038421491045
[SOLVER] GOTO=16997.038421491045
**************dumpHash**************
GOTO_W26885.71412952936
GOTO16997.038421491045
**************dumpHash**************
[SOLVER] solve1 : sizeSolvedPre0 ; sizeSolvedPost2
**************dumpHash**************
GOTO_W26885.71412952936
GOTO16997.038421491045
**************dumpHash**************
Outputing : GOTO
reference is : GOTO_REF
TC: lib.xml.test result.TestCase@10385c1
REF:lib.xml.test_result.TestCase@1729854
GOTO
Outputing : GOTO_W
reference is : GOTO_REF
TC: lib.xml.test_result.TestCase@1bab50a
REF:lib.xml.test_result.TestCase@1729854
GOTO_W
~/mesure/bin$
```

2.2.5.2 Comments

This use of the extraction module illustrates how the execution time isolation is performed. We start by building a knowledge base with the filtered results. The aim is to make a base with everything we know about the measured values. Then, we need to solve the equation system. We first parse the "benchedUnit" fields. These fields indicate what we measured exactly. By default, "x" means we just need to subtract the measure obtained for the reference test from the measure obtained for the test. For each loop within the solver, we print the value of all the deduced values, till we reach a fix point. The produced output is by default the file ../tmp/extracted.xml. This file contains the deduced values for the GOTO and GOTO_W bytecodes.

2.2.6 Displaying synthetic results and calculating the mark

2.2.6.1 Typical invocation

```
~/mesure/bin$ ./profiler.sh -i ../tmp/results.xml \
      ../tmp/extracted.xml ../config/manager/CalibrateOutput.xml \
      ../config/profiler/coefs.xml ..
Processing arguments...
~/mesure/bin$
```

2.2.6.2 Comments

We invoked the profiling program with the input parameters containing the different files needed to produce an output. The different files are given in this order. The first file contains the raw results, the second one contains the extracted results, the third one contains the configuration, and the last one contains the coefficients to be applied to the results for each bytecode and API method.

In the HTML output, the raw results are shown as well as the marks (the global mark as well as the domain specific marks). Some graphical charts are also computed. A comparison with a reference performance is made.

3 References

[F1.2REQ] MESURE Project – F1.2 Requirements (Rapport sur l'expression des besoins) January 2007 – Version 1.0.

[F2.2MET] MESURE Project – F2.2 Methodology (Rapport sur la méthodologie d'application des mesures) – October 2007 – Version 1.0.