Standard efficient Storage of Simulation Results

Dr. Ingrid Bausch-Gall
ingrid.bausch-ball@bausch-gall.de
BAUSCH-GALL GmbH
Wohlfartstraße 21 b, 80939 München, Germany

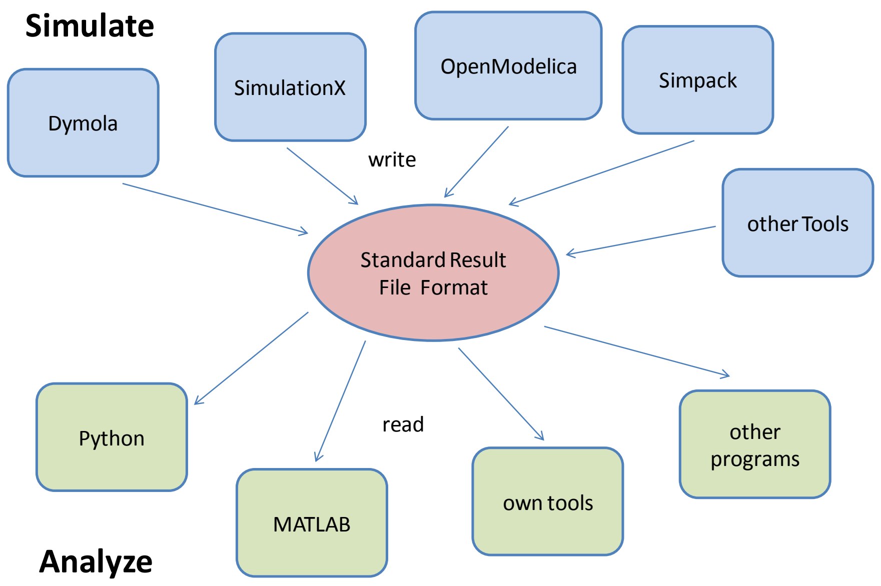
Andreas Pfeiffer
andreas.pfeiffer@dlr.de
Deutsches Zentrum für Luft- und Raumfahrt (DLR)
Institut für Robotik und Mechatronik
Münchner Straße 20, 82234 Weßling, Germany

Abstract

The problem is well known: Each simulator stores its results in its own format. If you want to use these results, you have to write own programs. The access to results may be necessary for evaluation of the simulation results or to compare these with results from another simulator or from measurement. Often the format of the result file is not open and the user has no access to the data. Many simulators provide export of ASCII or CSV files, which makes data access easy. In most cases, however, information supplied in the file is not complete. Also, reading of those files is inefficient and storage of huge amounts of data is almost impossible. In this paper we report about the development of a standard result file format (SRFF) for system simulators. This format is currently developed by the Modelica Design Group [2]. We show the direction into which this development will lead, talk about the advantages of such a format and discuss open issues.

1 Goal

Modelica [2] is a non-proprietary, object-oriented, equation based language to conveniently model complex physical systems containing, e.g., mechanical, electrical, electronic, hydraulic, thermal, control, electric power or process-oriented subcomponents. It will be supported by a growing number of simulators and more and more libraries are developed in Modelica. It is necessary to run these libraries and models with different simulators and test the results. Tests are also necessary, if new versions of simulators are released. Appart from testing, it might be desirable to collect results of parts of a model, which were computed by different simlators, as this is done e.g. in a cosimulation environment. Currently all this is quite clumsy, as each simulator offers a different and a more or less mighty environment for result evaluation. To allow a comfortable evaluation of simulation results with common scripting tools as MATLAB or Python, we started to develop a standard format for result files, which should be supported by as many simulators as possible. The goal is shown in following diagram:



Simulator developers could offer export to a SRFF file additional to other output formats. Many Modelica simulators support the FMI standard [1]. This standard allows to export models for import into other simulators or for cosimulation. The exported model consists of an XML-file which includes information about the model and the variables and of a DLL which includes the code. The definitions included in the SRFF should be exactly as in the FMI standard. SRFF shall be also the basis of a test environment to be developed for or by the Modelica Association for tests of the Modelica Standard Library.

The description of the file format and prototype programs to write and read the SRFF file will be distributed under the BSD-like Modelica license.

2 Requirements

A Standard Result File Format has to fulfill the following requirements:

- The file format must be an open and internationally accepted standard.
- Huge amounts of data (more than 10 Gbyte) must be written fast and efficiently.
- Extraction of data from huge files must be fast.
- The format has to be also accepted by simulator developers outside of the Modelica community.
- It has to be future proof, which means stable support by the developers of the standard is expected and it has to be supported by many tools.
- Data types should include:
  - all data types of [1]: double, boolean, all types of integer, strings
  - other data types to be discussed: real, structures, graphical information such as model diagrams.
- API to standard programming languages like C, C++ and FORTRAN should exist.
- Easy access from scripting programs as Python, MATLAB and others is needed.

3 Examined Data Formats

All examined data formats are binary formats, which are defined and supported by stable groups.

3.1 ASAM Data Formats

ASAM is the Association for Standardisation of Automation and Measuring Systems [5]. It was founded 1998 as an initiative of German car manufacturers with the goal of offering a platform for the development of universal standards. We looked into several standards of ASAM, such as MD-2 MC, HIL V1.0.1 and ASAM ODS V5.2.0.

A standard like ODS V5.2.0 was designed to store data coming from tests. These result data have about the same structure as those from simulation. Programming interfaces exist and the standard is widely used in car industry, also by suppliers of measurement software as National Instruments since several years. The standard is actively developed and the next release is scheduled for 2012. Other ASAM standards have similiar properties.

A standard from ASAM is not suggested here, because it is not possible to download and use the standard without being a member of ASAM. The license conditions are restrictive and the standards are not supported by MATLAB and Python. Furthermore, the standards are not accepted by many international scientific users.

3.2 NetCDF and HDF

NetCDF (Network Common Data Form) [6] is a set of software libraries and self-describing, machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data. The actual version 4.1.3 allows the use of the HDF5 data file format. Advantages of NetCDF are simple programming, efficient access to subsets of huge amounts of data and the support by MATLAB and Python. As disadvantages of NetCDF we see an inflexible usage of arrays with growing size. Furthermore new features of HDF5 are included in NetCDF, so it looks like HDF5 will replace NetCDF.

HDF, HDF4 and HDF5 (Hierarchical Data Format) [4] are names of a set of file formats and libraries designed to store and organize large amounts of numerical data. Originally developed at the National Center for Supercomputing Applications (U.S.A.), it is currently supported by the non-profit HDF Group, whose mission is to ensure continued development of HDF technologies and the continued accessibility of data currently stored in HDF. The HDF format, libraries and associated tools are available under a liberal BSD-like license. HDF is supported by many commercial and non-commercial software platforms, including Java, MATLAB, IDL, and Python. The freely available HDF distribution consists of the library, command-line utilities, test suite source, Java interface and the Java-based HDF Viewer. The currently existing two versions HDF4 and HDF5 differ significantly in design and API. We decided to look more thoroughly into HDF5.

As advantages we see that all dimensions in any number of objects might not be known in advance and the corresponding arrays might be constructed incrementally. This is needed by simulation programs. Many native data types are supported. Zipped files, graphics and videos can be stored in HDF5. Many scripting and mathematical tools support HDF5. MathWorks started in MATLAB Version 7.3 to write the mat-files optionally in HDF5 format. There are no limitations on the number and size of the objects stored in an HDF5 file. Use cases show that file sizes can be up to 1 Tbyte. We therefore expect this format to be future proof.

After these investigations we decided to use HDF5 for further investigations.

4 Suggested Standard Result File Format

We need to store information about the simulation model, the solver, information about the data as defined in [1] and the simulation results themselves. Discrete and continuous simulation results occur at different times, therefore it has to be possible to store several time series to save storage space. Results in the frequency domain should be stored in the same file. The suggested SRFF contains three sections:

- information to store all the informtion about the model, the solver and the specific simulation run
- variables to store the information about all variables and the data of parameters
- several series to store the results for the specific vector of the independent variables

The independent variable is stored as the first column of the relevant real data array. Structure of the suggested SRFF:

```
<file-Name>            // as default: use "modelIdentifier"
   Information         // fmiModelDescription.attributes as hdf5 attributes
      SimulationSetup  // as attributes are e.g.:
        startTime
        stopTime
        algorithm      // integration algorithm or algorithm of frequency calculation
        fixedStepSize  // for solvers with fixedStepSize
        relativeTolerance
      TypeDefinitions
      UnitDefinitions
      Further information as author, generationTool, version, description, etc.
      VendorAnnotations

   Variables
        // here information about all variables will be stored, e.g.
        // name, variability, of which type they are, etc.
        // Independent variables must be part of ScalarVariables, e.g.
        //    "time" (for continuous time), "discreteTime" for "discrete variables".
        <...>   // for variability "constant" or "parameter", the value is a data
                // value
        <...>   // for variability "continuous" or "discrete" a pointer shows to
```

```
             // the column of the specific data array

Series1      // variability ="continuous"
             // (attribute: independentVariable = <Name of  ScalarVariable>)
   Real      // independent variable as first column
   Integer
   Boolean
   String

Series2      // optional variability="tuneable" and "discrete"
   Real      // independent variable as first column
   Integer
   Boolean
   String

    ......

 SeriesN   //  optional results in the frequency domain
    ......
```

5 Status of Evaluation

We developed programs to write and read files up to 10 GByte. The contents of the files can be investigated with HDFview. Writing huge files needs only a little bit more time than the speed of the specific hard disk allows. Reading complete vectors of the signals or parts of the signals from 10 GByte files needs only some seconds. Writing was done with C and Python, reading with C, Python and MATLAB. Writing the files from C is about 50% faster than from Python. But it is much easier to develop programs in Python than in C.

5.1 Open issues

The standardization effort is still in progress. The current discussion includes the following subjects:

- Will it be better to write everything into one file as suggested above or should we use at least two files, an XML-file to define signals and attributes as in [1] and a data file in the HDF5 format including only the data?
- Which data formats will have to be supported and what is the most efficient way to do this? How to store booleans? Is it necessary to store single precision data? Is it necessary to support all types of integer data?
- How will timeBase or clock of discrete time series be defined?

- How will simulation results in the frequency domain be stored?

The standard will be published on [3]. Simple programs to read and write the files will be published under the liberal license of the Modelica Association. We expect to finish this work until end of 2011.

Summary and Outlook

We described how simulation data could be stored efficiently in a standard way. The standard is still under development in the Modelica Design Group. We expect this standard to be adapted by many of the Modelica simulators, as it is developed by the Modelica Design Group. It will be simple and openly available. Therefore, we hope that other developers of simulation software will also export their results into this format.

We thank Martin Otter of DLR for the intensive discussions of this subject and Manuel Solis Lopez of BAUSCH-GALL GmbH for his programming efforts.

References

[1] Functional Mock-up Interface for Model Exchange, MODELISAR Document Version 1.0, January 26, 2010
www.functional-mockup-interface.org

[2] Modelica - A unified Object-Oriented Language for Physical Systems Modeling. Language Specification, Version 3.2, March 24, 2010

[3] www.modelica.org

[4] HDF5: www.hdfgroup.org

[5] ASAM-Standards: www.asam.net

[6] http://www.unidata.ucar.edu/software/netcdf/

[7] www.mathworks.com