Efficient Parameterization of Modelica Models

Thomas Beutlich[1] Dietmar Winkler[2]

[1]Germany, modelica@tbeu.de
[2]University of South-Eastern Norway, dietmar.winkler@usn.no

Abstract

This article presents the different approaches and use cases for efficient parameterization of Modelica models by means of external data resources. The main motivation is to improve the overall quality, testability and reusability of Modelica application models (both on component and system level) by a separation of the behavioral implementation from its actual design parameters. The Modelica libraries `ExternData` and `ModelicaTableAdditions` are freely available to support library developers and vendors in their ambitions to offer clean and dedicated interfaces for the parameterization of the application models and to benefit from a large variability of commonly used file types, such as CSV, Excel, HDF, JSON, MATLAB MAT, XML or even domain-specific file types such as for tire properties or weather data.

Keywords: parameterization, external data resources, Modelica external function interface, SSP

1 Introduction

The separation of the design parameters from Modelica application models was already discussed within the MA (Modelica Association)[1] about 15 years ago. Tiller (2005) developed an in-house library `DataRetrieval`, that featured a generic approach applicable for different file formats or data bases. Supported file formats included for example XML (eXtensible Markup Language), HDF (Hierarchical Data Format) and MATLAB MAT. There even have been early ideas for the standardization of the appropriate interfaces and XPath query expressions. Similarly, Köhler and Banerjee (2005) presented an in-house library `ZFlib` based on simple ASCII text files for a generic parameterization of transmission models. This library was later extended by Kellner et al. (2006) to also support target platforms without a file system. Reisenbichler et al. (2006) bewailed that the XML technology had not yet established as a standardized concept for the parameterization of Modelica application models. They again proposed to use XML as file format for external data resources – being a standardized and widely accepted language with significant tool support for data processing. Their Dymola[2] library also featured the full power of the XPath query expressions and data processing capabilities.

The topic was raised again for the MSL (Modelica Standard Library)[3] without greater reception in 2008[4]. Therein it was mentioned, that with the current concept of implementing the data access by the Modelica external function interface, the library vendors and users are responsible to instrument the Modelica code to consider parameterization (described as *pull*-principle). However, with a *push*-principle this responsibility could be moved to the tool vendors, and as such library users could benefit from a greater reusability and flexibility of the layered parameterization.

When the MA project SSP (System Structure and Parameterization of Components for Virtual System Design)[5] was initiated in 2014, only the parameterization of networks of FMUs (Functional Mock-up Units)[6] was considered. Even though the SSP standard 1.0 misses support for array parameters, it was not yet contemplated to apply it as layered standard for the parameterization of Modelica models.

With no standardized interface available, Modelica users depending on external data resources either still need to write their own utility libraries or have to depend on proprietary, tool-specific features (e.g., the data base interface of SimulationX[7]) or commercially available libraries such as `Modelon.DataAccess` from Modelon[8].

The parameterization of Modelica models can be differentiated by the following usage scenarios.

- Property parameters are constant during a transient simulation. They are non-structural parameters, i.e., a translated simulation model can be reused with changed parameters. Examples are geometry dimensions (e.g., tire diameter), material constants (e.g., electrical resistance) or ambient conditions (e.g., ambient pressure or gravitational acceleration). They can be of `Real`, `Integer`, `Boolean` or `String` type and either be scalar or of one/two-dimensional array kind (e.g., consumption map or road excitation map).

[1]Modelica Association, https://modelica.org
[2]Dassault Systèmes, https://www.3ds.com

[3]MA project "Libraries", https://doc.modelica.org
[4]MSL issue #115, https://github.com/modelica/ModelicaStandardLibrary/issues/115
[5]MA project "System Structure and Parameterization", https://ssp-standard.org
[6]MA project "Functional Mock-up Interface", https://fmi-standard.org
[7]SimulationX by ESI, https://www.simulationx.com
[8]Modelon, https://www.modelon.com

- Stimulation parameters can be considered as time-driven inputs for a transient simulation and can be modeled by one-dimensional look-up tables. Examples are the environmental conditions such as weather.
- Structural parameters have influence on the overall system topology and thus on the dimension of the system of equations. They need to be constant during a transient simulation, but any change requires a new translation of the Modelica model. Special care needs to be taken if structural parameters are read from external data resources.

The Modelica libraries `ExternData`[9] and `ModelicaTableAdditions`[10] are available as open-source Modelica packages under the permissive BSD-2-Clause License. Both libraries can be directly obtained from GitHub or via the Modelica `impact` package manager (Tiller and Winkler 2014; Tiller and Winkler 2015).

- `ExternData` supports the user in reading properties or structural parameters from various file types of external data resources. Data access from CSV (Comma Separated Values), INI, JSON (JavaScript Object Notation), MATLAB MAT (including HDF), SSV (System Structure Parameter Values), TIR (Tire Properties), Excel XLS/XLSX and XML files is implemented.
- `ModelicaTableAdditions` is an extension of the Modelica Standard Tables (Beutlich, Kurzbach, and Schnabel 2014) with support for more file types beside Dymola MOS[11] and MATLAB MAT. Its blocks can be utilized to also model stimulation parameterization or look-up tables from CSV, EPW (EnergyPlus Weather)[12] or JSON files and work as a drop-in replacement for the Modelica Standard Tables of the MSL.

There exists the use case to reuse the time series stored in the result data of one simulation as stimulation parameterization for subsequent simulations. Whereas Pfeiffer, Bausch-Gall, and Otter (2012) designed an HDF based file format, Tiller and Harman (2014) invented two new file formats for efficient read and write operations while particularly avoiding the HDF dependency. Both proposals only gained experimental acceptance within the MA. As there also was no adoption in the Modelica tool environment, the workaround is to store the simulation result data as CSV file, which then can be read again by the one-dimensional look-up table blocks of `ModelicaTableAdditions`.

[9]ExternData Git repository, `https://github.com/modelica-3rdparty/ExternData`

[10]ModelicaTableAdditions Git repository, `https://github.com/modelica-3rdparty/ModelicaTableAdditions`

[11]There is no specific name or file extension for the Dymola-specific text/script files starting with "#1" as first line.

[12]EnergyPlus Weather Data, `https://energyplus.net/weather`

2 ExternData

The Modelica library `ExternData` developed out of the need to offer an open-source utility package for efficient parameterization of property parameters from external data resources. It has been successfully tested in Dymola, OpenModelica[13] and SimulationX.

2.1 Library Design

The library (as shown in Figure 1) consists of top-level data source records for each supported file type, the provided accessor functions in `ExternData.Functions` and the external objects in `ExternData.Types`.

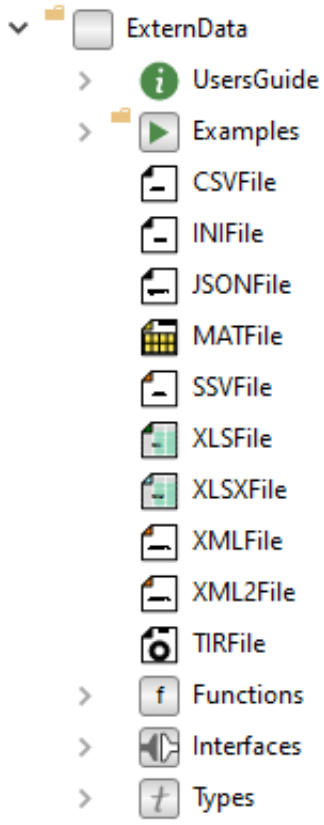

Figure 1. Library structure of `ExternData`

2.1.1 Data Source Records

The data source records are convenience types to encapsulate the external object component with its accessor functions.

A naïve record type definition together with an exemplary user call (application model) is given in Listing 1. (For the sake of clarity, the declaration of the external type `Types.ExtObj` and the external function annotation for `ExtFun` are skipped.)

[13]Open Source Modelica Consortium (OSMC), `https://openmodelica.org/`

DOI 10.3384/ecp21181141

Listing 1. Naïve record type definition

```
// Library
record DataSource "Data source record"
  parameter String fileName = ""
    "External data resource";
  final parameter Types.ExtObj obj =
    Types.ExtObj(fileName) "Ext. object";
  pure function get "Accessor function"
    input Types.ExtObj obj "Ext. object";
    input String s "Accessor id";
    output Real out "Data value";
    external "C" out = ExtFun(obj, s);
  end get;
end DataSource;

// Application model
parameter DataSource dataSource(
  fileName="dataSource.ext");
parameter Real p = dataSource.get(
  dataSource.obj, "id");
```

The disadvantage of such a naïve approach is that the handle of the external object, i.e. `dataSource.obj`, has to be passed by every call of the accessor functions despite it actually is an implementation detail of the record and could be a protected component[14].

Listing 2. Sophisticated record type definition

```
// Library
package Functions "Functions"
  pure function get "Accessor function"
    extends Interfaces.getBase;
    external "C" out = ExtFun(obj, s);
  end get;
end Functions;

package Interfaces "Interfaces"
  partial record DataSourceBase
    "Base data source record"
    replaceable function get = getBase;
  end DataSourceBase;
  partial function getBase "Base function"
    input Types.ExtObj obj "Ext. object";
    input String s "Accessor id";
    output Real out "Data value";
  end getBase;
end Interfaces;

record DataSource "Data source record"
  parameter String fileName = ""
    "External data resource";
  final parameter Types.ExtObj obj =
    Types.ExtObj(fileName) "Ext. object";
  extends Interfaces.DataSourceBase(
    redeclare final function get =
      Functions.get(obj=obj)
      "Accessor function");
end DataSource;

// Application model
parameter DataSource dataSource(
  fileName="dataSource.ext");
parameter Real p = dataSource.get("id");
```

[14]Only public sections are allowed for records, though.

A more sophisticated library design is based on clean interfaces for the records and accessor functions enabling inheritance, and thus the possibility of function redeclaration (Beutlich 2018). The general concept is presented in Listing 2.

With such a sophisticated library design the actual implementation (as an external object) is disguised from the caller as the handle of the external object no longer needs to be passed by the member accessor functions.

As of MLS 3.5 (Modelica Language Specification)[15], it is not yet fully specified, if external objects may be used in records[16].

2.1.2 External Functions

The actual external functions and objects serving the Modelica external function interface are implemented in C, i.e. no C++ is utilized.

Independent of the actual file type, the accessor functions for scalars of type `Real`, `Integer`, `Boolean` or `String` are named `getReal`, `getInteger`, `getBoolean` or `getString`, respectively. For one/two-dimensional arrays, the accessor functions are appended by `Array1D`/`Array2D`, e.g., `getRealArray1D` or `getIntegerArray2D`. There also are the corresponding functions to retrieve the array dimensions from the external data resource, i.e., `getArraySize1D` and `getArraySize2D` (and also `getArrayRows2D` and `getArrayColumns2D`).

2.1.3 Structural Parameters

Reading structural parameters from external data resources (as shown for an XML file by Listing 3) by functions `getArraySize1D` or `getArraySize2D` is not generally supported[17]. Of the tested Modelica tools, it only works in SimulationX.

Listing 3. Accessing structural parameters in SimulationX

```
// SimulationX application model
parameter String s = "vector"
  "XML element name";
parameter ExternData.XMLFile dataSource(
  fileName="dataSource.xml")
  "Data source record";
parameter Integer m =
  dataSource.getArraySize1D(s)
  "Structural parameter";
parameter Real p[:] =
  dataSource.getRealArray1D(s, m)
  "Array parameter";
```

To assist the Dymola users, alternative implementations using `readArraySize1D`/`readArraySize2D` functions are available. This comes with the disadvantage of redundant file I/O and is demonstrated by Listing 4.

[15]MA project "Libraries", https://specification.modelica.org/

[16]MLS issue #2399, https://github.com/modelica/ModelicaSpecification/issues/2399

[17]MLS issue #2425, https://github.com/modelica/ModelicaSpecification/issues/2425

Listing 4. Accessing structural parameters in Dymola

```
// Dymola application model
parameter String s = "vector"
  "XML element name";
parameter ExternData.XMLFile dataSource(
  fileName="dataSource.xml")
  "Data source record";
parameter Integer m =
  ExternData.XMLFile.Functions.
    readArraySize1D(
    varName=s,
    fileName="dataSource.xml")
  "Structural parameter";
parameter Real p[:] =
  dataSource.getRealArray1D(s, m)
  "Array parameter";
```

2.1.4 Missing Data

In some cases it may happen, that data of the external resources is missing, for example, an empty cell of an Excel file. By parameter `detectMissingData`, `ExternData` supports four options how to deal with missing data values.

- Return data-type specific defaults
- Return data-type specific defaults and print a message
- Return data-type specific defaults and raise a warning
- Stop the simulation with an error message

Similarly, the accessor functions (Section 2.1.2) also return a Boolean output `exist` to indicate if the retrieved data is available or missing. This way, the reaction on missing data can be modeled per function call.

2.2 Supported File Types

`ExternData` supports various file types for different kind of requirements.

2.2.1 CSV

CSV files contain exactly one data set that can be considered as matrix. An example file with three columns and a header line is given by Listing 5. Both the number of header lines to be ignored and the column delimiter character can be specified.

Listing 5. Example CSV file

```
x,y,z
0,0,0
0.5,0.25,0.125
1,2,3
```

2.2.2 INI

INI files contain scalar properties as key-value-pairs which are grouped by sections. The INI-keys can be fully qualified Modelica names using the dot notation. An example file with the default section and a named section is given by Listing 6.

Listing 6. Example INI file

```
# Default section
gain.k = 1
[Data set]
gain.k = 2
```

2.2.3 JSON

JSON files can be used to define scalars, vectors or matrices which can be arbitrarily structured. The JSON-keys must not contain the dot character to properly work with the accessor functions of `ExternData`. An example file with three different values is given by Listing 7.

Listing 7. Example JSON file

```
{
  "Data set": {
    "gain": {
      "k": "2"
    }
  },
  "vector": [1,2,3],
  "matrix": [[0,0],[0.5,0.25],[1,2]]
}
```

2.2.4 MATLAB MAT (including HDF)

MATLAB MAT files are binary files that can be used for scalars, vectors or matrices. Though it is a proprietary file format it is a common data exchange format for various scientific applications. MATLAB MAT of version 7.3 are HDF5 files and can be considered as a dedicated HDF5 data container. This format version is especially recommended for huge data volumes. However, it is not advisable to read huge arrays as Modelica variables. `ExternData` supports the access of nested structures using dot notation.

2.2.5 SSV

SSV files are standardized XML files that are used within the context of SSP to connect and parameterize FMUs. Certainly, they can not only be used to parameterize imported FMUs in the Modelica simulation environment. One SSV file describes exactly one parameter set where (as of version 1.0) only scalar parameter values are supported. Listing 8 displays an example SSV file.

Listing 8. Example SSV file

```
<?xml version="1.0" encoding="UTF-8"?>
<ssv:ParameterSet version="1.0" name="Data
  set" xmlns:ssv="http://ssp-standard.org/
  SSP1/SystemStructureParameterValues">
  <ssv:Parameters>
    <ssv:Parameter name="gain.k">
      <ssv:Real value="2"/>
    </ssv:Parameter>
  </ssv:Parameters>
</ssv:ParameterSet>
```

Technically, the external object `ExternData.Types.ExternXML2File` is reused while the SSV accessor functions call the appropriate

DOI 10.3384/ecp21181141

XML2 functions (`ExternData.Functions.XML2.*`) with dedicated XPath query expressions.

2.2.6 TIR

TIR files define domain-specific tire properties. They are similar to INI files and are implemented to share the same external object `ExternData.Types.ExternINIFile` with respective format permissions.

2.2.7 Excel XLS/XLSX

Both legacy XLS and Office Open XML based XLSX Excel files are supported for parameterization of scalars or matrices.

2.2.8 XML

There are two implementations available.

1. `ExternData.XMLFile` is a straightforward implementation to return values from XML element nodes

2. `ExternData.XML2File` enables full support of XPath query expressions to also query XML attributes or more complicated XML structures.

There is no restriction on the underlying XML schema, i.e. it can be used for arbitrarily structured XML data, being standardized (e.g., SSV or CPACS (Common Parametric Aircraft Configuration Schema)[18]) or customized.

3 ModelicaTableAdditions

The Modelica library `ModelicaTableAdditions` developed out of the need to offer reading of external data sources for stimulation parameters (e.g., look-up tables) from commonly used text file formats. Therefore, the blocks of `ModelicaTableAdditions` extend the Modelica Standard Tables of the MSL by support for additional file formats. It does not add other features like N-dimensional arrays, scattered data or spline approximation (as for example by Ungethüm and Hülsebusch (2009)). The Dymola MOS file format, which is the only text file format supported by the Modelica Standard Tables is not suitable for exchange between different applications. This is where CSV and JSON have their advantages, which can easily be processed by different applications. As with the Modelica Standard Tables, the external functions are implemented in pure C (i.e., no C++). For JSON, the same library dependencies are utilized as with `ExternData`. It is possible to use components of packages `ExternData`, `ModelicaTableAdditions` and MSL in the same application model. The library has been successfully tested in Dymola and OpenModelica (Linux only).

3.1 Library Design

The library (as shown in Figure 2) consists of the five look-up table blocks known from the MSL, but this time within the `ModelicaTableAdditions` name-space.

[18]CPACS, `https://cpacs.de`

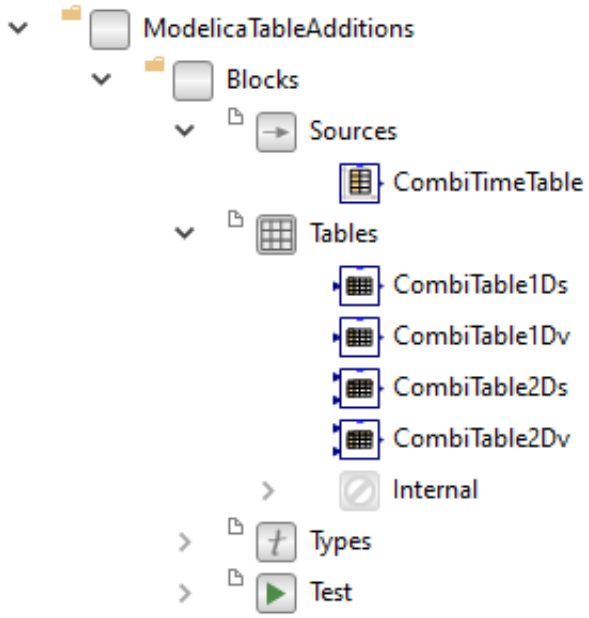

Figure 2. Library structure of `ModelicaTableAdditions`

3.2 Supported File Types

3.2.1 CSV

The table of a CSV file (as shown by Listing 5) can be used for time-driven simulation or one/two-dimensional look-up tables. Both the number of header lines to be ignored and the column delimiter character can be specified. Addressing certain columns by their names in the (optional) CSV header line is not supported.

Support for CSV files was also ported to the Modelica Standard Tables and merged to the MSL master branch in early 2021[19].

3.2.2 EPW

The weather conditions of one year are the typical stimulation parameters for building energy simulations, such as the Modelica `Buildings` Library (Wetter et al. 2014). EnergyPlus provides weather data for simulation in various formats, especially their EPW format. Since this file format is not natively supported by the Modelica Standard Tables it manually needed to be pre-processed and converted to either Dymola MOS or MATLAB MAT format.

This pre-processing no longer is necessary as the EPW format is directly supported by all one-dimensional look-up table blocks of the `ModelicaTableAdditions` library.

3.2.3 JSON

Similar as with CSV, tables can be read from JSON files and be utilized as stimulation parameters, e.g. parameter "matrix" of Listing 7.

[19]MSL issue #3691, `https://github.com/modelica/ModelicaStandardLibrary/pull/3691`

4 Conclusions and Outlook

The open-source libraries `ExternData` and `ModelicaTableAdditions` are an offer to Modelica library developers and users to efficiently parameterize Modelica application models. Its right to exist is due to a missing layered (MA) standard for the parameterization of Modelica models. As already mentioned by Tiller (2005), the benefits of such a standardization are the cutback of library code towards the Modelica tools to even further increase the efficiency, convenience and usability of the parameterization of application models.

There is no support for units so far, i.e., unit conversion is left to the application. This could be also addressed by a standardized parameterization.

Library-wise, one future idea is the support of (symmetrical or asymmetrical) encrypted external resources, which is not yet covered by the MLS. In such cases the external functions require the appropriate (private) key to decrypt the external resources at simulation run-time in memory. Again, encryption as an isolated application can only be considered a short-term solution towards a future standard.

Furthermore, it might be desirable if the mentioned open issues on the MLS regarding external objects could be clarified finally.

Acknowledgements

The authors would like to thank everybody who has contributed to the libraries `ExternData` and `ModelicaTableAdditions`, particularly, Adrian Pop, Hang Yu, Martin Sjölund, Mike Dempsey and Peter Harman.

References

Beutlich, Thomas (2018-08). *Modeling Hints for Modelica External Interfaces*. Presentation given at the 19th Modelisax Meeting, Dresden, Germany. URL: https://tinyurl.com/Modelisax2018Hints.

Beutlich, Thomas, Gerd Kurzbach, and Uwe Schnabel (2014-03). "Remarks on the Implementation of the Modelica Standard Tables". In: *Proceedings of the 10th International Modelica Conference*. Ed. by Hubertus Tummescheit and Karl-Erik Årzén. Lund, Sweden, pp. 893–897. DOI: 10.3384/ecp14096893.

Kellner, Matthias et al. (2006-09). "Parametrization of Modelica Models on PC and Real time platforms". In: *Proceedings of the 5th International Modelica Conference*. Ed. by Christian Kral and Anton Haumer. Vienna, Austria, pp. 267–273. URL: https://www.modelica.org/events/modelica2006/Proceedings/sessions/Session3b2.pdf.

Köhler, Jochen and Alexander Banerjee (2005-03). "Usage of Modelica for transmission simulation in ZF". In: *Proceedings of the 4th International Modelica Conference*. Ed. by Gerhard Schmitz. Hamburg, Germany, pp. 587–592. URL: https://modelica.org/events/Conference2005/online_proceedings/Session7/Session7c1.pdf.

Pfeiffer, Andreas, Ingrid Bausch-Gall, and Martin Otter (2012-09). "Proposal for a Standard Time Series File Format in HDF5". In: *Proceedings of the 9th International Modelica Conference*. Ed. by Martin Otter and Dirk Zimmer. Munich, Germany, pp. 495–506. DOI: 10.3384/ecp12076495.

Reisenbichler, Ulf et al. (2006-09). "If we only had used XML..." In: *Proceedings of the 5th International Modelica Conference*. Ed. by Christian Kral and Anton Haumer. Vienna, Austria, pp. 707–716. URL: https://www.modelica.org/events/modelica2006/Proceedings/sessions/Session6d1.pdf.

Tiller, Michael (2005-03). "Implementation of a Generic Data Retrieval API for Modelica". In: *Proceedings of the 4th International Modelica Conference*. Ed. by Gerhard Schmitz. Hamburg, Germany, pp. 593–602. URL: https://modelica.org/events/Conference2005/online_proceedings/Session7/Session7c2.pdf.

Tiller, Michael and Peter Harman (2014-03). "recon – Web and network friendly simulation data formats". In: *Proceedings of the 10th International Modelica Conference*. Ed. by Hubertus Tummescheit and Karl-Erik Årzén. Lund, Sweden, pp. 1081–1093. DOI: 10.3384/ecp140961081.

Tiller, Michael and Dietmar Winkler (2014-03). "impact - A Modelica Package Manager". In: *Proceedings of the 10th International Modelica Conference*. Ed. by Hubertus Tummescheit and Karl-Erik Årzén. Lund, Sweden, pp. 543–548. DOI: 10.3384/ecp14096543.

Tiller, Michael and Dietmar Winkler (2015-09). "Where impact got going". In: *Proceedings of the 11th International Modelica Conference*. Ed. by Peter Fritzson and Hilding Elmqvist. Versailles, France, pp. 725–736. DOI: 10.3384/ecp15118725.

Ungethüm, Jörg and Dirk Hülsebusch (2009-09). "Implementation of a Modelica Library for Smooth Spline Approximation". In: *Proceedings of the 7th International Modelica Conference*. Ed. by Francesco Casella. Como, Italy, pp. 669–675. DOI: 10.3384/ecp09430013.

Wetter, Michael et al. (2014-07). "Modelica Buildings library". In: *Journal of Building Performance Simulation* 7.4, pp. 253–270. DOI: 10.1080/19401493.2013.765506.