The ANDECS CACE Framework

G. Grübel

Computer-aided control engineering (CACE) environments, in order to be reusable in a broad spectrum of applications, have to be based on an open software framework that provides seven classes of services: database services; model-definition services; algorithmic services; tool-control services; task-control services; user-interaction services; and process-communication services. This is detailed herein with the available CACE framework for the application system ANDECS.

Introduction

Presently we face the transition from CACSD packages, notably Toolboxes, to integrated tool systems based on open software frameworks. While CACSD packages provide collections of tools for the computational execution of control and system theory, CACE (computer-aided control engineering) frameworks allow the integration of such CACSD tools with more general modeling, simulation, and optimization tools.

In the present discussion on frameworks it is helpful to refer to similar developments in the domain of CASE (computer-aided software engineering). This is done in [1] by proposing a CACE services "framework reference model" in analogy to the CASE-framework model of the European Computer Manufacturer's Association. It may also be helpful to substantiate and to concretize the demands and abstractions of [1] by describing a realization that has already proven its value in daily use. Such a realization is the CACE-framework of ANDECS. ANDECS [2, 3] is an integrated tool system for "ANalysis & DEsign of Controlled Systems" whose basic architecture is an open framework which supports seven classes of services:

- engineering-database services
- model/data definition services
- algorithmic services
- tool-control services
- task-control services
- user-interaction services
- process-communication services

These services are dealt with in the following sections of this paper.

Application-neutral, reusable services form a frame to configure specific application packages by "plugging in" specific tools, and to develop such specific tools based on a repository of algorithmic and data-handling software components. That means, these services provide the necessary infrastructure for "bottom-up" evolutionary software engineering for CACE applications. They support flexible-to-use tool systems where the application functionality is embodied in a set of individual function modules called tools; each one of these tools to be operated stand-alone or within tasks of computation chains and loops. Application flexibility is achieved via a monitor-controlled task configuration mechanism. Strict tool modularity is guaranteed in that all tools are allowed to communicate their input/output data "in public only," namely via Abstract Data Types which are handled in a database. The task configuration mechanism with its parsing facility is also used as a "supervisor" system for integrating proprietary engineering software packages by coherently interfacing package commands and package data. An example is the way of integrating [5]MATLAB Toolboxes in ANDECS. Besides increasing the flexibility and engineering efficiency in configuring application systems, in further software developments and functional extensions, a modular framework with clear-cut decomposition interfaces and composition mechanisms also serves as integrator on syntactic, semantic, or method level [6].

Frameworks are domain-specific "Engineering Operating Systems" which help to translate engineering tasks to computation tasks. They sit on top of a computer operating system, which then translates computation tasks to the data processing tasks of a computer. The ANDECS CACE framework supports the engineering domain of Computational System Dynamics. It relies on the Engineering Operating System RSYST [7], which is completely domain-neutral and so far has proven its basic value in such diverse application domains as thermal analysis engineering, nuclear power engineering, environmental engineering and (bio-)mechanical multibody dynamics engineering, besides control system engineering by ANDECS.

Engineering Database

A database or a data repository is necessary for consistent data sharing in tool integration on the syntactic level [6]. It allows a formally controlled data access and by that a monitored data flow between different tools. In this function it can be the physical layer for an "engineering databas" which integrates a tool-based system (Fig. 1). Besides the short transactions in controlling the data flow in computation chains, a database also serves the more traditional purpose of long-term archiving of model- and computation-experimentation data.

The engineering operating system RSYST permits up to eight databases within one application. This allows nested transactions for specific subtasks, and transaction grouping for design objects which go through several transient phases before a stable data state is reached. Each database is hierarchically structured and provides a class schema with inheritance to formally specify explicit object types, which in turn can be composed of the

---

A version of this paper appeared in the Proceedings of the IEEE/IFAC CADCS'94 Symposium, Tucson, AZ, 1994. The author is with the Institute of Robotics and System Dynamics, DLR—German Aerospace Research Establishment, Oberpfaffenhofen, D-82234 Wessling, email: df12@master.df.op.dlr.de.

IEEE Control Systems
following implicit object types: one-, two-, three-dimensional arrays of integers, real- and double-precision numbers; parameter sets of numeric and symbolic values; text; and pixel-pictures. Data objects are automatically tagged with a timestamp and an identifier indicating the module by which they are created or modified. Simple versioning is possible by a numerical index. For high data processing performance, the implicit object types are recognized by the database as Abstract Data Types. Furthermore, the RSYST database is designed to work efficiently with large data objects, e.g., numeric arrays of time series, and can be kept resident in main memory for very fast access, e.g., when used as a database within an optimization loop.

Services are command operations embedded in an X-Windows user-interaction environment. It provides as utilities for implicit object types: initialize and delete, browse and name within tree data structures, read and write elements, and specify logical links between objects in different tree branches. It provides as utilities for explicit object types: generate new object types, modify, browse data hierarchy and data items, read and write object components, and delete.

Note that a tool-based software architecture using an "engineering database" is the only means to keep the interface complexity problem of functionally large application systems manageable, because here the number of interfaces increases only linearly with the number n of function tools. Otherwise the complexity growth is of order $O(n^2)$. The Engineering Operating System RSYST has been developed under these premises since 1969 to compose large software systems in a structured and data-processing-efficient way.

Model/Data Definition

Information systems are sometimes viewed as being structured into three layers: an application layer, a logical layer, and a physical layer. The dynamic systems model-definition services of our framework correspond to the logical data layer, while the database services then correspond to a "physical" layer. The model-definition services support a model/data integration on the semantic level [6]. Services are translators to and constructors for neutral (i.e., application- and realization-independent) formally defined data structures that can be used both for system-internal module input/output data communication and for data import and export with external systems.

CDO = Control Data Objects

Control-dynamics methods use and generate data such as time and frequency signals, transfer functions, and linear system descriptions in state-space form. These generic data structures have been formally defined as "Control Data Objects" (CDO) [8], in conformance with the standards discussed in the German VDI/GMR Committee on Control Software-Tools. In order to store and access CDOs on a database, they are defined as Abstract Data Types or they are formally defined as data objects via the database class schema. Also, for a system dynamics data sharing among different packages, corresponding definitions in the ISO STEPExpress standard information-communication language have been explored [9]. Graphical editors provide a most transparent user interface to these complex data structures. So far, graphical editors are available as portable Motif-widgets for the CDO "signal" and for system matrices [10]. Portable graphical editors for other CDOs are in development using the already available ones as basic components.

DSblock = Dynamic System Block

DSblock is a standardized computer-readable protocol for interfacing generic numeric methods (time integration, steady-state computation by solving nonlinear equations, linearization, etc.) with nonlinear system models described by parameterized, variable structure, time-, state-, and step-event dependent, ordinary differential- or differential-algebraic equations. This is a neutral computational model format, with the detailed protocols and utilities [11] being in international discussion for a possible standardization agreement.

DSblock serves for computational models integration on the syntactic level as far as models are concerned which originate in different engineering domains and in different modeling environments [12] (Fig. 2). From the view of numerical algorithms, it serves for model integration on a semantic level.

Automatic code generation of DSblock models is supported by two facilities:

- There is an external translator of the CSSL-syntax and the simulation-model definition functionality of ACSL level 10, to DSblock code [12]. ACSL allows dynamics model integration on the syntactic level and partly on the semantic level.
- The object-oriented modeling environment Dymola [13] directly generates DSblock code from given model specifications based on first physical principles or bond-graph methodology. Dymola allows dynamics model integration on the semantic and method level [6] in using physical-

Fig. 1. Tool integration by a monitored data flow via an "Engineering Data Bus": the corresponding functional view is shown above.
2-D plots and specific control engineering plots like Bode, nonlinear event-driven ordinary differential, and differential-algebraic equations. The library provides routines pertinent to control and systems engineering analysis, integration on the method level.

Numerical Algorithms

In engineering, numerical-, mathematical-, and rule-based algorithms play as important a role as data handling. Our CACE-framework provides generic numerical support via the control-engineering subroutines library RASP [15], and via MATLAB functions for command-controlled matrix-algebra calculations.

RASP = Regelungstechnische Analyse und Synthese Programme

RASP provides the services of a software-engineered numerical-subroutines library. This is more than a mere collection of programs: The common "look and feel" of the library serves to integrate on the syntactic level various quality-software items collected from different professional sources. The modularity of the library provides integration on the semantic level. A common errors-exception handling mechanism supports numerics- software integration on the method level. The RASP chapters of contents are: CONTROL, i.e., state and frequency domain subroutines pertinent to control and systems engineering analysis, synthesis and model-order reduction; GRAPHICS, for general 2-D plots and specific control engineering plots like Bode, Nyquist, root-locus, and time-response plots; OPTIMIZE, for nonlinear mathematical programming, SIMULATE, for solving nonlinear event-driven ordinary differential, and differential-algebraic equations; and MATH, for complex analysis and solving matrix equations. RASP makes use of BLAS, EISPACK, LINPACK, and in its present and future development, LAPACK. RASP now constitutes a major part in a European effort for a product-quality public-domain numerics library for control engineering and systems theory [16].

Matlab = ANDECS-MATLAB

The MATLAB-syntax is a de facto standard in matrix-based linear systems—and control theory. Therefore a CACE framework must assist the use of this syntax. Matlab is a QPARSER+ based re-implementation of this syntax, in order to generate, read, and write matrices on a database, to deal with CDOs, and to define matrix calculations by line commands [5]. The scope of available basic functions is comparable to MATLAB, the underlying numerics is that of RASP. By the MATLAB syntax, basic matrix-algebra computations are integrated on the semantic level. (MATLAB M-files are integrators on the method level.) Integrating symbolic-math operations by using MAPLE as a tool is under development. To support a basic knowledge base system functionality, MEDAL [17] is coupled and applied in the same way as a MATLAB Toolbox. In a present application this supports semantic consistency checking in visual programming.

Function-Module/Tool Control

The application functionality of ANDECS is embodied in its application tools. These tools are either generic function modules based on the analysis and synthesis numerics of RASP, or external tool packages like MATLAB whose functionality can be used within ANDECS applications in exactly the same form as generic function modules.

Framework services are provided to build and to control function modules. This supports tool integration on the semantic level.

- To build function modules. A function module is an encapsulated computation entity with data inputs and outputs only, which is integrated in its surrounding environment by standardized, monitor controlled interfaces. Module inputs and outputs are Abstract Data Types or formally defined data objects on the database. RSYST provides "wrapper" subroutines to build such a function module by, e.g., encapsulating the computational source code through RSYST- Initialization and Exit subroutines, and by replacing Fortran READ and WRITE by database- and dialog-system subroutine calls.
- To control function-modules. RSYST allows the operation of modules in both interactive and batch mode by means of a homogeneous command, dialog, and help system. In particular, parameter sets and command values can be input both in string command and menu form within an X-Windows user-interaction environment. Inputs are automatically checked for syntactic errors, and module-help text in short and long versions can be sought.
- To use MATLAB as function-modules. A communication module (Fig. 3) uses a combination of UNIX command

![Fig. 2. The simulation-model bus DSblock as a conceptual separator to interface different simulation-modeling environments with different run-time environments such as DSSIM for simulation and DSLIN for linearization, etc.](image)

![Fig. 3. The ANDECS module PROMAT provides a process interaction with MATLAB Toolboxes in such a way that they can be used in computation chains together with generic ANDECS modules. The same kind of process communication link is used with the knowledge base system MEDAL.](image)
Fig. 4. The ANDECS-MOPS frame for modular multicriteria design compromising.

pipes and file transfer to link MATLAB as a slave process. This is done in such a manner that MATLAB Toolboxes can be called by ANDECS and used in the same way as generic ANDECS function modules. For more details and for an application we refer to [5, 18]. Since MATLAB is a software standard for implementing control and system theory, this is a rich source for enhancing the tool set of an application system.

**Application Task Control**

Application flexibility in general is guaranteed by the combination of two basic schemes: the database module communication, where all modules communicate their input/output data via database only and no direct module-to-module data communication is allowed, and the macro facility of the monitor system. This allows setup of strictly modular computation sequences of function modules with a control logic including "while" and "if-then." Computation chains and loops can be operated interactively. Or macro scripts can be stored on database and operated in batch mode in combination with automatic sequential parameter iterations. In particular, there is a design-automation frame MOPS which uses this modularization and the macro-monitor and database services.

MOPS = Multi-Objective Programming System

MOPS [19] is a "macro-module" for computational design experimenting on three strata: (I) model/synthesis experiments, (J) analysis experiments, and (K) parametric compromising experiments. This "macro-module" controls a computation loop where different application function modules (tools) of the following method classes are called in an ordered way [20] (Fig. 4): (parameter-equation-) synthesis P(T), model evaluation M(P), analysis and simulation "indicator" computation I(M), performance criteria evaluation C(I), and variation of tuning parameters T(C). In particular, the tuning parameters can be "pareto-optimized" by a multi-criteria goal attainment system using mathematical programming. In addition, MOPS generates an automatically evolving experiments-project database with a hierarchy corresponding to the experimentation strata I, J, K (Fig. 5). This allows a directed navigation through a design-experimentation history, as well as backtracking, branching, and restart of experiments. MOPS serves for integration of computational experiments on the syntactic, semantic, and method levels. As application examples we refer to [18, 21].

Note that the MOPS software frame itself is completely application-neutral. It is open to get instantiated by any appropriate application-domain specific models and synthesis and analysis tools. For instance, we have used it in a multicriteria parameter study for optimizing the material layers of heat shields for reentry space vehicles.

**User Interaction**

A group of modules called VISTA [22], i.e. "Visualization and Interactive Steering for Task Activation," is provided for the setup and on-line control of application/user interactions (Fig. 6). Just three application-independent tools, VISMAN, VISSTEER, and VISEVENT, allow setup and interactive control of visualization in multiple diagrams in multiple windows, and steering of on-line parameter variations through multiple sliders or by using a six-dimensional steering ball. The user interface of VISTA is...
based on X-Windows Motif. VISTA provides integration on the syntactic, semantic, and method levels for all interactive tasks of application/user interaction.

Under development is a facility for "visual programming" of computation chains and loops. Here the function modules, called CMblocks (Computational Module blocks) are displayed as cmBLOCK Motif-Widgets. Their CDO data inputs and outputs are graphically connected in their causal functional order, while an expert system automatically checks the syntactic and semantic compatibility of the corresponding input/output data structures. This will allow non-experts an engineering-efficient setup, modification, and instrumentation of application-specific computation sequences, as it is required in modular computational experimentation [20]. In particular, this can be combined with an automatic scheduling for parallel computing, using the PVM (Parallel Virtual Machine) software. For more details we refer to [23].

Process Communication

To distribute the computation workload on different computers in a heterogeneous computer network or to share proprietary software only available on another computer, we routinely use the PVM (Parallel Virtual Machine) process communication and scheduling software, or we use suited UNIX services. As far as the different framework services are concerned, the obvious trend goes to a software realization by several independent processes. The present release (3.5) of the core system RSYS'T has no explicit message services system by its own. But the next release (4.0) will support a general network messaging system for client/server distributed computations. To be as portable as possible, the implementation, which is presently in testing, will be based on the Client Message Events of the X-Window system. Higher-level services, such as lock server and database manager, then are built on this messaging system. Database, interactive monitor command, and help services then can be realized as independent processes.

Summary and Conclusion

The CACE framework we have described is an example of an already realized, coherent "Control-Engineering Operating System." It links CACE tasks to computational tasks of informatics and numerics. This is on top of the UNIX Computer Operating System, which links computation tasks to the data processing of the computer. It serves both for ANDECS tool integration and as a supervisor system for "external" package integration. For example: object-oriented modeling using Dymola, system simulation and linearization using the ANDECS modules DSSIM and DSLIN, linear analysis and synthesis using any appropriate MATLAB Toolbox, and multi-modal result visualization using ANDECS_VISTA, is easily achieved.

A flexible-to-use design-automation frame is provided by ANDECS_MOPS (Multi-Objective Programming System), together with the set of user-interaction modules of ANDECS_VISTA (Visualization and Interactive Steering for Task Activation), this allows a "soft" configuration of application-specific synthesis/analysis loops to find best-possible compromise solutions in a multi-criteria problem setup, which is the most natural way to deal with dynamically complex control design tasks.

Applications of ANDECS at DLR are in robotics, in aerospace, and in ground-vehicle system dynamics. These include trajectory optimization and control of kinematically redundant manipulators and servicing robots in space, active pointing of large space telescopes, flying-quality control of high-performance aircraft, helicopter higher-harmonic rotor blade control, and optimization of active car suspensions and semi-active aircraft landing gear.

References


Georg Griebel received the Diplom-Ingenieur degree from the Technical University of Munich, Germany, a master of science degree from Cornell University, Ithaca, NY, and the Dr.-Ing. degree in electrical engineering from Ruhr-Universität Bochum, Germany. In 1977 he received the venia legendi for control engineering in the Department of Mechanical Engineering at Ruhr-Universität Bochum, where he has been an adjunct professor since then. Since 1978, he has been head of the Control Design Engineering Group in the Institute for Robotics and System Dynamics at DLR-German Aerospace Research Establishment, Oberpfaffenhofen, where he concentrates research and development on computer-aided control engineering for analysis and design of dynamically complex aerospace and robotic systems.