JMulTi/JStatCom - A Data Analysis Toolkit for End-users and Developers

Technology White Paper

JStatCom Engineering, www.jstatcom.com

by Markus Krätzig, June 4, 2007

Abstract

JStatCom is a software framework that bridges the gap between the Java world and special purpose programming languages for data analysis, math, and statistics, like for example **Aptech's Gauss**.[1] It provides an extendable communications interface to a number of popular statistics packages, a very flexible event-driven and thread-save data model, integrated symbol management, configurable Swing GUI components, and an application programming model. It can thus be used for the rapid development of specific data analysis tools, which are typically either rich client desktop applications, web components embedded in an EJB solution, or clients in a Java-based parallel processing environment.

JMulTi is a popular Open Source tool for econometric modeling, especially time series analysis, that was build with JStatCom.[2] It uses the **Gauss Runtime Engine** for its numerical computations and combines it with a user friendly Java GUI, offering project management, innovative input selection components, and powerful data handling features. Due to its modular structure, JMulTi can itself be extended and serve as a platform for building sophisticated rich GUI clients for econometric analyses.

[1] www.aptech.com

[2] JMulTi is licensed under the General Public License (GPL) and is available from www.jmulti.com.

1 Introduction

Today, the market for data analysis software is populated by many software vendors and Open Source communities, offering a wide range of very different products, from spreadsheet calculators to domain specific GUI applications and special purpose programming languages. Although some products have a significant market share, there is no such concentration as, for example, in the market for office software. This heterogeneity is driven by very specific demands that in turn attract the development of specific software solutions.

One common feature of software development in that domain is that it typically requires considerable expert knowledge to implement the required algorithms. It is also often directly related to research in the field. To meet the needs of algorithmic development and to speed up coding, special purpose programming languages have been developed wich typically offer a convenient syntax for math and especially linear algebra expressions, optimized performance for certain matrix operations, many domain specific functions and powerful plotting capabilities.

For domain experts, it is typically more effective to code in such a programming language than in a general purpose language, like Java. However, domain specific languages offer a limited set of features and often need to be embedded in a solution that uses a general purpose language for GUI building or web development. JStatCom fills this gap by providing a Java environment that allows to easily embed calls to external computational engines. Furthermore, it provides a programming model that has proven to be flexible and convenient for the development of data analysis tools. Certain aspects of the framework are highlighted in the following sections.

2 JStatCom System Overview

The basic components that make up a typical runnable program based on JStatCom are shown in Figure 1. The application, for example JMulTi, uses features provided by the framework to setup a GUI, to parse data files of different formats, to manage projects, and so on. In JMulTi there is code that configures components provided by JStatCom and that implements application specific logic, like gathering all user input for a certain econometric model.

Furthermore, JStatCom provides the prerequisites to let JMulTi call Gauss procedures which are defined in external files. The Gauss code contains the procedures for the needed estimation and testing procedures, as well as routines for graphical output. To run the Gauss code, some engine specific executables must be available, which for Gauss is either an installed version, or the Gauss Runtime Engine, a distributable library version.

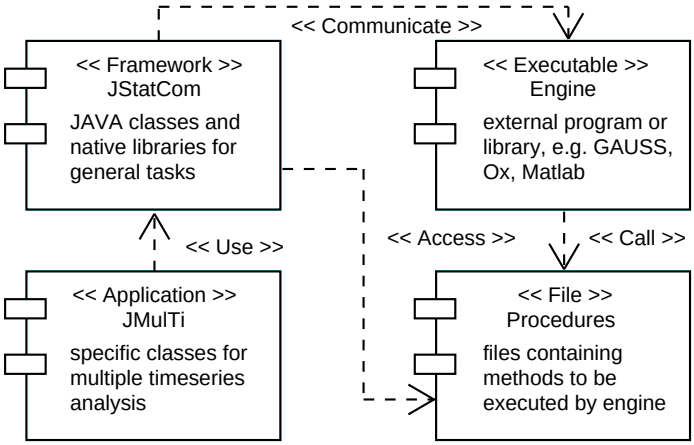

Figure 1: Components of JStatCom

JStatCom consists of several subsystems that depend on each other. Every subsystem is related to a typical requirement for data analysis software. The different subsystems are separately be looked at in the following.

2.1 Data Model

JStatCom needs to represent data internally, because it maintains inputs and results of numerical computations. Furthermore, it must be easy to let data objects interact with GUI components that display or change the underlying values. Therefore JStatCom provides special editing components for each data type. For example, it is rather simple to use a table that enables users to view and edit number matrices intuitively.

The data objects that are used within JStatCom on the Java side must conform to the types that are used by a specific engine. For example, a 2-dimensional number array in Java has a natural counterpart in Gauss.

However, the Gauss procedure does not know anything about the Java data type but expects a specific input value, which is in fact some C structure. JStatCom handles the required type conversions in both directions automatically. The idea is to have a consistent data management system within the framework that can contain various different types to adjust to any potential modeling situation. Any technicalities and API specific details are hidden from the developer.

2.1.1 Type System

JStatCom uses a metadata model to represent values of different types. Core attributes are standardized for all data types by defining a very general interface `JSCData`, which all specific types must implement. The `JSCData` interface only specifies methods that are common to all potential types. Any specialized functions to access or modify the contents of data objects are defined in implementations of the interface.

Figure 2 shows the complete interface and most types that are currently implemented. For the sake of clarity, only very few methods of the actual data classes are given, a complete documentation can be found in the API documentation. It should be noted that the implemented types are responsible to facilitate interaction with GUI components and to operate as storage units, instead of carrying out computations on them directly. For example, the `JSCNArray` class is a basic matrix class for JStatCom, but it does not try to compete with existing Java matrix implementations for linear algebra calculations. The benefit is that the interfaces of all types are kept quite simple. However, data can easily be moved from `JSCData` types to instances of specialized math classes. But typically sophisticated linear algebra calculations are done with the computational engine, which is especially suited and optimized for that purpose.

A distinguishing feature of the JStatCom Type System is that listeners can be installed on all instances of `JSCData`. This makes it easy to implement specific actions that are triggered when a data object changes its value. Another feature is that all data objects can take an empty state where no value is stored. Listeners can also be notified about changes from empty to non-empty, which is much more efficient and in most cases sufficient as a trigger event.

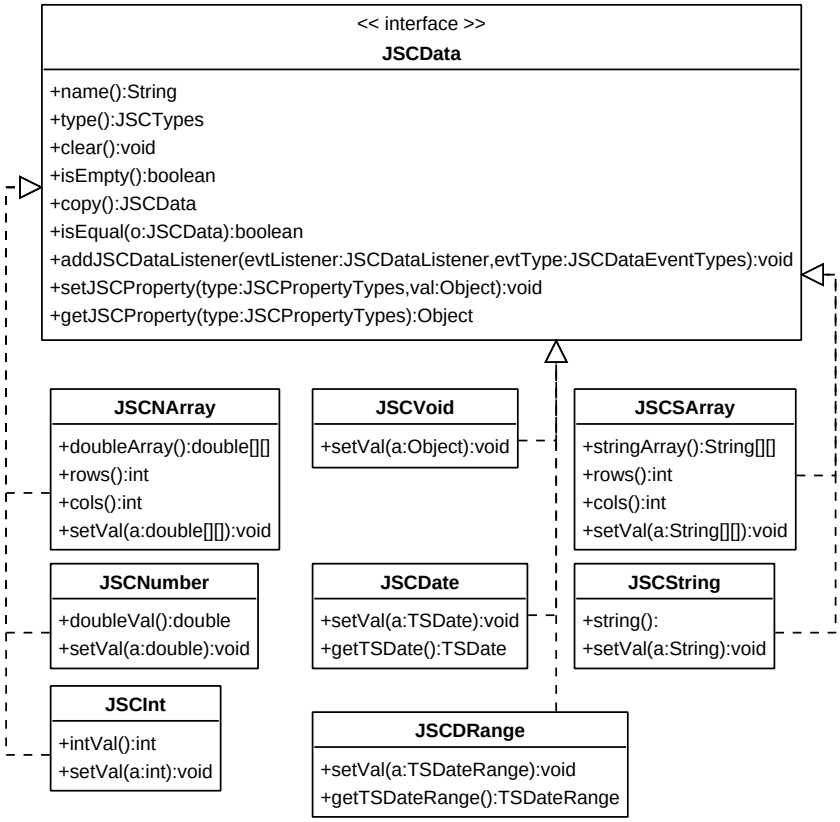

Figure 2: Type System

2.1.2 Symbol Management

The Type System introduces various ways to store and manipulate data of different kind. However, a common problem when designing applications for complex models is that various classes and GUI components need to share data values. The user interface typically consists of several components that handle different modeling steps, like specification, estimation, diagnostics and forecasting. All these components need to have access to the model state. It would certainly not be a good idea to exchange data directly between these components, because this would create unnecessary dependencies among them.

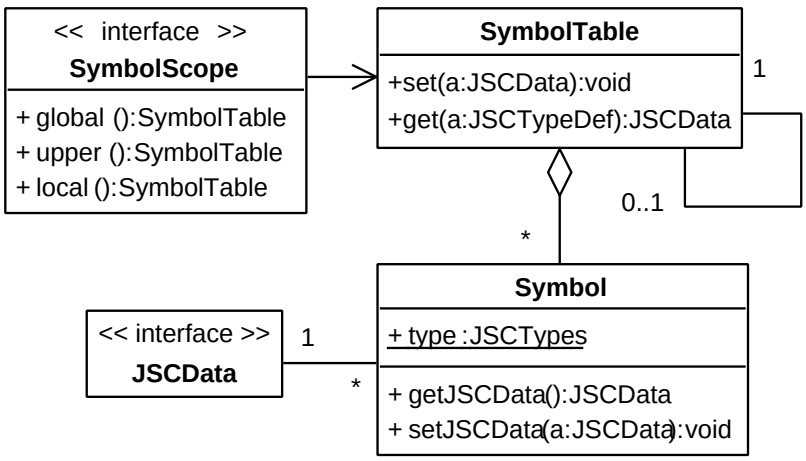

Figure 3: Accessing shared data repositories

Figure 3 gives a simplified overview of the Symbol Management system which is the JStatCom solution to address this issue. The class `SymbolTable` is an aggregation of an arbitrary number of `Symbol` instances. Each symbol object represents exactly one instance of `JSCData`. Symbol objects are identified via their name in the symbol table, which operates as a shared data repository. Via the symbol table it is possible to access the symbol elements and finally the actual data values. Symbols can be understood as pointers to variables. The referenced values, instances of `JSCData`, can be changed efficiently during runtime. The `SymbolTable` can represent the state of arbitrary models as an aggregation of symbols of different types. Therefore it is used to represent arbitrary

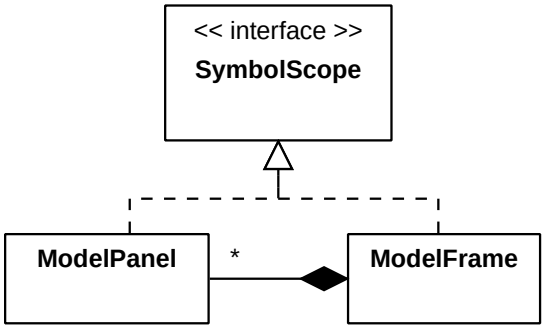

Figure 4: SymbolScope inheritance

models and to access the model data from various GUI components without creating dependencies among them.

One might ask, whether this is not just another way of introducing global data. In a way it is, but there is another part of the Symbol Management system which allows for fine-grained definition of access scopes. JStatCom offers a way to limit the visibility of symbol tables to components that belong to one model. Furthermore, it is possible to share data on different levels, which is somewhat similar to global and local variables. For this, the interface `SymbolScope` is provided. Implementations of this interface have access to symbol tables on three different levels: global, upper and local. Every symbol table keeps a reference to the next higher symbol table in the hierarchy defined by implementations of `SymbolScope`.

To be more specific, Figure 4 shows, how the `SymbolScope` interface is implemented by components of the model. Every model should be implemented with a `ModelFrame` as the top level GUI container. This can be the starting point for any application based on JStatCom. A `ModelFrame` is typically a composition of a number of `ModelPanel` containers which hold the buttons, labels, tables, and other components. Both classes provide access to the Symbol Management system and can use it to set and retrieve variables. The `SymbolScope` interface imposes a hierarchical ordering of symbol tables. The `ModelFrame` and `ModelPanel` implementations of this interface use the component hierarchy for this.

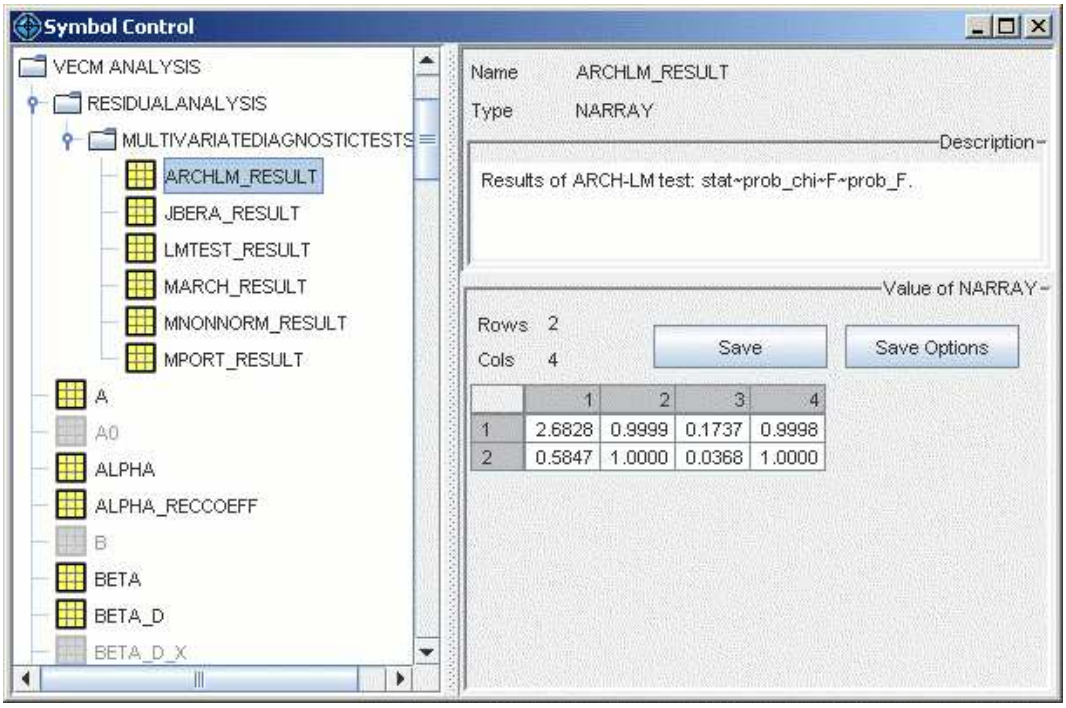

Figure 5: Screenshot of symbol frame with selected `NARRAY`

Developers only need to understand that `ModelPanel` instances can be used to define access scopes. One could also think of other possible implementations of `SymbolScope`, reflecting different hierarchical schemes. However, for the purpose of GUI building this solution has proven to be very fruitful. Components may even register listeners to symbols which are notified in synchronization with the Event Dispatching Thread.

Storing data in symbol tables is not only meaningful when variables should be shared, but it can also be used to publish results in the Symbol Control system, which is another subsystem of JStatCom that provides access to variables that are currently used. Figure 5 shows a screenshot of the graphical component. It presents a tree view of the symbol table hierarchy and it has components to display and export all symbols that have been put in one of the symbol tables.

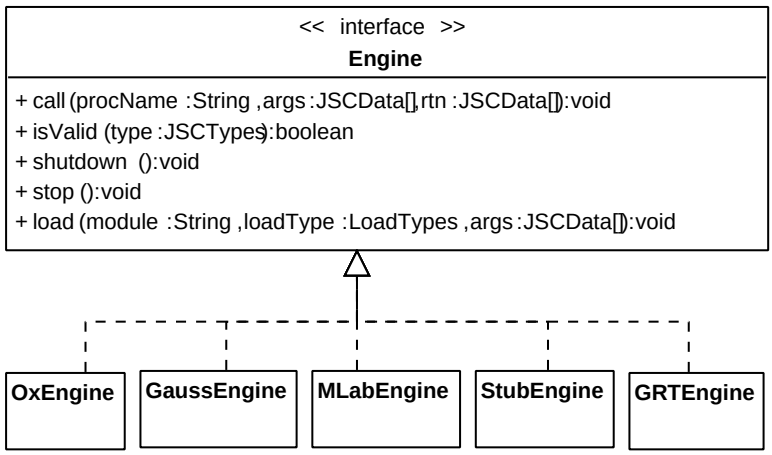

Figure 6: Engine interface and available implementations

2.2 Engine System

This section introduces the system for communicating to different execution engines. Typically these engines rely on external resources, which means that extra software packages or libraries must be installed. There are software vendors who provide redistributable stand-alone versions of their computational engine, for example Aptech with the Gauss Runtime Engine. The advantage is that users do not need any extra packages to be installed on their computer.

The framework provides access to different engine implementations via a unified interface. Figure 6 presents the complete interface `Engine` and all implementations currently available. Clients should use the engine only via its abstract implementation, thus making similar calls for every implementation. The solution found manages to integrate engines with very different characteristics and calling conventions.

2.3 Component System

JStatCom offers a number of innovative Swing components that closely work together with the Symbol Management System but can also be used without it. For example, sophisticated table components can be easily configured by just setting the name of a symbol that should be displayed

and/or edited. A large set of predefined renderers, editors and mouse listeners exist to configure table displays quickly and reliably. As an example, see Figure 7 with tables displaying user selectable subset restriction matrices in an equation system.

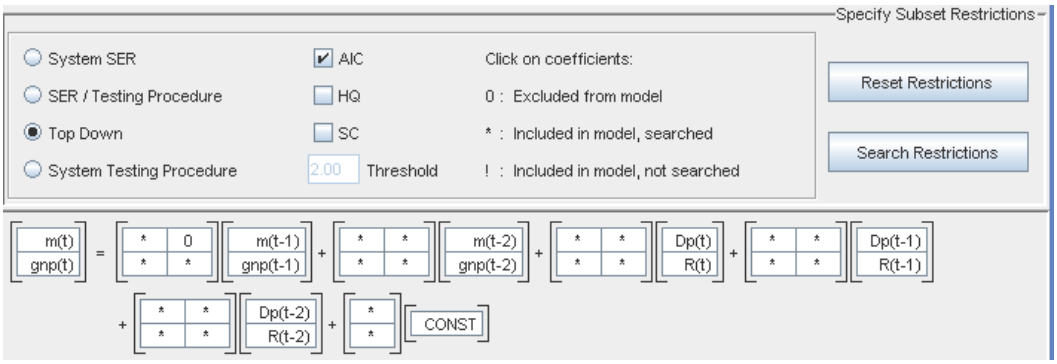

Figure 7: Screenshot of manual/automatic subset specification for the VAR analysis in JMulTi

3 Conclusion

This whitepaper just scratched the surface of what can be done with JStatCom. The interested reader is pointed to the API documentation, the tutorials and the architecture documentation.

JStatCom can be used as a key component in software development projects for data analysis tools using Java. It provides a flexible yet powerful programming model that relieved developers from many time-consuming tasks. JStatCom is Open Source and can be extended and modified by users under the terms of the Lesser General Public License, and is therefore business friendly, as it may be used in closed source projects as well.

JStatCom Engineering offers professional support and services related to the framework.