GARTEUR Open

DETERMINING REQUIREMENTS FOR A
COMPUTATIONAL AIRCRAFT ENGINEERING ENVIRONMENT (CACEE)
BASED ON A LITERATURE SURVEY

by

J.B.R.M. Spee

GARTEUR aims at stimulating and co-ordinating
co-operation between Research Establishments and Industry
in the areas of Aerodynamics, Flight Mechanics, Helicopters,
Structures & Materials and Propulsion Technology
DETERMINING REQUIREMENTS FOR A COMPUTATIONAL AIRCRAFT ENGINEERING ENVIRONMENT (CACEE) BASED ON A LITERATURE SURVEY

by

J.B.R.M. Spee

This report has been prepared under auspices of the Responsables for Flight Mechanics, Systems and Integration of the Group for Aeronautical Research and Technology in EURope (GARTEUR)
LIST OF AUTHORS

J.B.R.M. Spee

1 Affiliation:
National Aerospace Laboratory NLR
P.O. Box 90502
1006 BM AMSTERDAM, The Netherlands
Telephone : +31 (0)20 511 3113
Fax : +31 (0)20 511 3210
SUMMARY

This document is deliverable FM-AG08-4-D-1.2 of workpackage WP-4.1.2 within the GARTEUR Flight Mechanics Action Group 'Robust Flight Control in a Computational Aircraft Control Engineering Environment' (GARTEUR FM-AGO8).

This report describes determining requirements for a 'Computational Aircraft Control Engineering Environment (CACEE)', based on a limited literature survey. The CACEE will have to increase engineering productivity, by offering computer-based support for integrated application of methods and tools in the process of aircraft control systems engineering.

Requirements are listed in a tree with main elements:
- Functional Requirements
  - Process Support
  - Engineering Method Support
  - Common Tool Support - Framework Services
  - Control Engineering Tools
- Design Limitations
- General Requirements

Comments are given for most of the requirement statements.

Developments are most needed for: Data Base Management System support, facilities for modelling general dynamic systems, and a functional language standard.
### DISTRIBUTION LIST

**FM-AG08 principal persons**

<table>
<thead>
<tr>
<th>Name</th>
<th>Organization</th>
<th>Copies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ahmed S (GB)</td>
<td>CCL</td>
<td>1</td>
</tr>
<tr>
<td>Ambrosino G (IT)</td>
<td>UN</td>
<td>1</td>
</tr>
<tr>
<td>Delgado I (ES)</td>
<td>INTA</td>
<td>1</td>
</tr>
<tr>
<td>Dormido S (ES)</td>
<td>UNED</td>
<td>1</td>
</tr>
<tr>
<td>Duranti P (IT)</td>
<td>ALN</td>
<td>1</td>
</tr>
<tr>
<td>Helmersson A (SE)</td>
<td>LiTH</td>
<td>1</td>
</tr>
<tr>
<td>Irving J (GB)</td>
<td>BAe (Warton)</td>
<td>1</td>
</tr>
<tr>
<td>Joos H D (DE)</td>
<td>DLR</td>
<td>1</td>
</tr>
<tr>
<td>Luckner R (DE)</td>
<td>DAA</td>
<td>1</td>
</tr>
<tr>
<td>Magni J F (FR)</td>
<td>CERT-ONERA</td>
<td>1</td>
</tr>
<tr>
<td>Muir E (GB)</td>
<td>DRA (Bedford)</td>
<td>1</td>
</tr>
<tr>
<td>Post H (NL)</td>
<td>FAC</td>
<td>1</td>
</tr>
<tr>
<td>Sheen P (GB)</td>
<td>AVRO Intl</td>
<td>1</td>
</tr>
<tr>
<td>Ståhl-Gunnarsson K (SE)</td>
<td>SMA</td>
<td>1</td>
</tr>
<tr>
<td>Terlouw J C (NL)</td>
<td>NLR</td>
<td>3</td>
</tr>
<tr>
<td>Vaart J C van der (NL)</td>
<td>DUT</td>
<td>1</td>
</tr>
<tr>
<td>Verde L (IT)</td>
<td>CIRA</td>
<td>1</td>
</tr>
</tbody>
</table>

**Members of the Flight Mechanics, Systems and Integration Group of Responsibles**

<table>
<thead>
<tr>
<th>Name</th>
<th>Organization</th>
<th>Copies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brännströmm B (SE)</td>
<td>FMV-F-FL</td>
<td>1</td>
</tr>
<tr>
<td>Doorn J T M van (NL)</td>
<td>NLR</td>
<td>1</td>
</tr>
<tr>
<td>England P (GB)</td>
<td>DRA (Bedford)</td>
<td>1</td>
</tr>
<tr>
<td>Rodloff R (DE)</td>
<td>DLR</td>
<td>1</td>
</tr>
<tr>
<td>Verbrugge R A (FR)</td>
<td>ONERA-IMFL</td>
<td>1</td>
</tr>
</tbody>
</table>

**Heads of National Delegates**

<table>
<thead>
<tr>
<th>Name</th>
<th>Organization</th>
<th>Copies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bouchet J (FR)</td>
<td>DGA/DPSA</td>
<td>4</td>
</tr>
<tr>
<td>Earwicker M J (GB)</td>
<td>DRA (Farnborough)</td>
<td>4</td>
</tr>
<tr>
<td>Döllinger W (DE)</td>
<td>Bundesministerium für Forschung und Technologie</td>
<td>4</td>
</tr>
<tr>
<td>Houweilingen J van (NL)</td>
<td>NLR</td>
<td>4</td>
</tr>
<tr>
<td>Persson L B (SE)</td>
<td>FFA</td>
<td>4</td>
</tr>
</tbody>
</table>
Members of the Executive Committee

Abbink F (NL)  NLR  1 copy
Duc J M (FR)  ONERA  1 copy
Gustafsson A (SE)  FFA  1 copy
Haupt R (DE)  DLR/PD-L  1 copy
Coleman G (GB)  DRA (Farnborough)  1 copy

Secretary GARTEUR

Sombroek L (NL)  NLR  1 copy

Others

Aranda J (ES)  UNED  1 copy
Bennani S (NL)  DUT  1 copy
Boumans L (NL)  NIVR  1 copy
Ciniglio U (IT)  CIRA  1 copy
Cruz M de la (ES)  UNED  1 copy
Dang Vu B (FR)  ONERA (Salon)  1 copy
Game G (GB)  BAe (Filton)  1 copy
Gennuso D (IT)  ALN  1 copy
Grübel G (DE)  DLR  1 copy
Green M (GB)  AVRO Intl  1 copy
Hulme K (GB)  BAe (Warton)  1 copy
Huynh H T (FR)  ONERA (Salon)  1 copy
Hyde R (GB)  CCL  1 copy
Labarrère M (FR)  CERT-ONERA  1 copy
Lambrechts P (NL)  NLR  1 copy
Martinèz A (ES)  INTA  1 copy
Menard P (FR)  AS  1 copy
Moormann D (DE)  DLR  1 copy
Morales M (ES)  INTA  1 copy
Mulder A (NL)  FAC  1 copy
Rouwhorst W (NL)  NLR  1 copy
Parra de la (ES)  INTA  1 copy
Scala S (IT)  CIRA  1 copy
Schuring J (NL)  NLR  1 copy
Schuring-Coops (NL)  NLR  1 copy
<table>
<thead>
<tr>
<th>Name</th>
<th>Institution</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith P (GB)</td>
<td>DRA (Bedford)</td>
<td>1 copy</td>
</tr>
<tr>
<td>Spee J (NL)</td>
<td>NLR</td>
<td>1 copy</td>
</tr>
<tr>
<td>Tonon A (IT)</td>
<td>ALN</td>
<td>1 copy</td>
</tr>
<tr>
<td>Valk P (NL)</td>
<td>DUT</td>
<td>1 copy</td>
</tr>
<tr>
<td>Weiden A van der (NL)</td>
<td>DUT</td>
<td>1 copy</td>
</tr>
<tr>
<td>Weise K (DE)</td>
<td>DAA</td>
<td>1 copy</td>
</tr>
</tbody>
</table>
CONTENTS

LIST OF ABBREVIATIONS

1 INTRODUCTION

2 DETERMINING REQUIREMENTS
   2.1 Functional Requirements
      2.1.1 Process Support
      2.1.2 Engineering Method Support
      2.1.3 Common Tool Support - Framework Services
         2.1.3.1 Data Base Services
         2.1.3.2 Modelling Services
         2.1.3.3 Task Management Services
         2.1.3.4 User Interface Services
         2.1.3.5 Message Passing Services
         2.1.3.6 Knowledge Base Services
      2.1.4 Control Engineering Tools
         2.1.4.1 Simulation Tools
         2.1.4.2 Trimming Tools
         2.1.4.3 Linearisation Tools
         2.1.4.4 Linear Control System Design
         2.1.4.5 Result Visualisation Tools
         2.1.4.6 Numerical Analysis Tools
         2.1.4.7 Symbolic Manipulation Tools
         2.1.4.8 Documentation Tools
         2.1.4.9 Controller Implementation Tools
      2.2 Design Limitations
      2.3 General Requirements

3 CONCLUSIONS

4 REFERENCES
LIST OF ABBREVIATIONS

ALN       Alenia Un'Azienda FINMECCANICA S.P.A.
AS        Aérospatiale
AVRO Intl  AVRO International Aerospace
BAe       British Aerospace
CACE      Computer Aided Control Engineering
CACEE     Computational Aircraft Control Engineering Environment
CCL       Cambridge Control Ltd
CERT      Centre d'Etudes et de Recherces de Toulouse
CIRA      Centro Italiano Ricerche Aerospaziali S.p.A.
DAA       Daimler-Benz Aerospace Airbus
DBM       Data Base Manager
DBMS      Data Base Management System
DLR       Deutsche Forschungsanstalt für Luft- und Raumfahrt
           (German Aerospace Research Establishment)
DRA       Defence Research Agency
DUT       Delft University of Technology
ECMA      European Computer Manufacturers Association
FAC       Fokker Aircraft B.V.
FFA       Flygtekniska Försöksanstalten
           (The Aeronautical Research Institute of Sweden)
FM-AG08   Flight Mechanics Action Group 8
FMV-F-FL  Försvarets Materielverk
           (Defense Material Administration)
FRM       Framework Reference Model
INTA      Instituto Nacional de Técnicas Aerospacial
           (National Institute for Aerospace Technology)
LITH      Linköping University
NIVR      Nederlands Instituut voor Vliegtuigontwikkeling en Ruimtevaart
           (Netherlands Agency for Aerospace Programs)
NLR       Nationaal Lucht- en Ruimtevaartlaboratorium
           (National Aerospace Laboratory, The Netherlands)
ONERA     Office National d'Etudes et de Recherches Aérospatiales
           (National Institute for Aerospace Research and Studies)
SEE       System Engineering Environment
SMA       Saab Scania AB, Military Aircraft
UN        Università di Napoli "Fedirico II"
UNED      Universidad Nacional de Educación a Distancia
1 INTRODUCTION

Background
The state of practice of aircraft control systems engineering has notable limitations. Control engineering methods do not support the entire aircraft control system development process and lack provisions for integrated application. Numerous computer-based tools exist, but they run as stand-alone packages on various platforms, each with its own user interface and data format. Insufficient support is present for concurrent multi-disciplinary engineering in current methods and tools.

It has become evident that integration of methods and tools is required to significantly improve the efficiency of the aircraft engineering process and at the same time achieve maximum aircraft performance. The prominence of highly interactive dynamic systems in aircraft gives aircraft control systems engineering a vital role in such an integrated process.

Advances in both computer technology and Systems Engineering Environments (SEE) make it feasible to build a 'Computational Aircraft Control Engineering Environment (CACEE)'. This CACEE will offer computer-based support for integrated application of methods and tools in the process of aircraft control systems engineering, thereby reducing development cost, shortening development time and improving (total) product quality.

Identification
This document describes the results of a literature survey into requirements for a CACEE. The literature survey has been carried out in the context of the GARTEUR Flight Mechanics Action Group 'Robust Flight Control in a Computational Aircraft Control Engineering Environment' (GARTEUR FM-AG08). This document is the deliverable of workpackage WP-4.1.2 within the GARTEUR project and is identified as FM-AG08-4-D-1.2.

Purpose
The purpose of this document is to recognise requirements for a Computational Aircraft Control Engineering Environment from literature sources. These requirements should complement the specific requirements for the FM-AG08 CACEE which will be put forward by the FM-AG08 participating organisations. Other CACEE requirements stem from a questionnaire into the processes and methods as applied in aircraft control engineering by the FM-AG08 participants. All three requirements sources will be used to compile a requirements document describing the set of requirements suited to CACEE users within the FM-AG08 organisations.

Scope
A limited time-frame was available to conduct this literature survey. The scope of this document is therefore limited to high level (determining) requirements for a CACEE; requirements that must be met by subsequent design(s) and implementation(s) of a CACEE.
Document overview

Chapter 2 gives a listing of determining requirements for the CACEE. This requirements list is organised as a tree, with a subsequent ordering of sections. The nodes of the tree are:

- Functional Requirements;
  - Process Support;
  - Engineering Method Support;
  - Tool Support - Framework Services;
    - Data Base Services;
    - Modelling Services;
    - Task Management Services;
    - User Interface Services;
    - Message Passing Services;
  - Control Engineering Tools;
    - Simulation Tools;
    - Trimming Tools;
    - Linearisation Tools;
    - Linear Control System Design;
    - Result Visualisation Tools;
    - Numerical Analysis Tools;
    - Symbolic Manipulation Tools;
    - Documentation Tools;
    - Controller Implementation Tools;
  - Design Limitations;
  - General Requirements.

Requirements statements are the leaves on the tree. Most of them have a comment for clarification.

Chapter 3 gives the conclusion to this document.
2 DETERMINING REQUIREMENTS

2.1 Functional Requirements

2.1.1 Process Support

1. The fundamental requirement for the CACEE (the main functional requirement) is to support the Aircraft Control Systems Engineering Processes as applied by the FM-AG08 members. The Flight Control Law Development & Clearance process will be modelled in workpackage WP-4.1.1 of the FM-AG08 project. Parts of this process model will be supported in the CACEE.

2. The CACEE environment should enable concurrent multi-disciplinary design of highly interactive systems (Ref. 7).

McCruer cites projections for aeronautical development in the near future, which emphasise strong dynamic interactions among aircraft technical disciplines and a promise of dynamic systems integration as a central feature in the creation of future aircraft. This requires broad and comprehensive understanding of the aerodynamics; structures; propulsion; and guidance, navigation and control disciplines and their interactions.

3. The CACEE shall offer computer based support for methods and activities in the different phases of the control engineering process (Ref. 12).

Schefström et al. (Ref. 12) describe the relationship between organisational level and support technology (Fig. 1). Activities are supported by tools, methods by toolsets and the development process by an environment.

The control law design process at Boeing Helicopters is presented by Landis et al. (Ref. 8). Four phases are distinguished in this process (Fig. 2):

- preliminary design;
- detailed design;
- implementation / verification;
- acceptance testing.

The preliminary design establishes the control law architecture, sensor requirements and system component bandwidth requirements. The detailed control law design ensures that all handling qualities, system and structural requirements are simultaneously met. Comprehensive models are developed during this phase for air vehicle, sensors, pilot, control law, fuel control and engines. The definition of a complete set of integrated control laws is the result of this phase. Verification and validation of the total flight control system including the control laws is performed in a test rig and on the aircraft to ensure that system integrity is maintained.

Taylor et al. (Ref. 5) describe the process as:
- Modelling the plant to be controlled;
- Determining the characteristics of the plant model;
- Modifying the configuration to make the plant more amenable to control (e.g. moving or adding actuators or sensors);
- Formulating the elements of the design problem;
  - Plant characteristics: is the plant that is to be controlled linear or nonlinear? How nonlinear is it? Can it be linearised? Is it stable or unstable? Are there any right-half-plane zeros? Are there resonances? Is the plant controllable and observable?
  - Design Constraints: What are the limitations with respect to controller implementation (analog or digital), complexity (order), structure (e.g., decentralisation), data rates or sampling time (if digital)?
  - Specifications: How must the control system behave or perform? Common indications or measures of performance include rise time, bandwidth, percent overshoot, pole position, gain margin, phase margin, quadratic performance indices, sensitivity, and robustness.
- Checking to see that the design problem is well-posed. Especially, determining the realism of specifications in light of the given plant model and design constraints;
- Executing appropriate design procedures;
- Performing design tradeoffs, if necessary;
- Validating the design;
- Providing complete documentation of the final design;
- Implementing the final design.

MacFarlane et al. (Ref. 1) describe control systems analysis and design from the individual engineer's point of view. Computer-aided design is an interactive process, in which data passed from man to machine are called 'drivers', and data passed from machine to man are called 'indicators' (Fig. 3).

The main problems lie in the interface between the different components of the design feedback loop, which can be re-phrased to requirements:
4. The CACEE shall support the mapping of customer wishes onto tractable (mathematical) specifications (Ref. 1);
5. The CACEE will enable the incorporation of specifications and constraints of various kinds into a design strategy (Ref. 1);
6. The CACEE shall have drivers which efficiently communicate the design decisions to the design machine (Ref. 1);
7. The design machine will be loaded with expert methods and procedures (Ref. 1);
8. The design machine shall have indicators which show decision-relevant information about candidate solutions and effectively communicate this information to the designer (Ref. 1);
9. The design results have to be formulated quantitatively, both in performance and cost, as detailed as required, and as close as possible to the specification language of the customer, in order to come up with trustful negotiations with the customer for compromise balancing.
This in turn implies that in performance modelling criteria of any kind must be allowed (Ref. 6).

10. The design computer will have an interface with the implementation computer (controller) (Ref. 1).

11. Appropriate means for an efficient re-iteration of activities shall be integrated into the overall design procedure (Ref. 1).

Control engineering is an iterative process (Ref. 1, Ref. 6). Grübel describes it as searching for a well balanced design tradeoff among conflicting goals and restraints.

12. Development environments must account for very flexible iteration among any kind of phases of the development process they try to impose (Ref. 9).

This problem is one of the most central, and occurs in many shapes. It is often talked about in terms of 'traceability', referring to the need for tracing back to the source of problems or decisions.

13. An information system should aim at creating a resource envelope within which people can adapt their work strategies to the current task and personal preferences without loosing support from the system. An integrated system should neither be based on normative work procedures, nor on past role allocations (Ref. 4).

14. Development environments must provide for several styles of working (Ref. 9).

2.1.2 Engineering Method Support

1. The CACEE shall support control engineering methods as applied by FM-AG08 partners.

A survey among the partners revealed as controller design methods (Ref. 11):
- Classical Control Theory;
- Robust Eigenvalue/Eigenvector Structure Assignment;
- Linear Quadratic, Linear Quadratic Gaussian, Loop Transfer Recovery;
- Hinf Techniques;
- μ-Analysis and Synthesis Techniques;
- Lyapunov methods;
- Quantitative Feedback Theory;
- Space Parameter;
- Algebraic Techniques;
- Linear Matrix Inequalities;
- Optimisation Techniques;
- Kharitonov techniques;
- Predictive Control;
- Non-linear Dynamic Inversion;
- Robust Inverse Dynamic Estimation;
Neural Networks.

Note that this list only contains controller design methods. Engineering methods for specification, verification, etc. would be required to give complete support for typical control engineering processes as identified in paragraph 2.1.1.

2. The engineering environment shall support the integrated application of methods on which the engineering process is based (Ref. 9).

No single method covers all phases and aspects of control systems engineering. This implies either applying one method outside it's intended domain or using best methods for each individual purpose and trying to cope with potential incompatibilities.

Kronlöff (Ref. 9) defines a method as consisting of:
- an underlying model, referring to the classes of objects represented, manipulated and analysed by the method;
- a language, the concrete means of describing the product of the method, serving as the user interface to the underlying model;
- defined steps and ordering of steps, referring to the manner of performing activities by the user of the method;
- guidance for applying the method, typically takes the form of informal text distributed over a diverse collection of literature, such as manuals, handbooks, guides, etc.

Different design objects can be attributed to different phases of the development cycle. Specialised and largely incompatible notations have often been developed for the roles corresponding to these different phases. Just the first step, of combining and harmonizing notations, is a special and large problem. It is sometimes called 'method integration' (Ref. 9).

Integration of (best) methods is based on the compatibility of both the underlying models and the languages of methods.

3. The CACE environment shall interface with methods from related (engineering) disciplines.

This is a derived requirement, stemming from the statement about concurrent multi-disciplinary design (section 2.1.1). Aerodynamics, structures, propulsion and systems can be considered as the main aircraft engineering disciplines. Control engineering can be seen as part of the systems engineering discipline.

A study by NASA identified 55 systems engineering methods (Ref. 10) in the categories:
- Concept development;
- System safety and reliability;
- Design-related analytical;
- Graphical data interpretation;
- Statistical;
- Total Quality Management;
- Trend analysis.

4. The CACE environment shall support the working methods of designers (Ref. 1).
   In terms of the amount of analytical knowledge the designer has at his disposal, we can broadly
   split his possible methods of working into those which are:
   - attribute-centred, or analytical, when an exactly computable answer is available;
   - operation-centred, or procedural, when he is working from an extensive knowledge base but
     does not have an exact prescription for a solution;
   - object-centred, or experimental, when he is working from a restricted knowledge base and is
     systematically searching for a solution.

5. For analytical design, we must from the underlying theory devise:
   - a set of overall organising principles which, although couched in an informal
     and highly intuitive way, are based solidly on the appropriate theoretical
     constructs (theorems, necessary and sufficient conditions, etc.);
   - a set of functions covering the whole range of attribute and behaviour
     generation, model manipulation and data transformation which are involved in
     analytical design procedures;
   - a formally-specified functional language in terms of which the functions may be
     interactively executed;
   - an appropriate set of indicators by means of which system attributes can be
     displayed at the man-machine interface;
   - suitable conventions for relating models to their underlying data structures
     which can be used to define and manipulate the models via the command
     language;
   - an informal framework for a highly supportive form of "soft" interaction, by
     means of which the machine may be used to examine the definition and use of
     functions and their relationship to the underlying concepts and theory (Ref. 1).

The specification, development, standardisation and adoption of a functional language should be a
key objective for future interactive design environments.

6. It is important that design by search is a fast process, and this requires:
   - drivers that are easy to use by the designer;
   - fast computation of candidate solutions;
   - visualisation of the essentials of a complex problem via indicators which allow
     an easy interpretation in terms of the original specifications and the plant
     subsystems and operation (Ref. 1).
7. An essential feature of the support of search by procedural methods is that they rule out a large set of candidate solutions by formal criteria (Ref. 1).

2.1.3 Common Tool Support - Framework Services

Barker et al (Ref. 2) propose a CACE Framework Reference Model (CACE FRM), based on an existing Framework Reference Model (FRM) for Computer-Aided Software Engineering (see fig. 4) from the European Computer Manufacturers Association (ECMA). The principal feature of the FRM is the separation of the environment into a set of tools which provides the specialised facilities required by the user, and a framework, which provides the common facilities required by the whole environment.

The common facilities are provided in the form of six groups of framework services:
- data base services;
- modelling services (ECMA FRM: data integration services);
- task management services (ECMA FRM: process management services);
- user interface services;
- message passing services;
- knowledge base services (extension to ECMA FRM).

Requirements are given per service group in this section. Requirements for control engineering tools are addressed in section 2.1.4.

2.1.3.1 Data Base Services

1. A Data Base Management System (DBMS) is a key component in any CACE architecture. The data environment ensures that data generated by any tool can be used by any other tool.

Schefström et al. (Ref. 12) claim that (engineering) environment databases must provide the following:
2. The datamodel offered should be close, preferably equivalent, to what an application program would anyway use. ... The database must also accommodate change in its basic structure, its schema, in a flexible way (data model and schema management);
3. The overall database must be possible to build up through pieces that can be independently accessed and distributed (distribution and multidatabase architecture);
4. It must efficiently handle a granularity of change that is quite large, and that can be extended over time (long transactions);
5. It must be possible to introduce, maintain, and track multiple versions of what otherwise should be considered as the same entity. This then implies that larger sets of such versions must be possible to manage systematically, that is, support for (version and) configuration management;
6. Different users must be able to independently work on different configurations without replicating parts that are still shared (multiple views);

7. The environment database is to act as the foundation for a highly dynamic and changing process: that of systems development. It must be possible to allow change in a controlled way, and to propagate changes to depending entities and even to other users (change and consistency management);

8. There must be support for systematically re-integrating the changes implied by the many and parallel long transactions in a way that maintains control over the consistency and makes cooperation among users smooth (support for work process and cooperation).

Taylor et al. (Ref. 3) consider the basic data elements in CACE to be models which are comprised of components and a description (containing model type, connection definition, key variable names, etc.). Associated with each model there are results. Models and results are often organised in terms of projects (Fig. 5).

9. Database integrity must be maintained at the component level (Ref. 3).

10. Relations between results and models shall be tracked (Ref. 3).

11. Any derived (linear) component in a database can be traced back to determine how it was obtained (Ref. 3).

12. It is essential to capture the so called condition specification (Ref. 3). This is the secondary data element in CACE (Ref. 3), containing information regarding operations performed on a model before the result is obtained. These operations include changing a parameter value, specifying an initial condition and/or output signal before performing a simulation, defining a frequency list before obtaining Bode plot data, etc. The condition specification also records numerical conditions, such as setting a tolerance for determining controllability or observability, selecting an integration algorithm for simulation, etc. Capturing this data is essential since it is the combination of model instance and condition specification that determines the result and thereby allows engineers to document or repeat the result.

13. Browsers should be provided to search the data bases and data definitions in a convenient manner and exhibit data in various formats (Ref. 2).

14. The user needs support for the organisation of the large amounts of data and intermediate results which are produced in the iterative, step-wise design procedure (Ref. 4).

15. The variability of possible design decisions and the amount of computational data necessary to get the proper indicators for design decisions, make it necessary to support the designer in automatically recording the various design steps and persistent data (Ref. 6).
16. An automatically evolving data structure should support the design process over time, by recording all design steps for backtracking and for comparing different design steps (Ref. 6).

2.1.3.2 Modelling Services

1. The CACE environment should provide a rich set of basic modelling facilities (Ref. 2). The need to support the many different models used within the control engineering discipline should be recognised as well as the need to provide support for models generated by engineers of other disciplines.

2. A common kernel based on a class of standard data types should be provided (Ref. 2). Joos et al. (Ref. 14) propose a set of control engineering data structures, based on the class concept and especially the inheritance mechanism of object-oriented data models. The control engineering data types they propose are:
   - General nonlinear dynamic systems;
   - Time-invariant state space linear systems;
   - Transfer matrices;
   - Real functions (i.e. time functions);
   - Complex functions (Frequency response).
These are based on the object types: integer, real, double, character (string), name (identifier). Feasible operations on these data types are:
   - Listing or graphical visualisation;
   - Connection of objects within a class;
   - Transformation within a class;
   - Transformation into another class;
   - Generation of partial object

3. Facilities for modelling general dynamic systems should be provided (Ref. 2). These include:
   - Non-linear systems (Ref. 3);
   - Linear systems (Ref. 3);
   - Continuous systems (Ref. 3);
   - Discrete-time systems (Ref. 3).

The unifying principle that underpins an open CACE environment is a model of a dynamic system. Such models represent the engineer's knowledge of the process to be controlled, provide abstractions of the plant suitable for simulation or linear analysis, enable suitable structures to be fitted to measurement data, provide suitable structures for controllers and represent implementation architectures.

4. At the level of the tool interface to the CACE modelling services layer there must be a single form of dynamic system model definition (Ref. 2).
The differential-algebraic equation (DAE) formalism provides the closest thing we have at present to a universal method of describing dynamic systems.

5. **New, more powerful, general purpose modelling methodologies and standards should be developed (Ref. 2).**

Continuous system simulation language (CSSL) and matrix-based linear systems models are becoming strained in an era where most controllers are discontinuous, complex nonlinear models from a wide variety of application domains need to be brought into control system design, and discrete events are common.

6. **A modelling standard should enable a higher-level definition of models for a variety of disciplines, without sacrificing the modelling methodologies that have evolved over many years.**

7. **A set of Control Data Objects (CDOs) should be provided by the modelling services of the CACE environment.**

CDOs include common linear control types such as state-space models and transfer function matrices and result types such as time and frequency responses. Thus CDOs form a central mechanism for facilitating the exchange of models and results with simulation and and analysis tools within a CACE environment.

8. **The representation should support a number of mathematical and logical frameworks for representing model behaviour (Ref. 4).**

9. **It must have concepts for structuring of large models, e.g. hierarchical submodel decomposition (Ref. 4).**

10. **It should be possible to reuse parts of models in other models (Ref. 4).**

11. **It should be possible to include "redundant" information in models for the purpose of documentation and automatic consistency check (Ref. 4).**

12. **It must be possible to extract information to support the needs of different tools (Ref. 4).**

13. **The representation should be standardised in order to allow exchange of models between users and tools (Ref. 4).**

14. **The designer must be enabled to interact with the physics of the real world through physically structured mathematical models, not through a particular type of computational model as it is required for the processing of a design computer (Ref. 6).**
2.1.3.3 Task Management Services

1. All tools in the CACE environment must obey a common command protocol so that the environment and other tools can communicate with the user, the environment and with each other (Ref. 2).

2. The main interface to the task management service should be provided by a standard command language. From the point of view of end-user and developer acceptance, this language should be compatible with the popular matrix-based command languages used in MATLAB, MATRIX, and Ctrl-C. But it should also provide for higher-level constructs such as control data object definition and manipulation, operator overloading, and extensibility through user-defined functions (Ref. 2).

2.1.3.4 User Interface Services

1. It is important that the user interface be consistent across the whole environment, thus making it easier to learn how to use both existing and new tools (Ref. 2).

2. In the modern computing environment, windows-based user interfaces are an essential feature (Ref. 2).

It makes a great deal of sense to provide a graphical front end to the CACE environment. This front end would communicate with the environment via the command language thereby providing efficient access to common commands and facilities.

3. However, the CACE environment would also provide a direct command line interface to the command language for experienced users (Ref. 2).

4. The 'designer-customer discussion' should be supported by a display of design alternatives which is quickly accessed by the designer and easily understood by the customer (Ref. 1).

5. The CACE package must allow the user to work at a high level without the necessity of mastering a large ensemble of low-level commands (Ref. 5).

6. The CACE man-machine interface should specifically support the design engineer's role as a decision maker (Ref. 6).

7. To improve making a decision of how to change the design to 'improve the balance' of a 'best-compromise' design requires to know which criteria behave in coherence and which criteria are in conflict (Ref. 6).
2.1.3.5 Message Passing Services

1. Exchange of messages and data between communicating processes should be incorporated in the CACE environment (Ref. 2).

The advantage of this is that the CACE environment need not be a single monolithic process running on a single machine but may be constructed as a distributed application which calls on a makes use of services wherever they may be located on an heterogenous network.

2.1.3.6 Knowledge Base Services

1. A powerful package for computer-aided control system design by procedure must have not only a user-friendly interface, reliable numerical software, good interactive graphics capabilities, and good data base management, but also an extensive control engineering knowledge base (accessed through an expert system) (Ref. 1).

2. The CACE package should provide guidance for the less-than-expert user (Ref. 5).

Guidance is needed for formulating a meaningful model of the plant, posing the design problem appropriately, knowing what procedures to execute, judging the meaningfulness of the results obtained at each step, knowing how to proceed.

3. The designer wants to use the interactive computing environment to handle formal declarative knowledge by evaluating for him the behaviour and attributes of any given dynamical model (Ref. 1).

4. The designer wants to use the interactive computing environment to handle formal imperative knowledge by executing appropriate sequences of procedures in order to attain the specified objectives (Ref. 1).

5. The designer wants to use the interactive computing environment to handle informal declarative knowledge in the form of textual descriptions of background theory, codes of practice, design databases, and so on (Ref. 1).

6. The designer wants to use the interactive computing environment to handle informal imperative knowledge in the form of design guidelines, rules of practice, mandatory design requirements, etc (Ref. 1).
2.1.4 Control Engineering Tools

Requirements for (control) engineering tools were found in several papers. No specific comments on these tools are incorporated here, since most are well known to the intended readership of this document.

2.1.4.1 Simulation Tools
1. *Open simulation engine (Ref. 2);*
2. *Initialising system state variables (Ref. 3);*
3. *Setting system parameters (Ref. 3);*
4. *Defining input signals (Ref. 3);*
5. *Designating simulation variables for storage (Ref. 3);*
6. *Executing a simulation (Ref. 3);*
7. *Linear and non-linear models (Ref. 3).*

2.1.4.2 Trimming Tools
1. *Tool for determining steady-state operating points.*

2.1.4.3 Linearisation Tools
1. *Eigenvalues, eigenvectors (Ref. 3);*
2. *Zeros (Ref. 3);*
3. *Controllability and observability (Ref. 3);*
4. *Model reduction (Ref. 3);*
5. *Model transformations (Ref. 3);*
6. *Root locus (Ref. 3);*
7. *Frequency response (Ref. 3);*

2.1.4.4 Linear Control System Design
1. *Frequency domain methods (Ref. 3);*
2. *Pole placement (Ref. 3);*
3. *Time domain methods, like LQG and LQR (Ref. 3);*

2.1.4.5 Result Visualisation Tools
1. *Time response visualisation;*
2. *Frequency response visualisation;*
3. *Analysis plots like Bode, Nyquist, etc.*

2.1.4.6 Numerical Analysis Tools
1. *The CACEE should provide an open collection of algorithms to support the applied control engineering methods.*
2.1.4.7 Symbolic Manipulation Tools
1. Symbolic manipulation should provide the means for:
   - Performing complex nonlinear analysis;
   - Enable analysis of structured models and discrete event systems;
   - Expert systems support.

2.1.4.8 Documentation Tools
1. A flexible system of documentation tools, which is itself part of the environment, is required (ref. 2).

2.1.4.9 Controller Implementation Tools
1. The CACE environment should support the selection of controller hardware, sensors, actuators and their positions (Ref. 1).
2. The CACEE should support the generation of controller software (Ref. 1).
3. The CACEE should enable easy implementation of controller software in the controller hardware (Ref. 1).
4. The CACEE will have to support of testing the implemented controller at the plant (Ref. 1).

2.2 Design Limitations
1. An absolute minimum for a realistic approach to CACE is software for nonlinear simulation, linearisation, linear analysis, and linear controller design (Ref. 5).

2. No proprietary tool should be required to implement the environment (Ref. 2).
The cost of other CACE development projects were extremely high because of the price of the proprietary software used in the systems. Open systems have ensured success in other areas of information technology, encouraging diversification and value for money for the consumer without loss of market share for the package developer (Ref. 2).

An open system standard is defined as "a specification developed by a consensus process to which any vendor can build products". Characteristic features of openness are:
- Products are implemented to internationally agreed standards;
- Standards are nonexclusive, nonproprietary, and vendor independent;
- Applications can be moved as necessary between systems of different makes and sizes;
- Usable information can be exchanged when required between different systems.

3. There are four important areas of integration which have to addressed to achieve tool integration (Ref. 2), (Ref. 12):
   - Presentation integration. The aim of presentation integration is to reduce the cognitive load on users by providing a consistent and predictable user interface.
It applies to individual tools, sets of related tools and to the environment itself. The aim is to reuse the user’s experience by:

- reducing the number of interaction and presentation paradigms;
- providing paradigms that match the user’s mental models;
- providing useful and correct help and information.

Two tools are said to be well integrated with respect to presentation integration if they use similar screen appearance and interaction behaviour and are based on similar mental models and metaphors.

Data integration. If they are to be truly integrated, tools must be able to share data. There are a number of levels of data integration:

- Interoperability. Two tools are said to be well integrated with respect to interoperability if they must do little work to be able to use each other’s (persistent) data. This requires a common view of data which may be achieved in CACE by recognizing the basic importance of equation-based models and cataloguing the basic data objects used in control systems design.

- Nonredundancy. Two tools are said to be well integrated with respect to nonredundancy if they have little duplicate data or data that can be automatically derived from other data. For example a linear system model can often be derived from a nonlinear model. To avoid problems of consistency, for example, it may preferable to perform linearisation prior to performing some analysis of a system rather than maintaining two different versions of the same system in the database.

- Data consistency. Two tools are said to be well integrated with respect to data consistency if each tool indicates its actions and the effects on its data that are subject of semantic constraints that also refer to data managed by the other tool.

- Data exchange. Two tools are said to be well integrated with respect to data exchange if little work on the format and the semantics of the data is required for them to exchange the data. This is similar to the interoperability definition, but it applies to nonpersistent data, for example data that is exchanged via a cut-and-paste clipboard.

- Synchronisation. Two tools are said to be well integrated with respect to synchronisation if all changes made to all shared data by one tool are communicated to the other.

Control integration. Tools must share functionality. Ideally all functions offered by all tools should be accessible to all other tools and tools providing functions need not know what tools will be constructed to use their functions. There is a need to communicate operations, data, or data references between tools and such facilities should be provided by message services in the engineering environment.
- **Process integration.** Process integration is concerned with the actual processing of information done by the integrated tools. It is an issue that is concerned with the particular application area (of the engineering environment). A process model provides larger grain, higher level semantics than the actions of individual tools. A tool embodies a set of assumptions about the process in which it may be used: two tools are said to be well integrate with respect to process if their assumptions about the process are consistent.

4. **The processing tools should be small, robust and efficient.** They should perform one task well and let the environment provide the "glue" that enables tools to be combined to create new applications. Such tools have usually been developed by experts in a specialised field (Ref. 2).

5. **It should easily be possible to:**
   - add a new tool to the environment;
   - extend the command language;
   - reconfigure the graphical user interface;
   - perform customisation on individual tools (for example, to change the language of the menus for local needs) without recompiling the whole system (Ref. 2).

The user need not be concerned with such details and may well not have access to the necessary facilities (extensibility and configurability, Ref. 2).

6. **It should also be possible to combine several different tools to create more powerful ones, thus providing the user with greater potential than that of the sum of the individual tools** (Ref. 2).

7. **It is absolutely essential that tools should be able to communicate with each other.** Data produced by one tool should easily be processed by another without the user being aware of any translation being performed (interoperability, Ref. 2).

8. **Provide support for other applications.** It should be possible to store the graphical or tabular output from any application in a form, such as a PostScript file, suitable for printing or incorporation into a document (Ref. 2).

9. **A key requirement is the need to run, possibly in parallel, many tools and facilities** (Ref. 2).

10. **It should be possible to operate the CACEE on heterogenous networks** (Ref. 2).

Use of standards for operation on heterogenous networks enables the application to be distributed if necessary.
2.3 General Requirements

1. The system must be (Ref. 1):
   - Easily comprehensible to a single individual;
   - Wide in scope;
   - Modular with a manageable number of distinct parts;
   - Predictable in its behaviour;
   - Integrated and coherent in the ways in which its different parts relate;
   - Helpful, with quick and efficient access to relevant information;
   - Tolerant of errors and supportive in enabling the effect of errors to be easily undone;
   - Extensible and adaptable;
   - Self-documenting.

2. Means for analysing the propagation of uncertainty through the design process should be available (Ref. 1).

3. The design engineer should not be bothered by details of data handling, data processing and algorithmic computation (Ref. 6).
3 CONCLUSIONS

Main requirements for a CACEE are:

- **Process Support:** To support the aircraft control systems engineering processes as applied by the FM-AG08 participants, including support for concurrent multi-disciplinary engineering and taking into account flexible iterations of activities in the process;

- **Engineering Method Support:** To support integrated application of control engineering methods on which the process is built, including interfaces with methods of related disciplines and assisting engineers in their working methods;

- **Common Tool Support - Framework Services:** To provide common facilities for all the tools in the environment.

- **Control Engineering Tools:** To provide tools for computer based support of the activities in the engineering process.

- **Design Limitations:** Implementation of the environment should be based on open systems standards. Tool integration should address issues of: 1) presentation, 2) data and 3) control. Tools for nonlinear simulation, linearisation, linear analysis and linear controller design are an absolute minimum.

A DataBase Management System (DBMS) is a key component in any CACE architecture, and which is not currently offered by (commercial) Computer Aided Control System Design packages.

Facilities for modelling general dynamic systems should be provided, based on a neutral description like Differential Algebraic Equation formalism.

The specification, development, standardisation and adoption of a functional language should be a key objective for future interactive design environments.
4 REFERENCES


Fig. 1  Relation to development process and organisation (from Ref. 12)

Fig. 2  Control Law Design Process (from Ref. 8)
Fig. 3 Overall design environment (from Ref. 1)
Fig. 4 ECMA Framework Reference Model for Computer Aided Software Engineering (from Ref. 13)
Fig. 5 Engineering Data Base organisation (from Ref. 3)