QualiMaster Infrastructure Configuration Tool – Overview

The QualiMaster Infrastructure Configuration Tool (QM-IConf) is the central tool to define the analysis tasks running on a corresponding QualiMaster[1] infrastructure and to administer the infrastructure. This document provides a short background explanation about the underlying approach, how to install the tool and login to synchronize the configuration and describes five usage scenarios, namely:

- A walkthrough the configuration options of QM-IConf to gain an overview of its capabilities and concepts.
- The creation of a new data source.
- The use of the data source in a modified pipeline. Further modifications will provoke a validation error.
- The creation of a new simple pipeline.
- The instantiation of the modified model into source code.

The document ends with a glossary of central terms used in QualiMaster.

Background

QM-IConf aims at three important steps in the lifecycle of a Big Data analysis application:

- **Configure** the Big Data application in terms of available resources, available data analysis algorithms, grouping of the algorithms to families consisting of algorithms with similar functionality, but different quality characteristics and, finally, designing the application in terms of one or more adaptive data analysis pipelines.
- **Validate** the configuration so that the pipelines can be derived automatically and (probably) be executed correctly on the QualiMaster infrastructure.
- **Derive** the application code, in particular the pipeline implementation. Therefore, QM-IConf turns the configuration into source code as well as related artifacts, generates build scripts based on the configured algorithms and integrates all together.

Installation of a binary variant of QM-IConf

In addition to the source code, QM-IConf is available in several different binary variants, e.g., for different operating systems. In this document, we focus on the demo variant, which contains a local version of the model. However, changes to the model are local so that the repository connection used in the release variant is not fully active. The local model can be reverted, but it cannot be committed. Furthermore, we focus on the specific variant for the Windows operating system[2].

The first steps with QM-IConf consist of unpacking the distribution archive, starting the tool and, in case of the release variant, logging into the tool to start the model synchronization.

Unpack the binary variant of QM-IConf. The Windows variants contain a start program called `QualiMasterApplication.exe`, the Linux variants a start program named `QualiMasterApplication`. **Execute** the start program according to your operating system (double click for Windows, single click or command line execution for Linux).

Preparing the source code variant of QM-IConf

Currently, this document explains the application of QM-IConf using the Configuration Model developed along with the project (located in the project's code repository or part of the demo variant). The open source repository contains an empty version of the configuration model. To use

[1] The EU FP7 QualiMaster project (http://qualimaster.eu) aims at researching a configurable and adaptive real-time data processing infrastructure and at demonstrating this infrastructure for financial systemic risk analysis.
[2] The Linux version requires the correct combination of 32/64 bit operating system, JDK and QM-IConf application, in particular for running the protobuf plugin.

the empty model instead of the model described here, some manual steps are required. We will provide some bootstrapping support in one of the next versions.

- Local configuration model
  - Unpack the `model.zip` archive into a directory of your choice.
  - Create a `conf.properties` file[3] in the `QualiMasterApplication` folder and set the following properties:
    - `model-location = <location of the model>`
    - `demo-mode = true`
- Remote SVN-based configuration model
  - Create a SVN repository for the model.
  - Create a userAndRoles.xml permissions file directly with the SVN repository according to the following XML structure:

```
<root>
  <user name="sassai">
    <role id="admin"/>
    <role id="infrastructure_admin"/>
  </user>
</root>
```

    with the following roles `admin`, `infrastructure_admin`, `pipeline_designer`, `adaptation_manager`.
  - Unzip the `model.zip` archive and commit it into a sub-folder of your SVN repository.
  - Create a `conf.properties` file[3] in the `QualiMasterApplication` folder and set

    `repository-url = <your model-dir URL>`

    By default, QM-IConf will search for the permissions file in the parent folder of the URL specified in repository-url.

Before starting the infrastructure derivation process, please do not forget to the *repository URL* in the infrastructure settings, i.e., the Maven repository containing your pipeline algorithms.

Startup and Login

First a **splash screen** appears[4]. As QM-IConf is based on Eclipse, loading the application may take some time (as indicated by a progress indicator).

Fig. 1: QM-IConf login dialog

Then, except for the demo variant (or the local model setup described above), the **login screen** appears as depicted in Fig. 1. On the very first execution of QM-IConf, you must now enter your SVN

[3] Actually, `conf.properties` is ignored during code commits so that you can modify it according to your needs. The default repository URL is stated in `app.properties`, which is part of the source code repository.
[4] Due to technical reasons, in some binary variants the splash screen may appear later or not at all.

credentials[5] and press "Login". Initially, QM-IConf will download the respective version of the Configuration Model. Depending on your internet connection, this may take some time (the progress is indicated by the progress bar above the "Login" button). Upon further starts, you can either

- **Enter your credentials**, which will lead to a synchronization of the Configuration Model. Potential conflicts between repository content and local changes will be displayed and you will be asked for the preferred resolution.
- **Select the "No login - local data" option**, i.e., QM-IConf will use the bundled locally available model and not try to update this model through a synchronization with the server. However, you will not be able to commit your changes without a full login.

Now, QM-IConf will continue loading its components, i.e., the **splash screen** re-appears and indicates the loading progress. Finally, the QM-IConf main window depicted in Fig. 2 opens.

Getting a first overview

This scenario enables the user to gain a first overview on the configuration tool by hands-on learning the most important concepts. This and the following sections explain the usage based on the Configuration Model developed in the QualiMaster project.

The main window (shown in Fig. 2) consists of a menu on the top, the *configuration options* tree on the left side, the editor area in the center, detailed views on the right side and the status bar at the bottom. Depending on the variant you are executing, you may not see the following configuration options: `Observables`, `Adaptation`, or `Runtime`. In all configuration editors, mandatory input is flagged by a "*".

Let's go now stepwise through the individual configuration options on the left side to make you familiar with QM-IConf and the underling concepts. While going through the concepts, please open for the respective top-level configuration option the *subtree of configured elements* (by clicking on the respective triangle on the left) and open at least one of the respective configuration editors (by double clicking the respective tree entry). For example, in Fig. 2 you can see the specific settings for a general purpose machine (`snf-618466.vm.okeanos.grnet.gr`). New configured elements can (where possible) be created through the context menu (right mouse button) of the top-level configuration option. Operations on the respective configured element can be accessed through its

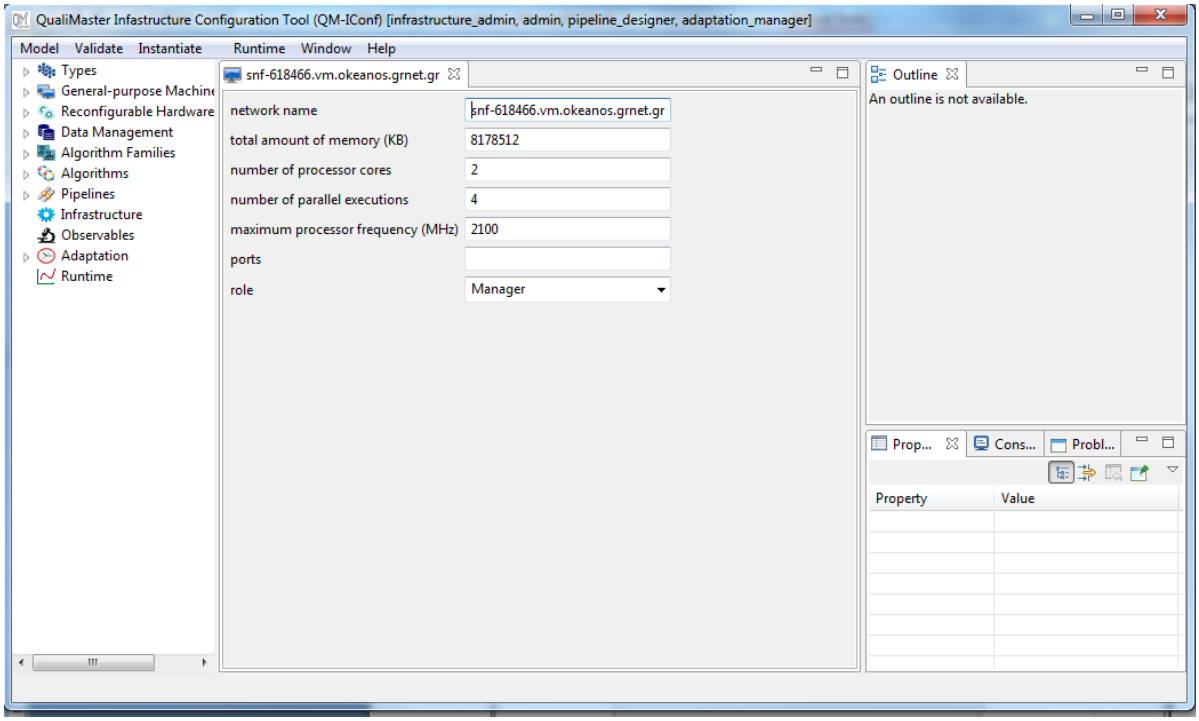

Fig. 2: Main window of QM-IConf, with menu (top), configuration options tree (left), editor area (center), detailed views (right), and status bar (bottom).

context menu. This may include cloning the selected element. Additional information on a configurable element is shown in the status bar if the mouse is placed for a certain time over the element or (in case of table-based editors only) over its label. The respective editors indicate whether changes to individual configuration settings have been made by a star symbol in the editor tab, and force a validation upon saving the editor (CTRL+S, closing the editor, Model|Save all[6]).

The top-level configuration options follow the typical path of configuring a QualiMaster infrastructure and the applications / pipelines to be executed:

- **Types:** As part of executing a Big Data application, data is passed through different data processors (algorithms). Thereby, data can be transformed, modified, enriched, etc. Here, applications can rely on predefined types such as INTEGER or STRING, but also define new application-specific types, which must be realized by a technical implementation (given as a class name and the implementing Maven[7] artifact). We will use these types to define algorithms and algorithm families below. For efficient processing, a technical helper class, a so called Serializer, can be configured (again as class and implementing Maven artifact). In QM-IConf, Maven artifacts can be entered directly or selected from the Maven repository (Browse…-Button). If the Browse…-Button remains grayed out, please check the repository settings of the infrastructure part (see below). When opening the Maven selector the first time, it reads the structure of the entire Maven repository (which may take some time) and presents a selection tree. By default, it stores the selection tree so that even offline selection of artifacts is possible. If the tree is not up-to-date (the cache expiry time is 24h), the user can explicitly for a refresh using the Refresh-Button of the selector dialog).
- **General-purpose Machines** define the pool of available execution resources for software-based algorithms, in QualiMaster mostly the compute nodes of a Storm[8] cluster. Some static information must be provided in order to support the adaptive assignment of algorithms to the compute nodes. However, the details needed for the adaptive execution as well as additional information to derive the Storm cluster configuration in order to support the administrator may change in the next months.
  General-purpose machines are automatically grouped according to their names in terms of IP range prefix or domain name suffix. The entries representing the groups are not editable. If neither a valid IP nor a domain name is detected, general-purpose machines are displayed at the end of this section.
- **Reconfigurable Hardware Machines** complement the resource pool for hardware-based execution. While the presence of General-purpose Machines is mandatory, reconfigurable computing is optional. Currently, configuring a reconfigurable machine involves technical information such as its IP address, the ports for accessing the machine for different purposes from the QualiMaster infrastructure as well as the maximum number of host CPUs and units for hardware-based execution (here, so called Data Flow Engines, DFEs).
- **Data Management** takes care about providing input data to the data analysis (data source), storing intermediary data on demand, and, finally, making analysis results available to end users (data sink). Data sources and sinks are implemented by some software (denoted by a class name and its implementing Maven artifact), an optional description for future search functionality, some information whether and how the related data shall be stored automatically and, most importantly, the data fields it handles and the runtime parameters it accepts. Further, runtime constraints on the measured characteristics can be given in order to force the adaptive execution in case of violated constraints. These three parts, data fields, parameters and constraints are displayed as tables, which can be manipulated through changing the existing table entries (click into the desired cell) or through the context menu of

[6] We use this notation to denote main Menu entries, here use the Model menu and the Save all sub-entry.
[7] https://maven.apache.org/, a build system for Java projects with dependency management support based on so called Artifacts denoted by a qualified name and located in a repository.
[8] http://storm.apache.org/, a flexible stream processing environment.

the table. However, before using the context menu, you need to select the row you want to manipulate, e.g., by clicking into the leftmost area of its first cell. This is illustrated in Fig. 3. Constraints are specified in the extended Object Constraint Language used for defining constraints in the Integrated Variability Modeling Language IVML of the underlying product line toolset EASy-Producer[9]. Currently, the "Upload" button is not enabled, but the next version will allow taking over information like data fields or parameters directly from the implementation to ease the configuration. Also the "Browse" button for selecting the implementation class (akin for algorithms below) is currently disabled as this will be enabled as part of the integration of the "Upload" button.

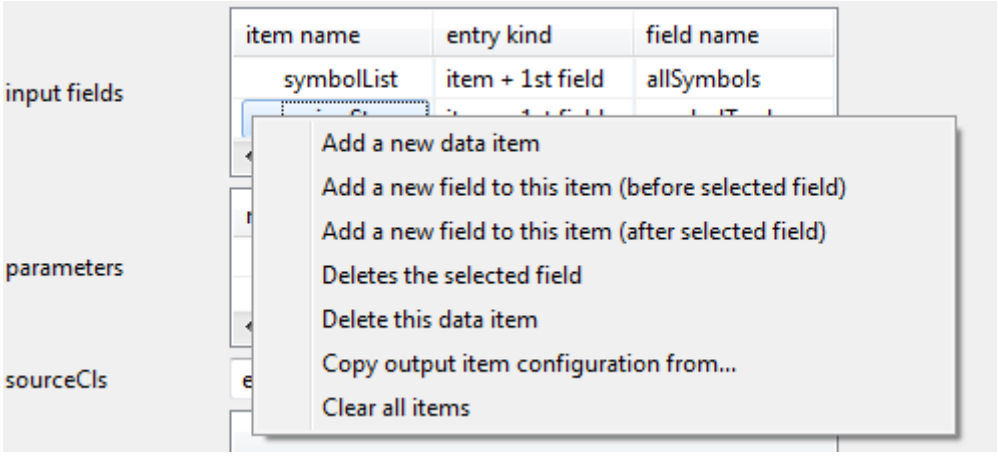

Fig. 3: Manipulating input / output data items (similarly for parameters and constraints).

- **Algorithm Families** represent groups of algorithms with similar behavior but different quality characteristics. Akin to Data sources and Data sinks, algorithm families specify an optional description, the input data fields, the output data fields and the runtime parameters. Further, the members of the actual family can be specified.
- **Algorithms** implement the actual data processing. Basically, an algorithm requires filling the same configuration entries as a family to enable validation of families and their related algorithms. To be valid, an algorithm must match the input and output fields of its family as well as the runtime parameters. In addition, an algorithm can specify whether it runs (only) on a dedicated hardware node and, thus, becomes a hardware-based algorithm, whether it is a complex distributed algorithm (in that case it must specify a topology class defining the distributed structure) and a required successor node in case that an algorithm implementation covers connected parts of a pipeline (this is a specific case for hardware-based algorithms). Akin to data sources and sinks, we will enable in the future the artifact selector to take over most of the data directly from the selected implementation.
  Algorithms are grouped by the families declaring the respective algorithms as member. Thus, an algorithm may occur multiple times, but leads to the same configurable element. Akin, opening a family in this section leads to the editor of the family. Algorithms not used by any family are displayed at the end of this section.
- **Pipelines** define the Big Data application in terms of a data flow graph, which connects data sources, algorithm families, data management elements (for storing temporary data) and data sinks. Data flows connecting the elements can act as implicit filter, i.e., if output items of the source element of the flow are not present in the downstream target element, this implies filtering. In contrast to the configuration editors utilized before, the pipeline editor is a graphical editor, which allows you to define the data processing in a drag-and-drop manner. Here the detailed views on the right side become active, in particular the outline view (showing a scaled view on the whole pipeline) and the properties view after selecting

[9] For more details, see https://github.com/SSEHUB/EASyProducer and its release page where you can also obtain the IVML language specification.

the pipeline (background) or individual pipeline elements. To be valid, the inputs and outputs of the connected pipeline elements must match, e.g., the data fields provided by a data source must match (at least overlap due to the implicit filtering of data flows) the expected data fields of the downstream algorithm family. We will come back to this specific editor in the pipeline configuration usage scenarios (task 3 and task 4).

- **Infrastructure** is the top-most view on the running QualiMaster infrastructure. It allows configuring the active pipelines, i.e., which of the configured pipelines shall be executed (some may be in preparation or outdated), the (target) artifact containing all pipeline interfaces, the (target) model artifact containing the actual deployed version of the Configuration Model (so that the QualiMaster infrastructure can make use of it) and the URL where the Maven repository with all required artifacts is located.

As mentioned above, further top-level configuration options, which do not occur in the demo variant of QM-IConf, such as `Observables`, `Adaptation`, or `Runtime` are out of the scope of this evaluation.

Please note that configuring Data sources, sinks and algorithm families before the individual algorithms is intended. The idea is that an administrator / pipeline designer shall first entire the required (input / output) data fields and runtime parameters, which actually define the technical interfaces of data sources, data sinks and algorithms. QM-IConf allows deriving this configuration into the technical interfaces, which can be passed to the algorithm providers (developing new algorithms or wrapping existing ones). Thus, the pipeline designer can define pipelines without having implementing algorithms at hand and the administrator can add algorithms as soon as they become available.

Please note further that in the menu bar of the application, the entries of the runtime menu may be disabled in demo mode while they are available in all other modes. Furthermore, configuration tree entries such as Observables, Adaptation or Runtime may not be visible in demo mode.

Creating a Data Source

Based on the concepts and overview explained above, the aim of this usage scenario is to create a new data source. We will base the settings on an existing data source so that the new source can seamlessly be used with existing pipelines.

Navigate to the top-level configuration option `Data Management`, use the context menu and `Add a 'Data Source' to 'Data Management'`. Configure the new data source as follows (please consider upper and lower case as given below):

- Name: `MySource`
- Artifact: `eu.qualimaster:spring-client-simulator:3.0-SNAPSHOT`
- Storage Strategy: `None`
- Input fields (using the context menu you have to create an item, which contains a first field):
  - First item name: `symbolList`
    - First field
      - name: `allSymbols`
      - type: `STRINGLIST`
      - Retrieval key: only relevant to denote the database keys when storing temporary data
  - Second item name: springStream
    - First field
      - Name: `symbolTuple`
      - Type: `STRING`
      - Retrieval key: see above
- sourceCls: `eu.qualimaster.algorithms.imp.correlation.SpringClientSimulator`

Store your data source using STRG+S or by closing the editor. To be on the safe side, run the validation by Validate|Validate All. No validation errors shall occur.

Modifying a Pipeline

Open the QualiMaster priority pipeline (PriorityPip). We will first replace the data source by the new data source created above and validate the pipeline. Then, we will introduce a conflicting change that cannot be validated. Finally, we will fix the conflicting change and save the pipeline.

- Select the FinancialDataSource and change in the properties the data source to MySource. Store the pipeline using STRG+S or by closing the editor. To be on the safe side, run the validation by Validate|Validate All.
- Create a family element, that will not match anywhere to provoke a validation error
  - Drag a new family element from the pallet and place it below FinancialDataSource and Preprocessor. Select the family element and configure in the properties view the name of the family to unmatching and the processing family to fMismatchedFamily (a family that fits into nowhere in this Configuration model). So far, the pipeline is still valid as there are no data flows to / from the new processing family.
  - Drag a new flow from the pallet and connect FinancialDataSource with unmatching (select the flow and change its name in the properties view to uf1).
  - Drag another new flow from the pallet and connect unmatching with Preprocessor (select the new flow and change its name in the properties view to uf2).
  - Store the pipeline using STRG+S or by closing the editor. Perform a validation and Validate|Validate All and you shall receive an error message pointing to unmatching.
- Fix the problem by deleting the problematic family element change.
  - Delete the unmatching family (mark the family in the pipeline editor and pres DEL), which will also delete the related flows.
  - Store the pipeline using STRG+S or by closing the editor. Perform a validation and Validate|Validate All and there shall not be any problems.

Creating a Pipeline

In this section, you will create a new pipeline. As the QualiMaster priority pipeline is already a complex structure, we will create a smaller pipeline (basically a test pipeline which pipes random data through a family to a sink). All pipeline elements can be dragged from the palette. Detailed information must be entered in the properties details view after clicking the respective element. Create a pipeline with the following settings:

- Pipeline
  - Name: MyPipeline
  - Artifact: eu.qualimaster.test:MyPipeline:0.0.1-SNAPSHOT
  - Numworkers: 1
  - Run in debug mode: Upon your choice, enables or disables log output.
  - Use fast serialization: Upon your choice, enables the serialization preferred by Storm or uses the fallback with the slower default Java serialization.
- Source:
  - Name: src
  - Data source: Random Source
- Family element:
  - Name: processor
  - Processing family: switchFamily
- Sink:
  - Name: snk

    - Data sink: `Random Sink`
- Flow (from src to processor):
    - Name: `f1`
    - Grouping: `shuffleGrouping`
- Flow (from processor to sink):
    - Name: `f2`
    - Grouping: `shuffleGrouping`

Store the pipeline using STRG+S or by closing the editor. To be on the safe side, run the validation by `Validate|Validate All`. Actually there shall not be any validation errors, but if there are some, please try to fix them.

Finally, go into the top-level configuration option Infrastructure and enable `MyPipeline` as an active pipeline. Store the configuration using STRG+S or by closing the editor.

Instantiating the Infrastructure

Having a valid model at hands, we can now instantiate the model, i.e., turn the model into source code and related artifacts and to prepare the deployment of the pipelines. This will include the new pipeline that you have created above (please ensure that it is marked as an active pipeline in `Infrastructure`).

Start the instantiation by `Instantiate|Instantiate local`. If there are any editors with changes, QM-IConf will ask you whether the changes shall be saved, and finally, QM-IConf will perform a validation. If the validation is successful, QM-IConf will ask you to select the target folder for the instantiation on your hard disk, i.e., a folder where the generated sources, artifacts and deployable archives shall be stored. After the selection of the target folder, QM-IConf performs the instantiation of the model. You can either follow the console output or have an eye on a file system browser for the target directory. Depending on the number of pipelines to be instantiated and the power of your computer, the entire process may run 45 seconds or longer. Thereby, various source code, data and Maven build specifications are created. The build specifications are executed to integrate the algorithms with the pipelines and to obtain deployable artifacts. You may browse through the generated artifacts, e.g., using your favorite tools.

At this stage, the configuration and instantiation support of QM-IConf ends. Deploying the generated pipelines into a production pipeline repository would be a next logical step. Currently, this is done by the continuous integration server, which runs the instantiation process in a headless manner and finally deploys the resulting pipelines, but shall become part of the functionality of QM-IConf. Using QM-IConf as a runtime dashboard is currently under development and out of scope of this evaluation.

Glossary

(Infrastructure) Administrator: Role in the QualiMaster project filled by persons administering the infrastructure, defining the algorithms etc (except for pipelines).

Algorithm Family: A set of algorithms with the same or rather similar semantics but different quality tradeoffs. A human user (infrastructure administrator) configures the algorithms, which belong to a certain family.

Algorithm Provider: Role in the QualiMaster project filled by persons developing specific data analysis algorithms according to given interfaces defined in the Configuration Model.

Configurable Element: Something that can be configured, typically characterized by certain limitations and interdependencies given in the Configuration Meta Model, e.g., an algorithm family cannot be used at a certain position within a pipeline if the input/output fields of the family does not match the predecessors and successors in the pipeline structure.

Configuration (Meta) Model: The QualiMaster Configuration Model captures the user decisions on settings (configurable elements) of the QualiMaster Infrastructure, including pipelines, algorithms to use and available hardware. Not visible to the user is the Configuration Meta Model, i.e., the definition of the configurable elements, their characteristics and inter-dependencies / constraints defining the rules for validation.

Data source: An element representing the input of data for a data analysis. Typically, a data source is implemented by some code to retrieve the actual data.

Data sink: An element representing the end of the processing, i.e., the end of a pipeline. Typically, a data sink is implemented by some code transferring the data to an externally visible web server.

Derivation: Based on the configuration model and the captured user settings, the tool derives automatically the software artifacts needed to execute the configured pipelines.

General-purpose machines: Computers used for software-based execution, such as standard server machines.

Hardware-based execution: Algorithms implemented for specialized hardware, typically accelerating the algorithm significantly. In QualiMaster, such algorithms are created through a hardware design process an algorithm and executed on a Maxeler Data Flow Engine.

Instantiation: see Derivation

Resource Pool: The available resources for computing data analysis consisting of general-purpose hardware (e.g., server machines) for the execution of software-based algorithms as well as reconfigurable hardware machines for running the hardware-based algorithms.

Software-based execution: Execution of analysis algorithms in terms of usual software programs, in our case as implementation of extensible components of the Apache Storm framework.

Pipeline (also Data Analysis Pipeline or Adaptive Data Analysis Pipeline): Describes the data analysis application to be performed in terms of a data flow graph, consisting of data sources, algorithm families (for the processing) and data sinks. In addition, transparent data management elements can be used to store intermediary data.

Pipeline Designer: Role in the QualiMaster project filled by persons defining data analysis applications, i.e., pipelines.

Reconfigurable hardware machines: Specialized hardware co-processors, such as Maxeler Data Flow Engines used in QualiMaster.

Validation: Automatic analysis of the configuration with respect to the constraints defined on its Configuration Meta Model. To be valid, all constraints must be fulfilled.

The research leading to these results has received funding from the European Union Seventh Framework Programme [FP7/2007-2013] under grant agreement nr. 619525.