Nationaal Lucht- en Ruimtevaartlaboratorium
National Aerospace Laboratory NLR

NLR-TP-2002-331

A SPINEware based computational design engine for integrated multi-disciplinary aircraft design

W.J. Vankan and M. Laban

Nationaal Lucht- en Ruimtevaartlaboratorium
National Aerospace Laboratory NLR

NLR-TP-2002-331

A SPINEware based computational design engine for integrated multi-disciplinary aircraft design

W.J. Vankan and M. Laban

This report is based on a presentation held at AIAA/ISSMO 2002, Atlanta, GA, USA, 4-6 September 2002.

Customer:	National Aerospace Laboratory NLR
Working Plan number:	I.1.A.5
Owner:	National Aerospace Laboratory NLR
Division:	Information and Communication Technology
Distribution:	Unlimited
Classification title:	Unclassified
	August 2002

Summary

This paper deals with the software architecture and the global functionality of the Computational Design Engine (CDE) that is being developed in the MOB project. This CDE comprises a multidisciplinary set of tools for design, analysis and optimisation of blended wing body aircraft. Automatic design evaluation processes, involving complex sequences of analysis computations and data exchange, are available to the user. To guide the user through the complex structure of software tools and data, a user oriented framework, based on the SPINEware middleware system, has been built on top of the functional level implementation of the CDE.

List of acronyms

ASCII	American Standard Code for Information Interchange
BWB	Blended wing body
CAD	Computer aided design
CDE	Computational Design Engine
CFD	Computational fluid dynamics
CM	Controllability margin (flight mechanics)
CORBA	Common object request broker architecture
CU	Cranfield University
DLR	German Aerospace Center
DMZ	De-militarised zone
FE	Finite element
GUI	Graphical user interface
HTTP	Hypertext transfer protocol
LAN	Local area network
L/D	Lift over drag (aerodynamics)
MOB	Project acronym for EU project: A Computational Design Engine Incorporating Multi-Disciplinary Design and Optimisation for Blended Wing Body Configuration
MSIE	Microsoft internet explorer
NLR	National Aerospace Laboratory of the Netherlands
ORB	Object request broker
PC	Personal computer
SPIRL	SPINEware resource locator
SSH	Secure shell
TUD	Delft University of Technology
URL	Universal resource locator
WAN	Wide area network
WWW	World wide web

Contents

(29 pages in total)

This page is intentionally left blank.

1 Introduction

In the MOB project [1] a distributed engineering environment for multidisciplinary design and optimisation of blended wing body (BWB) aircraft configurations is developed. This engineering environment, which is referred to as Computational Design Engine (CDE), comprises a number of design and analysis tools that are provided by the partners of the MOB project's consortium. For example, the tool ICAD for Computer Aided Design (CAD) and multi-model generation [2] is provided by the Delft University of Technology (TUD). Several tools for high and low fidelity Computational Fluid Dynamics (CFD) and structural mechanical analyses [3] are provided by NLR. In addition, tools for flutter analysis [4] are provided by DLR (German Aerospace Center), and tools for low-fidelity aerodynamic analysis [5] are provided by Cranfield University (CU). A tool of the CDE is executed at the site of the partner that makes it available. Exchange of data and control among the tools at the different sites is achieved via secure connections over the internet. Design loops involving a complex sequence of design analyses of the BWB configuration are currently operational. A partially automated process to run these design loops has been integrated on operating system level and can be run by command line execution of a number of shell scripts and executable programs. On top of this "system level CDE" a more user-oriented environment has been built using the SPINEware middleware system [9]. SPINEware is an object-oriented system that supports the construction and usage of user-defined working environments in heterogeneous computer networks.
A SPINEware working environment is based on the definition of a set of objects [10]. These objects and their associated methods represent the functionality of the working environment, and are interpreted by *SPINEware Object Servers*. SPINEware makes use of the CORBA (Common Object Request Broker Architecture) standard for object interactions. This also ensures compliance to other external –CORBA compliant- software systems. SPINEware provides a suitable set of object classes for proper definition of many components in regular engineering environments [8], such as the Atomic Tool and Workflow classes for software tool and workflow components. With its user-oriented graphical desktop system –the *SPINEware User Shell*- SPINEware supports the realisation of flexible and easy-to-use application oriented working environments. It provides a suitable basis for the realisation of a user-oriented multidisciplinary environment as required in the aircraft design process. The SPINEware User Shell is available as a Java applet implementation, by which SPINEware object services are accessible via standard web browsers. Accessibility of and interoperability within SPINEware

working environments is thus well facilitated, not excluding WAN environments where security regulations such as firewalls are to be dealt with.
In this paper the software architecture and the global functionality of the MOB CDE are described, in particular of the SPINEware layer. For a detailed description of the application of the CDE and its underlying analysis tools, reference is made to the companion MOB papers [2][4][5][6][7], and in particular to [3].

2 SPINEware functionality

SPINEware is a middleware system that supports the construction and usage of working environments on top of heterogeneous computer networks. A SPINEware working environment presents a local- or wide-area computer network as a single “metacomputer” on the user’s desktop computer. The working environment provides uniform and network-transparent access to the information, applications, and other resources available from the computer network. To support flexibility, openness, and easy extendibility of SPINEware as well as the SPINEware-based working environments, standard modern technologies are applied. In particular, the application of the CORBA standard facilitates the integration of commercial and other third-party software and the reuse of object-oriented support and communication services provided by software implementations of the CORBA standard (so-called ORBs). SPINEware is available for different Unix architectures (such as SGI Irix, SUN Solaris, HP-UX, Linux) and for Microsoft Windows NT and 2000 [11].
The two main components of SPINEware are the *SPINEware Object Server* and the *SPINEware User Shell*. A *SPINEware Object Server* is a process that manages the availability, accessibility and interpretation of SPINEware objects. SPINEware is fully object-oriented: in a SPINEware working environment, all resources, such as software tools, data and documents, are modelled as objects. SPINEware provides a set of basic object classes such as File and Directory (for representing the native system's files and directories), ObjectFolder and WorkingEnvironment (folder objects for organising SPINEware objects), AtomicTool (for advanced “wrapping” of executable programs), Workflow (for easy tools chaining), Job and Queue (for managing and scheduling tool execution). The user can modify existing classes and create new object classes - whether or not inheriting from existing classes. An object class has a set of methods associated to it, by which a user can manipulate the objects. In the *SPINEware User Shell*, the default method for an object class is a standard access operation on an object, where typical examples

are *Edit* for File, *View* for Directory and ObjectFolder, and *Execute* for AtomicTool. Each object in SPINEware has a world wide unique object identifier, usually represented as a *SPIRL* (SPINE Resource Locator). Beside object identification, SPIRLs can also specify object method invocation. SPIRL syntax is based on the well-known URL naming scheme used in WWW. A complete SPIRL specifies the object's class, a method name (if used for method invocation), its host's Internet name, its location on that host (e.g. its file system path), and optionally any method arguments. SPIRLs are used internally in SPINEware for object operations, but can also be issued by users directly to the command interpreter. SPIRL interpretation is dealt with by the *SPIRL broker* utility, which enables invocations of object methods from within tools, scripts, and command-line interpreters. When a user starts a SPINEware session, one SPINEware object server is started initially on the local host, which will handle all requests from the *SPINEware User Shell*. If an object on another host is requested, the initial SPINEware object server starts ("on demand") a SPINEware object server on the other host, and relays all subsequent method invocations (and the corresponding results) involving objects on that host. The communication among the SPINEware objects is based on CORBA, and is implemented using an off-the-shelf CORBA implementation called *ILU*. This software product supports CORBA-based inter-object communication over LAN and WAN.

The *SPINEware User Shell* provides an intuitive, tailorable Graphical User Interface (GUI). The user shell acts as the interface between the user and the local SPINEware object server, by presenting the required information of SPINEware objects in the connected network on the user's local graphical desktop, and by translating the user's GUI input to SPIRL commands for the local object server. The SPINEware object information is presented through dedicated browsers for each of the SPINEware objects. The user shell can be run in different ways. One way is that the user starts a local user shell process on his local system (e.g., desktop computer) which directly communicates to local or remote SPINEware object servers. The other way is to run the user shell by making use of Java applets and servlets to communicate to the SPINEware object servers [10]. The first way requires the user to have the SPINEware software installed on his local system, or a remote system he has graphical access to. The second way requires the user to have (as a minimum) only a Java-2 enabled web browser on his local system. The access to the SPINEware server system, which runs the actual SPINEware software (either remote or local), is arranged via an HTTP server with Java servlet support. For example, if a user logs on to the system via his web browser, the Login servlet is invoked by the HTTP daemon. The Login servlet starts up the user's SPINEware session by publishing a new instance of the SPINEware User Shell object to the CORBA naming service and launches a SPINEware object

server on the user-specified host using a rexec command. Figure 1 explains the two different operational modes of the SPINEware user shell.

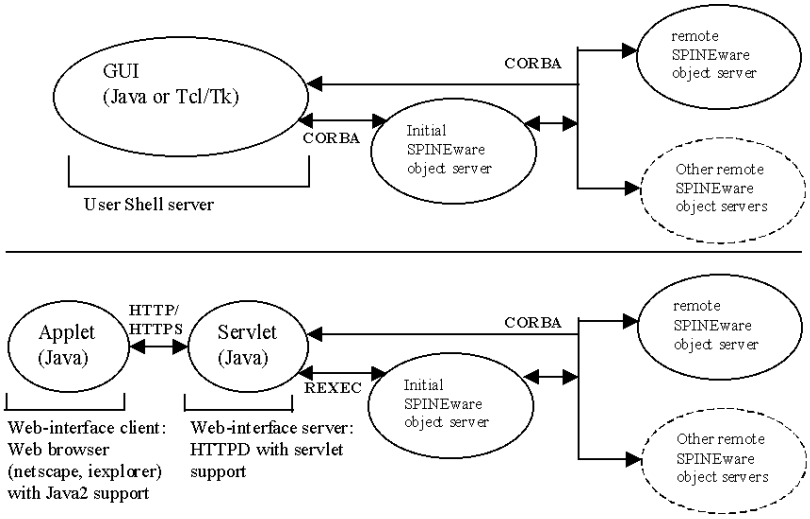

Figure 1: SPINEware User Shell architectures: upper half: local User Shell server; lower half: Java applet based user interface.

The practical application of the SPINEware middleware system is that it provides an integration platform for heterogeneous computer networks in which users can operate in pre-defined working environments, build up, manage and operate their own working environment, and exchange or share their working environment or its objects with other users. The user can for example browse through the working environment's contents, create and execute tools, and submit jobs using point-and-click and drag-and-drop operations. In addition to browsing through information and launching tools on the user's local host, the user may open browsers for accessing resources on other hosts as well. Objects can simply be moved and copied by dragging and dropping icons from one browser window to another. Working environments can be tailored for particular end users and application areas.

SPINEware is used for the construction of the CDE, in order to effectively deal with accessibility, interoperability and exchange of data and tools. The CDE is considered as a SPINEware WorkingEnvironment object, which is distributed over, and accessible from, the sites of the different MOB partners. The components that are integrated into the CDE, such as design and analysis tools, are to be combined into complex sequences of calculations, including data conversion and exchange. Therefore these components should be specified in generic

terms. For example for tool components, such a specification would comprise the tool's execution directory, host (or architecture) specification, path of the executable, exit status, input and output files, options, and lots of other properties. The SPINEware AtomicTool object is able to store all this information, and tool components are integrated into the CDE WorkingEnvironment as AtomicTool objects by making use of a dedicated graphical editor (Figure 2) that guides the user through the integration process.

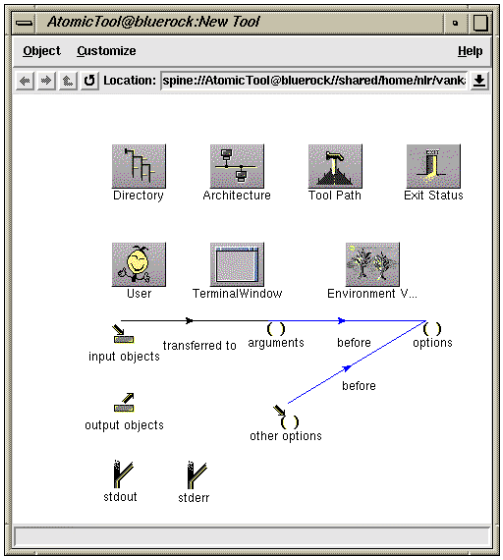

Figure 2: Example of the graphical editor for AtomicTool objects.

Once integrated, an AtomicTool object is readily available for execution in the WorkingEnvironment by easy mouse-click and drag-and-drop operations. In addition, AtomicTool objects can be directly integrated –also by drag-and-drop operation- into chains of interconnected tools, the so-called SPINEware WorkFlow objects. These WorkFlow objects can be assembled into nested structures, leading to multi-level hierarchies of interconnected WorkFlow and AtomicTool objects (Figure 3). WorkFlow objects have several possibilities for complete or partial automatic and conditional execution.

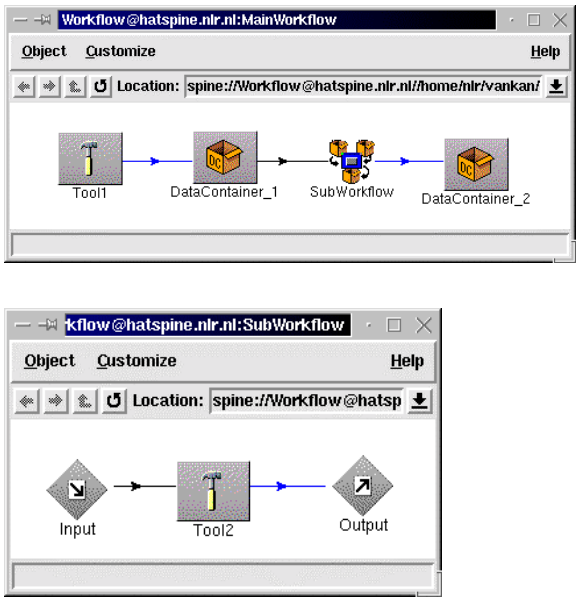

Figure 3: Example of a basic two-level hierarchical WorkFlow, consisting of a main WorkFlow object (upper panel) that contains a so-called ChildWorkFlow (SubWorkFlow in lower panel).

It should be noted that SPINEware objects can be moved, linked, or copied throughout the WorkingEnvironment. In the case of the CDE, the main functionality is stored in a library of integrated tools. Other components that are constructed from these tools in the library, in particular WorkFlow components, are most effectively created by creating links (or “shortcuts”) to these tools, because of software management advantages.

3 CDE software architecture

One of the main objectives for the creation of the CDE is to set up an integrated engineering environment that comprises the tools and methods for aircraft design and that is available to a distributed design team. This engineering environment should incorporate concept, preliminary aircraft and main phase design, and has to ensure continuity of the information flow through the design cycles. The aircraft design processes involve analyses of multiple technical disciplines and of multiple levels of fidelity. The CDE focuses on the multi-disciplinary preliminary aircraft design process, of which the global functional layout as implemented in the CDE is given in Figure 4. Besides this design process, the CDE also contains the functionality for calculation of design-dependent objectives and constraints, and for response surface optimisation [3].

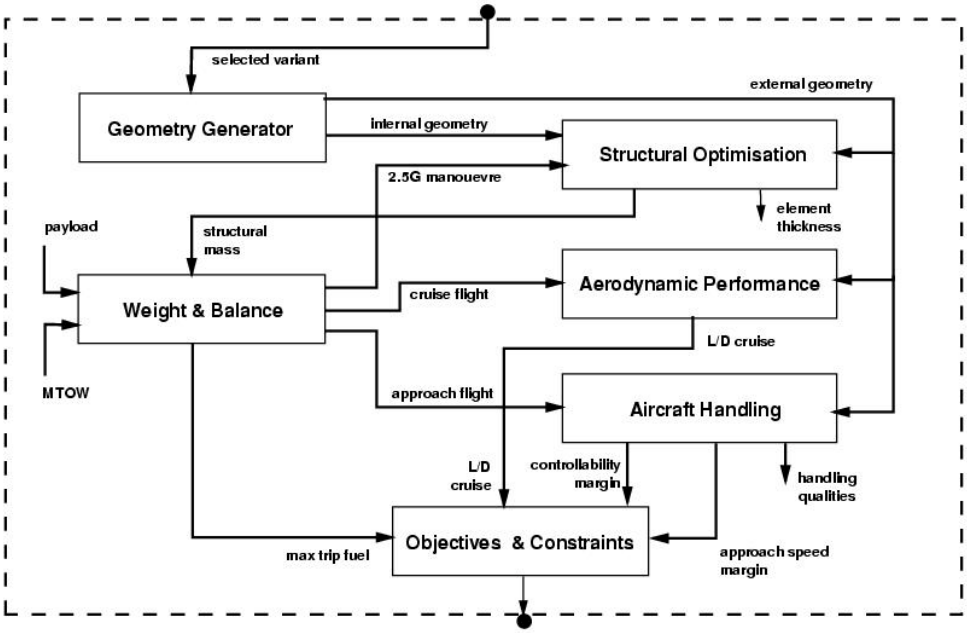

Figure 4: General functional structure of the multi-disciplinary preliminary design process of the BWB in the CDE (from Laban et al.[3])

The actual implementation of the CDE consists of two layers. The first layer consists of a system level implementation of the CDE. This layer is in principle fully functional, and contains basic control and execution functionality directly on the operating system level. The second layer consists of the SPINEware CDE WorkingEnvironment, and can be considered as an additional user-oriented system based on SPINEware functionality. The first layer requires no installation of SPINEware to run the CDE, but does require detailed knowledge of the tools, processes and data management of the CDE, and a good level of Unix experience of the user. The second layer does require the SPINEware software to be operational, but provides the functionality to easily access and navigate through the CDE, and execute (or modify) the tools and analysis processes that are offered as AtomicTool and WorkFlow objects via the SPINEware User Shell.

In the first layer of the CDE all the functional components are stored in a Unix file-system tree-structure. On the top level, the tree has two branches: one for tools and one for data. The tools branch contains the software for all the CDE-specific tools, ordered per discipline. These disciplines more or less correspond to the technical disciplines in the design process, and the disciplines indicated by the following key words (which correspond to the directory names in the tools branch) are considered:

1. geometry (CAD and BWB geometry generation for both aerodynamic and structural mechanic analysis),
2. aerodynamics_and_trim (high- and low-fidelity aerodynamic - and trim analysis),
3. weight_and_balance (static weight and balance evaluation of the BWB),
4. structures (structural mechanical analysis and structural weight optimisation),
5. flight_mechanics (flight mechanics analysis and handling quality evaluation),
6. flutter (flutter analysis),
7. objective (overall design objectives evaluation),
8. response_surface (objectives' response surface calculation for overall optimisation),
9. database (data handling).

In addition to these disciplines there are two main groups of supporting tools, indicated by the following key words:

1. toplevel (CDE process control and data management. Among others, the analysis processes for the full BWB design evaluation and its four main sub-processes (1. geometry generation, 2. weight evaluation, 3. aerodynamic performance, and 4. flight mechanics, objective and constraints evaluation) are defined here. Process control, batch execution and job distribution over a workstation cluster is provided for these processes.)
2. include (include-scripts containing generic functionality that is applied in more than one of the execution scripts of one or more disciplines).

Each discipline contains a directory structure indicated by the following key words:

1. source (containing the source code of the CDE-specific software),
2. compile (containing the compilation procedures for the CDE-specific software),
3. bin (containing the (platform specific) executables of the CDE-specific software, ordered per supported system architecture; currently only SGI-IRIX-6.5, and (partly) Linux 2.2 are supported),
4. run (containing Unix Bourne shell scripts for execution of the analysis processes and for data management within the discipline),
5. setup (containing the "setup information" for each discipline, i.e., the static, common data needed in each BWB variant simulation (e.g., the flow conditions input for the CFD simulations)).

Besides the CDE-specific software, also generic (such as Unix system utilities) and proprietary software (such as in-house developed and licensed software programs) are used in the CDE. The two latter are not included in the CDE software because of license and contractual limitations. A global overview of the tools branch of the first layer of the CDE is given in Figure 5.

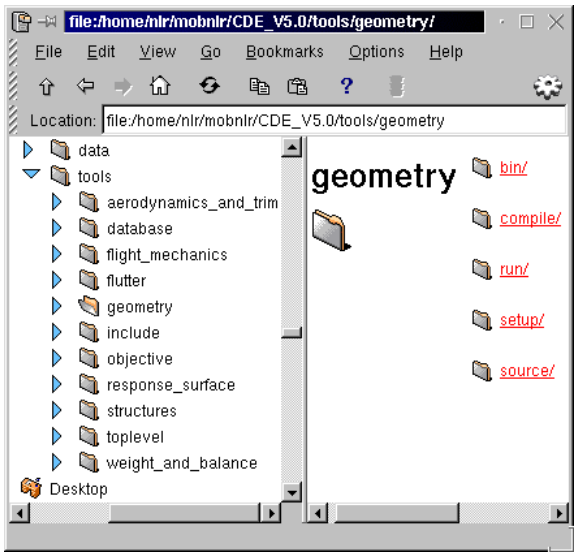

Figure 5: Overview of the tools branch of the first layer (system level) of the CDE presented by a file-system browser, with the disciplines directories on the left, and, as an example, the contents of the geometry discipline, on the right.

The data branch in the CDE contains the data of the analyses of all the disciplines, and a specification of the BWB design run that is considered – the so-called experiment set-up. The data is ordered per discipline, using the same discipline key words as in the tools branch, and the design run specification is stored under the key word Experiment_setup in the file bwb_experiment_setup (Figure 6). Under each discipline directory the data of each individual BWB design instantiation (or BWB variant), which is defined by a fixed set of design parameter values and identified by a unique 2-character string in the file bwb_experiment_setup, is stored under the key word bwb_variant_*ij* , where *ij* represents the unique 2-character string.

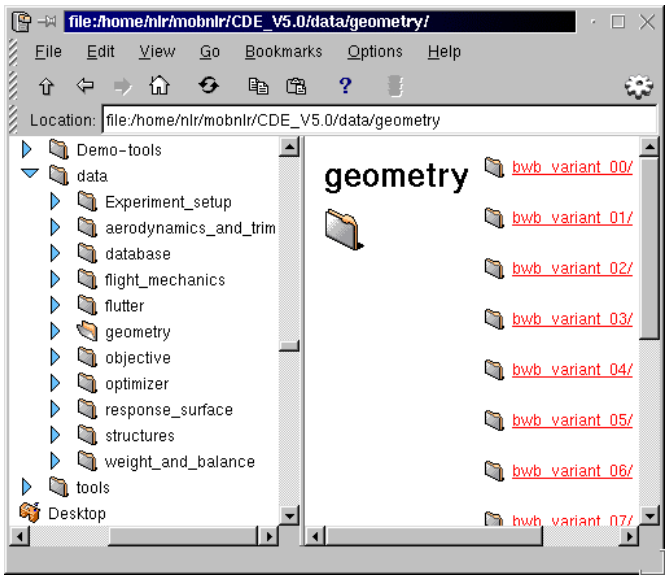

Figure 6: Overview of the data branch of the first layer (system level) of the CDE presented by a file-system browser, with the disciplines' data directories on the left, and, as an example, the contents of the geometry data, on the right.

Data exchange between disciplines and between BWB variants is achieved via symbolic links in the file-system. Global design and optimisation data is stored per discipline and exchanged among disciplines in database files in straight forward ASCII-format, and the main database files are gathered in the database discipline.

A global overview of the contents of the first layer of the CDE is given in Figure 7.

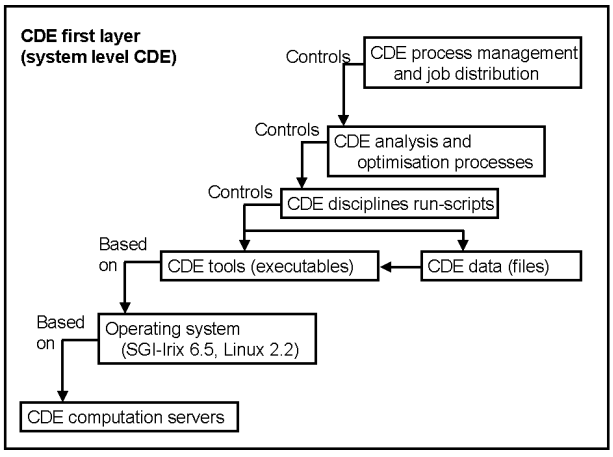

Figure 7: Overview of the structure of the first layer (system level) of the CDE.

The second layer of the CDE is built using the SPINEware system. This SPINEware based CDE consists on the top level of a SPINEware WorkingEnvironment object. The main components

within this CDE WorkingEnvironment are ordered by means of SPINEware ObjectFolder objects, as shown in Figure 8.

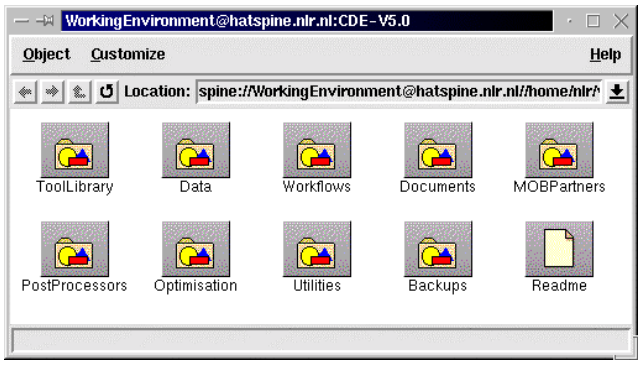

Figure 8: The second layer of the CDE: SPINEware object browser of the top level CDE WorkingEnvironment object, containing the main CDE ObjectFolders.

The SPINEware based CDE makes use of the functionality of the first layer of the CDE, i.e., the system-level CDE. The main ObjectFolder in the CDE WorkingEnvironment is the ToolLibrary ObjectFolder (Figure 9), containing a library of SPINEware based CDE tools.

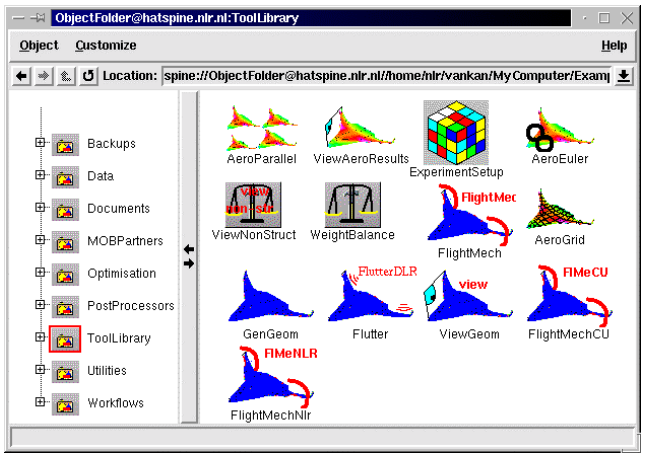

Figure 9: SPINEware object browser of the CDE ToolLibrary ObjectFolder.

In the tool library the SPINEware objects (mainly AtomicTool objects) with the key functionality of this CDE layer are stored. These objects can be considered as the main “building blocks” of the SPINEware CDE. The objects in other ObjectFolders, in particular in the Workflows ObjectFolder, are composed -as far as possible- from these building blocks. Because of obvious software management advantages, this composition is preferably based on links to, rather than copies of, the original objects in the tool library. The tools in the ToolLibrary are more or less directly related to the system level CDE functionality. For

example, some of the tools in the tool library directly control the system level CDE main design sub-processes: geometry generation, weight evaluation, aerodynamic performance, and flight mechanics, objective and constraints evaluation. As another example from the tool library, the ExperimentSetup tool can be mentioned, which directs the user to the Experiment_setup data directory and presents the selected bwb_experiment_setup file in an appropriate editor. As an example from the Workflows ObjectFolder, Figure 10, shows a two-level hierarchical workflow of the CDE analysis process. In this example, the BWB weight evaluation, which is performed by low-fidelity aerodynamic and structural mechanics evaluations, is stored in a separate sub-workflow that is included twice in the main workflow of the whole analysis process for iterative weight calculation.

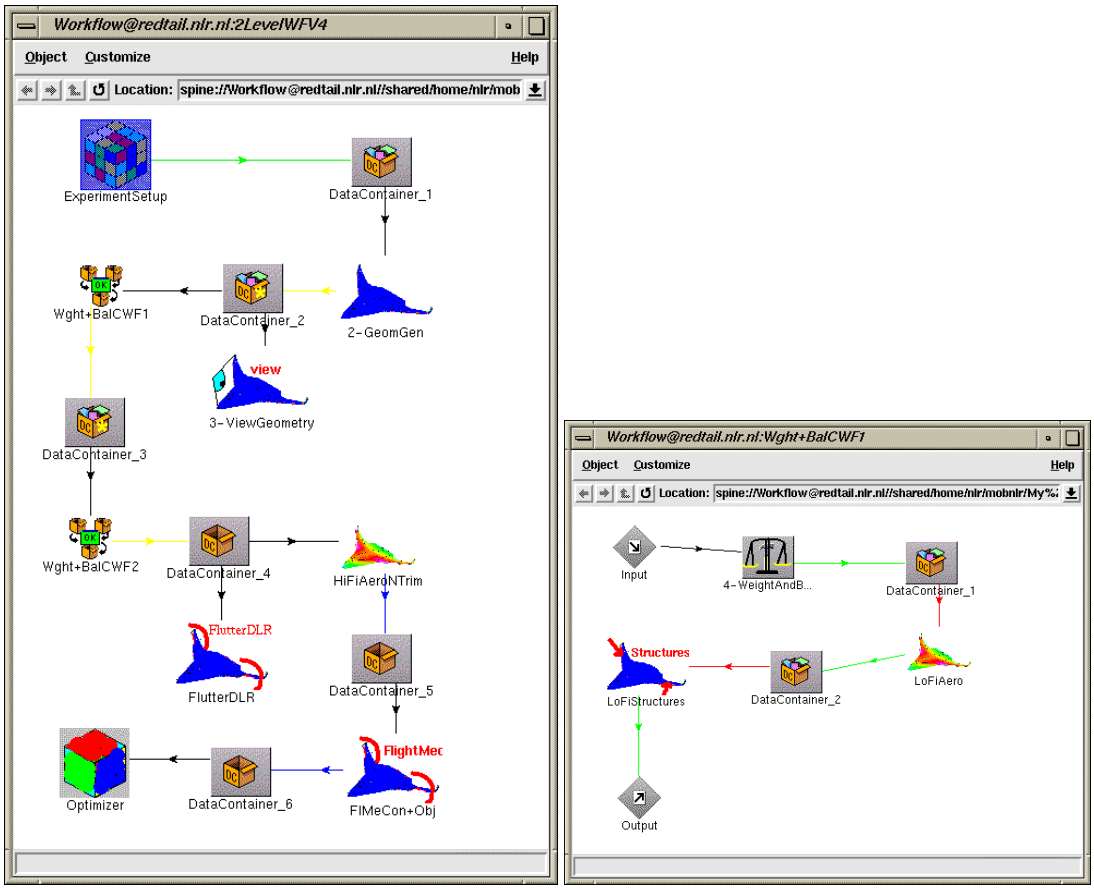

Figure 10: Multi-level workflow of CDE analysis process; the top level workflow with the CDE analysis process is given in the left panel, and the sub-workflow (so-called ChildWorkFlow) with the weight calculation process is shown in the right panel.

The Data ObjectFolder contains links that directly point to the proper CDE data directories (recognised as Directory objects), which are the actual data directories in the system level CDE (Figure 11).

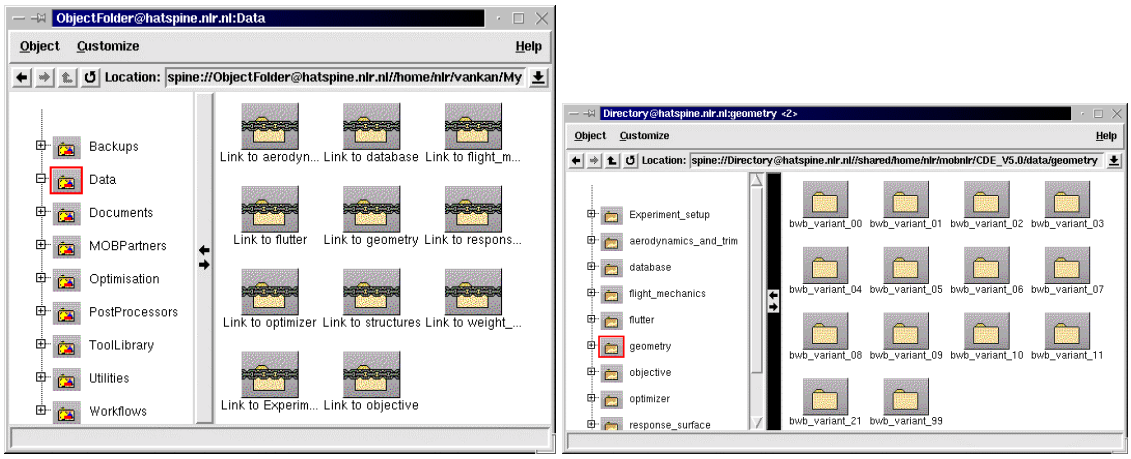

Figure 11: SPINEware object browser of the CDE Data ObjectFolder (left panel), which contains links to all the disciplines' data directories, and, as an example, the contents of the CDE data directory of the geometry discipline (right panel).

Other CDE ObjectFolders, like PostProcessors, Optimisation and Utilities, contain selected tools for their respective purposes. The Documents ObjectFolder contains (links to) the relevant documentation, like MOB documents, papers and manuals. The MOBPartners ObjectFolder contains the information about (and links to) the partners in the MOB project. The Backups ObjectFolder contains backups of CDE components.

A global overview of the contents of the second layer of the CDE and of its relation to the first layer, is given in Figure 12.

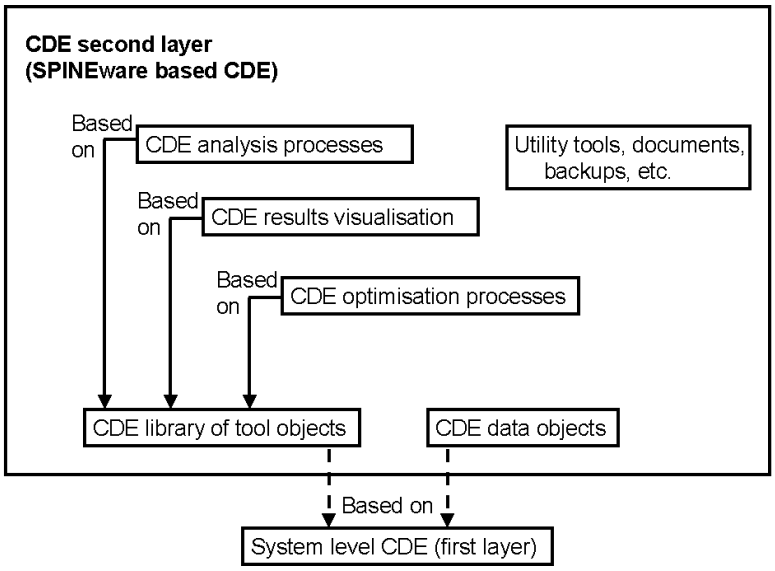

Figure 12: Overview of the structure of the second layer of the CDE (the SPINEware based CDE).

The majority of what has been presented so far about the CDE functionality, has been developed, and is operational, at NLR. As indicated earlier, for certain specific analysis the CDE automatically makes a connection to the MOB partner that provides this particular functionality. In this way, the following connections are currently operational:

1. NLR – TUD for BWB geometry generation,
2. NLR – DLR for flutter analysis,
3. NLR – CU for low fidelity aerodynamic analysis.

These connections are secured (by authentication and encryption) and based on a secure shell (ssh2 [12]) protocol, set up exclusively from the NLR CDE communication server to the MOB partners' respective servers (Figure 13).

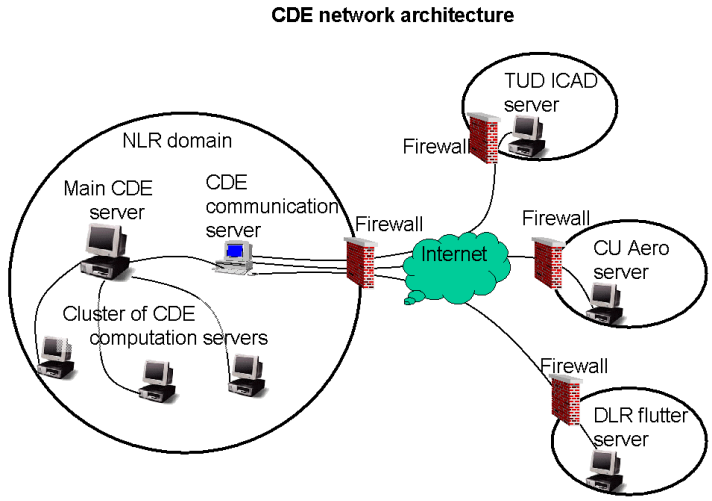

Figure 13: Overview of the current multi-site network architecture of the CDE.

The architecture shown in Figure 13 presents the multi-site situation for the CDE that is presently operational. Other MOB partners that would provide additional CDE functionality could be connected to the CDE similarly to the partners (i.e., tool providers) currently connected.

With respect to the *accessibility* of the CDE, there is the possibility for the MOB partners to access the CDE via the SPINEware HTTP server using a standard web client (e.g., MS Internet Explorer). The CDE tools and data, which are on the NLR file-system server, can be directly accessed. However, for execution of the CDE, access to the NLR CDE server is necessary (Figure 14). This access is normally not available, because CDE server is within the NLR secured domain, but can be made available technically relatively easy.

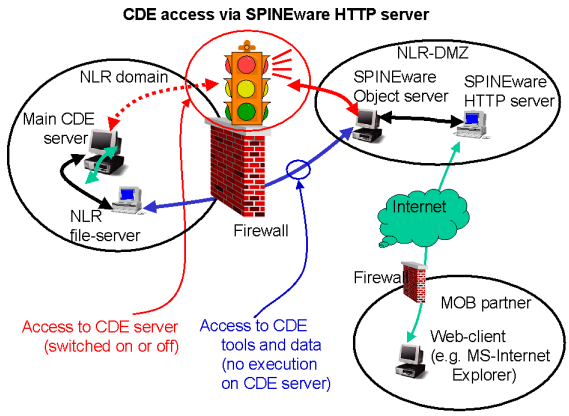

Figure 14: Overview of the accessibility of the CDE using web-based access via the SPINEware HTTP server, which is in the NLR network zone with limited security, the so-called de-militarised zone (DMZ).

4 CDE operation

In order to use the SPINEware CDE, first of all a SPINEware session should be started, either by running locally the SPINEware User Shell and Object Server, or by connecting to a remote Object Server via a web client (see also Figure 1). Note: in both cases the necessary access permissions must be available. It is beyond the scope of this paper to go into further detail of this aspect. An example of access to the CDE from a PC via a Microsoft Internet Explorer web client is illustrated in Figure 15. In this case the SPINEware User Shell is initiated via the HTTP server, and presented to the user via Java applets. In the User Shell, the user may open the CDE WorkingEnvironment by a double mouse click on the CDE icon. Then the CDE WorkingEnvironment will be opened in a browser and its top-level contents will be presented to the user (see also Figure 8). In the CDE WorkingEnvironment a README file can be found (Figure 8), which contains some basic information on installation and usage of the SPINEware CDE and can be opened in an editor by a double mouse click. The ObjectFolders in the top-level of the CDE are all accessible by double mouse clicks.

Figure 15: Illustration of a SPINEware start-up session in order to access the CDE via a Microsoft Internet Explorer web client. From the web-interface server (upper window), the SPINEware session is launched. Subsequently Java-applet windows appear, presenting the session status (middle-left), login dialogue (middle-right) and the toplevel ObjectFolder of user mobnlr (lower window).

The tools in the ToolLibrary ObjectFolder (see also Figure 9) are directly accessible to the user and can be executed by double mouse clicks. For example, the tool ExperimentSetup, which

leads the user to the Experiment_setup data directory and presents the selected bwb_experiment_setup file in an appropriate editor (Figure 16), can be easily found and executed.

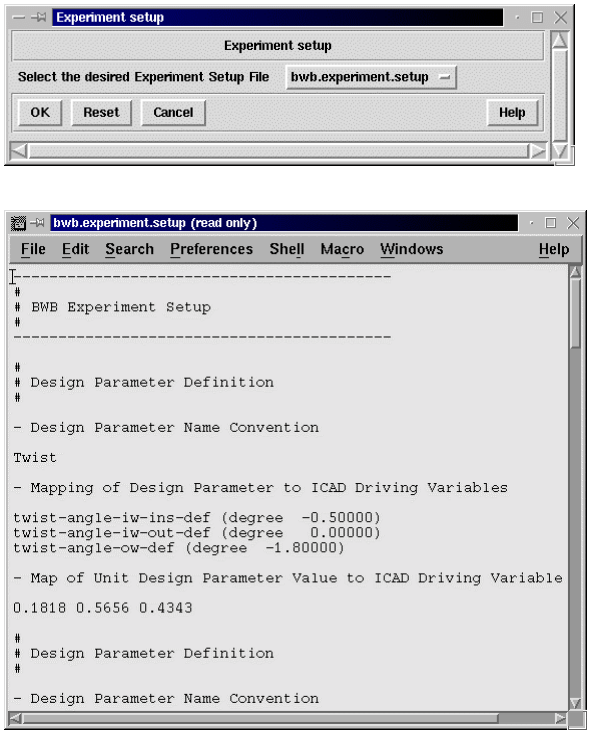

Figure 16: User dialogue window for selection of the bwb_experiment_setup file (upper), and the selected file in an editor window (lower).

In the bwb_experiment_setup file the user specifies (in terms of geometric design parameter values) and identifies (by the unique 2-character string) the BWB variants for which design evaluations are desired (Figure 16). In this way, users are able to rather intuitively find their way to the starting point of a CDE analysis, i.e., via the execution of the ExperimentSetup tool directly from the tool library.

When the desired BWB variants have been defined and identified, the design evaluation process can be performed. In this process, first of all the geometry of the BWB variant is needed. This geometry can be generated by the GenGeom tool, which is also available in the tool library. This tool will request the user to specify a BWB variant identification, and a choice of which geometry data should be generated (surface geometry for aerodynamic analysis, structural geometry for the structural analysis, or both) and will run further automatically (Figure 17). This tool will initiate sub-processes that prepare the input for the CAD program ICAD, transfer

the data to University of Delft where the ICAD process is run, and transfer the results back to the local machine.

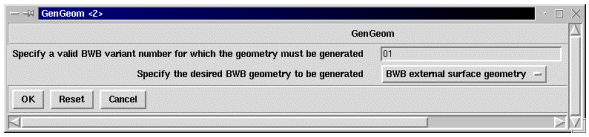

Figure 17: User dialogue window of the GenGeom tool for specification of the BWB variant number (bwb_variant_ij) and selection of the desired type of geometry (external surface, internal structures, or both).

When the geometry generation has been run successfully, the results can be inspected with the tool ViewGeom. This tool requests the user only to select a BWB variant out of a list of available BWB variant geometries and then automatically applies an appropriate visualisation program to the selected data (Figure 18).

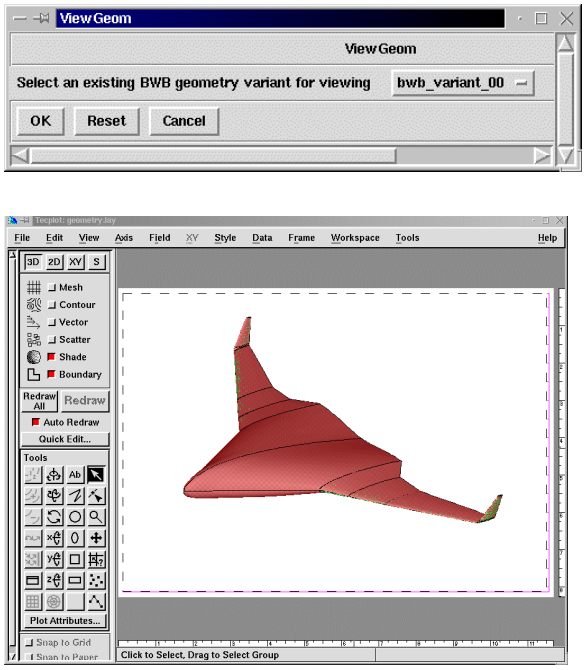

Figure 18: User dialogue window of the ViewGeom tool for selection of the BWB variant number (bwb_variant_ij) (upper), and the resulting BWB geometry visualisation (lower).

Once the geometry data of a BWB variant is available, various other parts of the BWB design evaluation can be executed. Typically the next step would be to perform the weight and balance evaluation. This evaluation is based on estimates of the contributing mass items (structural, non-structural, payload and fuel masses). Initially the correct structural mass of a BWB variant is unknown because this will be computed in a local structural optimisation evaluation. Therefore an initial estimate, based on for instance the reference BWB variant, is used, which will then be

corrected iteratively by subsequent low-fidelity aerodynamic and structural mechanic evaluations. This process is also illustrated in the WorkFlow in Figure 10. The structural mechanic weight optimisation that is performed within this iteration can be performed using a low-fidelity approximation model, or a high-fidelity Finite Element (FE) model including a flutter constraint evaluation with the CDE's flutter discipline [3]. After weight and balance evaluation, high-fidelity aerodynamic analysis of the BWB under transonic cruise flight conditions are performed in order to accurately predict the aerodynamic performance (the so-called lift over drag L/D). This L/D is a key quantity for the determination of the overall design objective, the so-called Bréquet Range [3], which is computed in the objective discipline where use is made of the data gathered in the database discipline. The optimisation of this objective is constrained by the so-called Controllability Margin (CM) of the BWB, which is evaluated by the flight mechanics discipline in the CDE. This very brief description of the BWB design process gives an indication of the order in which the disciplines in the CDE are applied. For a more detailed description of this process we refer to [3].

The discipline that was not yet mentioned in the above process description is the response surface discipline. In this discipline the results of a set of evaluations of BWB variants is collected and further processed to an appropriate polynomial functional representation. For example, a design study has been performed with the CDE by variation of five of the BWB design variables: wing-twist, wing-thickness, wing-sweep, fuselage-length and fuselage-camber. A total set of 51 BWB variants, each with different values for the design parameters, was subjected to the design evaluation process. The resulting values for the design objective and constraint were collected and fitted by polynomial functions with the design parameters as independent variables. The resulting fitted functions give a representation of the dependency of the design objective and constraint on the design parameters. An illustration of the dependence of the design objective (Bréquet Range) on three of the design parameters is given in Figure 19).

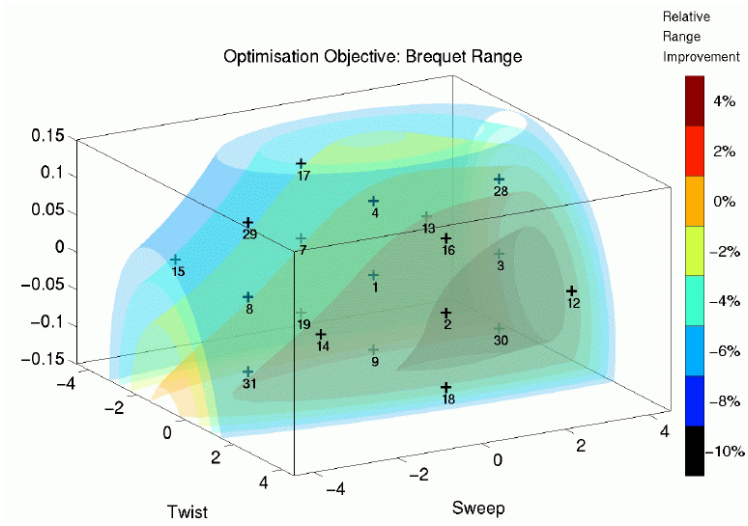

Figure 19: Relative Bréquet Range improvement as a function of the three BWB design variables : wing-twist, wing-sweep and wing-thickness (vertical axis).

This brief description of the application of the CDE and the function of its disciplines illustrates only a small part of the total CDE functionality. It should be noted that besides the execution of the complete BWB design evaluation as one large, more or less automatic process, it is also possible, under certain conditions of availability of required data, to run different sub processes for specific analyses of the BWB variant. This flexible functionality is under further development, together with further application of the CDE to other BWB design studies.

5 Conclusions

The Computational Design Engine that is being developed in the MOB project is built up from the components that are provided by the MOB partners. These components, mainly software tools for design analysis, data conversion and exchange, are first developed and implemented on system level. While maintaining as much as possible functionality on this level, another layer with user oriented functionality is built on top of it. This second layer is based on the SPINEware middleware system, which directly provides the functionality for a user oriented, object based and network transparent approach. Via the SPINEware User Shell, the CDE is presented graphically to the user, either directly on the user's local system, or via a web client-server approach.

The CDE contains and manages the tools, data, documents, and other related information and utilities for the design and analysis of BWB aircraft. Several automatic design processes, based

on different sequences of analysis runs using the CDE components, are directly available to the user. Further enhancements to the functionality of the CDE, such as more flexibility in concatenating tools into workflows, are envisaged as extensions of the current functionality.
The CDE has been applied successfully to a design and optimisation study of the BWB aircraft. It has shown a typical applicability to evaluation of feasible design in the early design stage. Future work will involve application of the CDE in further enhanced BWB design evaluation studies.

Acknowledgements

European Community financial support for the present study through the European Commission GROWTH Programme, Research Project *MOB - A Computational Design Engine Incorporating Multidisciplinary Design and Optimisation for Blended Wing Body Configuration*, is gratefully acknowledged. In addition, the collaborative contributions and valuable recommendations of Mr. P. Arendsen of the NLR Structures and Materials Division and Mr. W. Rouwhorst of the NLR Flight Division are acknowledged.

6 References

[1] MOB: A Computational Design Engine Incorporating Multidisciplinary Design and Optimisation for Blended Wing Body Configuration, EC 5th Framework Programme, Contract Number: GRD1-1999-11162, 2000.

[2] G .La Rocca,.L. Krakers, M. van Tooren: Development of an ICAD generative model for blended wing body aircraft, 9th AIAA/ISSMO Conference, Atlanta, GA, USA, September 2002. AIAA-2002-5447.

[3] M. Laban, P. Arendsen, W.F.J.A. Rouwhorst, W.J. Vankan: A computational design engine for multi-disciplinary optimisation with application to blended wing body configuration, 9th AIAA/ISSMO Conference, Atlanta, GA, USA, September 2002. AIAA-2002-5446.

[4] M. Stettner, R. Voss: Aeroelastic, flight-mechanic, and handling qualities of the MOB BWB configuration, 9th AIAA/ISSMO Conference, Atlanta, GA, USA, September 2002. AIAA-2002-5449.

[5] N. Qin: Aerodynamic studies of blended wing body aircraft, 9th AIAA/ISSMO Conference, Atlanta, GA, USA, September 2002. AIAA-2002-5448.

[6] P. Bartholomew: The development of the MOB data and product management system, 9th AIAA/ISSMO Conference, Atlanta, GA, USA, September 2002. AIAA-2002-5497.

[7] A.M.J. Morris: MOB – A European distributed multi-disciplinary design and optimisation project, 9th AIAA/ISSMO Conference, Atlanta, GA, USA, September 2002. AIAA-2002-5444.

[8] W.J. Vankan, R. Maas, M. ten Dam: ICT Environment for multi-disciplinary design and multi-objective optimisation: a case study, *Proc. ICCS 2002 conference*, pp. 663-672, 2002.

[9] E.H. Baalbergen, H. van der Ven: SPINEware – a framework for user-oriented and tailorable metacomputers, *Future Generation Computer Systems*, 15, pp. 549-558, 1999.

[10] B.C. Schultheiss, E.H. Baalbergen: Utilizing supercomputer power from your desktop, HPCN 2001 conference, Amsterdam, 2001.

[11] **http://www.spineware.com**

[12] **http://www.ssh.org**