DAE Tools Software

Introduction

D.D. Nikolić

Updated: 1 April 2016

DAE Tools Project, http://www.daetools.com

Outline

1. General Information

2. Motivation

3. Programming Paradigms

4. Architecture

5. Developing models with DAE Tools

6. Use Cases

General Information

What is DAE Tools?

Process MODELLING, SIMULATION, and OPTIMISATION software

- Areas of application:
  - Initially: chemical process industry (mass, heat and momentum transfers, chemical reactions, separation processes, thermodynamics, electro-chemistry)
  - Nowadays: MULTI-DOMAIN
- Free/Open source software (GNU GPL)
- CROSS-PLATFORM (GNU/Linux, MacOS, Windows)
- MULTIPLE ARCHITECTURES (32/64 bit x86, ARM, ...)

What is DAE Tools? (cont'd)

- DAE Tools IS NOT:
  - A modelling language (such as Modelica, gPROMS, ...)
  - An integrated software suite of data structures and routines for scientific applications (such as PETSc, Sundials, ...)
- DAE Tools IS:
  - A HYBRID approach between modelling and general-purpose programming languages
  - A higher level structure – an architectural design of interdependent software components providing an API for:
    - Model development/specification
    - Activities on developed models (simulation, optimisation, ...)
    - Processing of the results
    - Report generation
    - Code generation and model exchange

What can be done with DAE Tools?

- Simulation
  - Steady-State
  - Transient
- Optimisation
  - Non-Linear Programming (NLP) problems
  - Mixed Integer Non-Linear Programming (NLP) problems
- Parameter estimation
  - Levenberg–Marquardt algorithm
- Code-generation, model-exchange, co-simulation
  - Modelica, gPROMS, Matlab, Simulink
  - Functional Mockup Interface (FMI)
  - C99 (for embedded systems)
  - C++ MPI (for distributed computing)

Types of systems that can be modelled

INITIAL VALUE PROBLEMS OF IMPLICIT FORM, (described by systems of linear, non-linear, and (partial-)differential algebraic equations).

- CONTINUOUS with some elements of EVENT-DRIVEN systems (discontinuous equations, state transition networks and discrete events)
- STEADY-STATE or DYNAMIC
- With LUMPED or DISTRIBUTED parameters (finite difference, finite volume and finite element methods)
- Only INDEX-1 DAE systems at the moment

Motivation

Why modelling software?

In general, two scenarios:

- Development of a new product/process/...
  - Reduce the time to market (TTM)
  - Reduce the development costs (no physical prototypes)
  - Maximise the performance, yield, productivity, purity, ...
  - Minimise the capital and operating costs
  - Explore the new design options in less time and no risks
- Optimisation of an existing product/process/...
  - Increase the performance, yield, productivity, purity, ...
  - Reduce the operating costs, energy consumption, ...
  - Debottleneck

Why YET ANOTHER modelling software?

Currently available options:

1. MODELLING LANGUAGES (domain-specific or multi-domain) (Modelica , Ascend , gPROMS , GAMS , Dymola , APMonitor)
2. GENERAL-PURPOSE PROGRAMMING LANGUAGES:
   - Lower level third-generation languages such as C, C++ and Fortran (PETSc , SUNDIALS)
   - Higher level fourth-generation languages such as Python (NumPy, SciPy, Assimulo), Julia etc.
   - Multi-paradigm numerical languages (Matlab , Mathematica , Maple , Scilab , and GNU Octave)

Why YET ANOTHER modelling software? (cont'd)

The advantages of the HYBRID approach over the MODELLING and GENERAL-PURPOSE programming languages:

1. Support for the RUNTIME MODEL GENERATION
2. Support for the RUNTIME SIMULATION SET-UP
3. Support for COMPLEX RUNTIME OPERATING PROCEDURES
4. INTEROPERABILITY with the THIRD-PARTY SOFTWARE packages (i.e. NumPy/SciPy)
5. Suitability for EMBEDDING and use as a WEB APPLICATION or SOFTWARE AS A SERVICE
6. CODE-GENERATION, MODEL EXCHANGE and CO-SIMULATION capabilities

Additional features

- Support for the AUTOMATIC DIFFERENTIATION (ADOL-C)
- Support for the SENSITIVITY ANALYSIS through the auto-differentiation capabilities
- Support for the PARALLEL computation (OpenMP, GPGPU, MPI)
- INTEROPERABILITY with the 3^{rd} party numerical software (NumPy, SciPy, ...)
- Support for a large number of DAE, LA and NLP solvers
- Support for the generation of MODEL REPORTS (XML + MathML, Latex)
- EXPORT of the SIMULATION RESULTS to various file formats (Matlab, Excel, json, xml, HDF5, Pandas)

Programming Paradigms

The hybrid approach

Object-oriented modelling

- Everything is an object (models, parameters, variables, equations, state transition networks, simulations, solvers, ...)
- Models are classes derived from the base daeModel class (inheriting the common functionality)
- Hierarchical model decomposition allows creation of complex, re-usable model definitions
- All Object Oriented concepts supported (such as multiple inheritance, templates, polymorphism, ...) that are supported by the target language (c++, Python), except:
  - Derived classes always inherit all declared objects (parameters, variables, equations, ...)
  - All parameters, variables, equations etc. remain public

Equation-oriented (acausal) modelling

- Equations given in an implicit form (as a residual)

$$F(\dot{x}, x, y, p) = 0$$

- Input-Output causality is not fixed:
    - Increased model re-use
    - Support for different simulation scenarios (based on a single model) by specifying different degrees of freedom
- For instance, equation given in the following form:

$$x_1 + x_2 + x_3 = 0$$

can be used to determine either x_1, x_2 or x_3 depending on what combination of variables is known:

$$x_1 = -x_2 - x_3 \ or \ x_2 = -x_1 - x_3 \ or \ x_3 = -x_1 - x_2$$

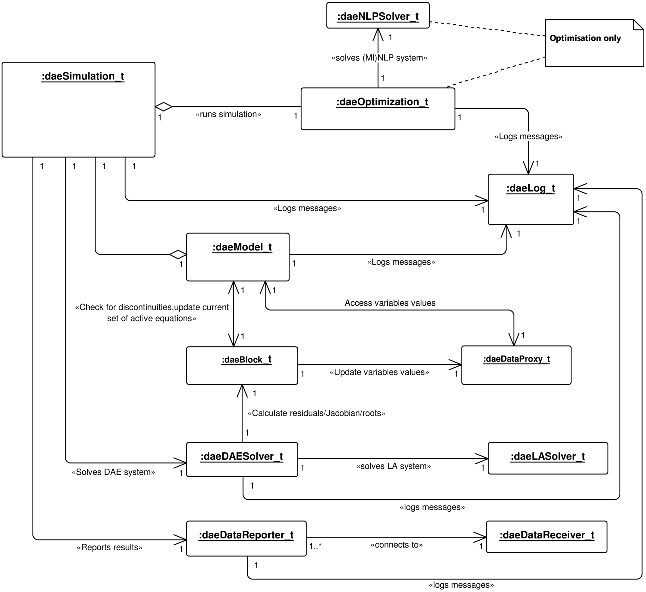
Separation of model definition from activities on models

- The structure of the model (parameters, variables, equations etc.) given in the model classes (*daeModel*, *daeFiniteElementModel*)
- The runtime information in the simulation class (*daeSimulation*)
- Single model definition, but:
  - One or more different simulation scenarios
  - One or more optimization scenarios

Architecture

The fundamental concepts/software interfaces

- Concepts/Interfaces:
  - daeModel_t
  - daeSimulation_t
  - daeOptimization_t
  - daeBlock_t
  - daeDAESolver_t
  - daeLASolver_t
  - daeDataReporter_t
  - daeBlock_t
- In 6 packages:
  - CORE
  - ACTIVITY
  - DATAREPORTING
  - SOLVERS
  - LOGGING
  - UNITS



Package CORE

The key modelling concepts in the CORE package.

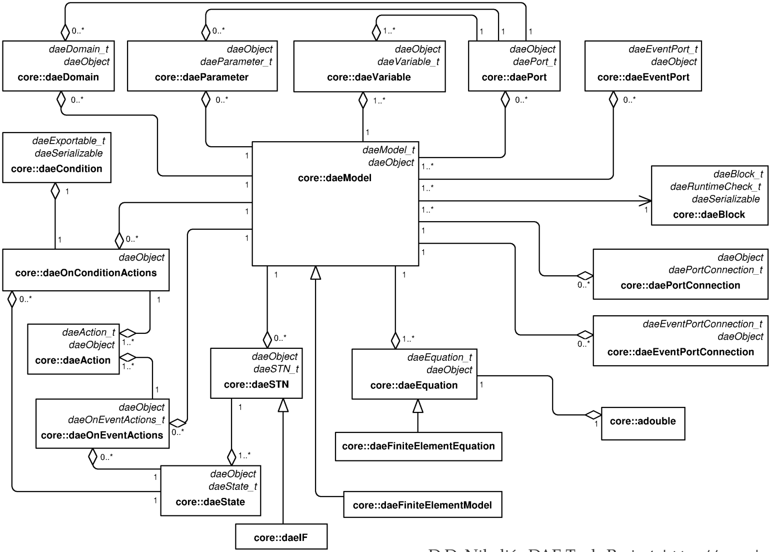
Concept	Description
daeVariableType_t	Defines a variable type that has the units, lower and upper bounds, a default value and an absolute tolerance
daeDomain_t	Defines ordinary arrays or spatial distributions such as structured and unstructured grids
daeParameter_t	Defines time invariant quantities that do not change during a simulation
daeVariable_t	Defines time varying quantities that change during a simulation
daePort_t	Defines connection points between model instances for exchange of continuous quantities
daeEventPort_t	Defines connection points between model instances for exchange of discrete messages/events

Package CORE (cont'd)

The key modelling concepts in the CORE package (cont'd).

Concept	Description
daePortConnection_t	Defines connections between two ports
daeEventPortConnection_t	Defines connections between two event ports
daeEquation_t	Defines model equations given in an implicit/acausal form
daeSTN_t	Defines state transition networks used to model discontinuous equations
daeOnConditionActions_t	Defines actions to be performed when a specified condition is satisfied
daeOnEventActions_t	Defines actions to be performed when an event is triggered on the specified event port
daeState_t	Defines a state in a state transition network
daeModel_t	Represents a model

Package CORE - interface implementations



Package ACTIVITY

The key concepts in the ACTIVITY package.

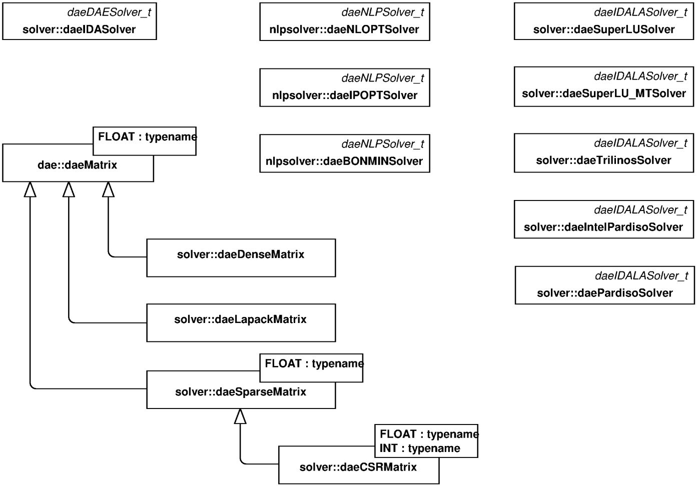
Concept	Description
daeSimulation_t *daeOptimisation_t*	Defines ...

Package SOLVERS

The key concepts in the SOLVERS package.

Concept	Description
daeDAESolver_t *daeLASolver_t* *daeNLPSolver_t* *daeIDALASolver_t*	Defines ...
daeMatrix_t<typename FLOAT>	

Package SOLVERS - interface implementations

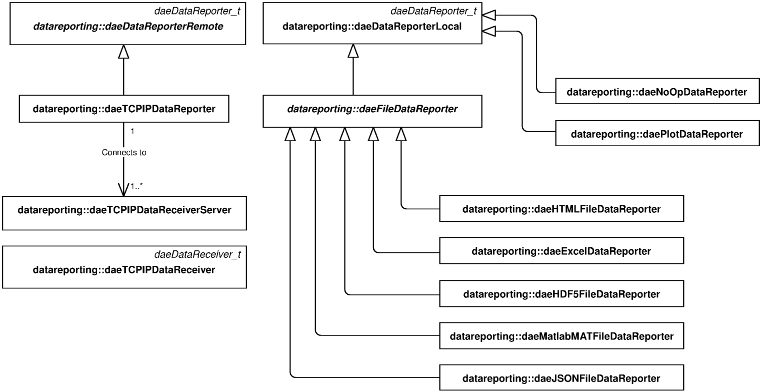


Package DATAREPORTING

The key concepts in the DATAREPORTING package.

Concept	Description
daeDataReporter_t *daeDataReceiver_t*	Defines ...

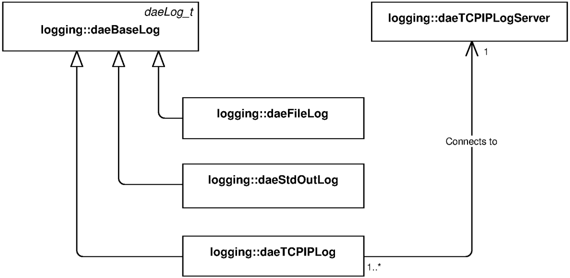
Package DATAREPORTING - interface implementations



Package LOG and its interface implementations

The key concepts in the LOG package.

Concept	Description
daeLog_t	Defines ...



Package UNITS

The key concepts in the UNITS package.

Concept	Description
unit	Defines ...
quantity	

Developing models with DAE Tools

Overview

Use Cases

Use Case 1 - High-Level Modelling Language

Use Case 2 - Low-Level DAE Solver

Use Case 3 - Embedded Simulator (back end)

Use Case 4 - Web Application / Web Service