A Systems Approach to Model the Conceptual Design Process of Vertical Take-off Unmanned Aerial Vehicle

A thesis submitted in fulfilment of the requirements for
the degree of **Doctor of Philosophy**

Ankush Rathore
B.Eng.

The Sir Lawrence Wackett Centre for Aerospace Design Technology
School of Aerospace, Mechanical and Manufacturing Engineering
Science, Engineering and Technology Portfolio
RMIT University
March 2006

ACKNOWLEDGMENTS

I acknowledge and thank the assistance, efforts and support of the following:

- **Senior supervisor** *Dr. Arvind Kumar Sinha*, Director, The Sir Lawrence Wackett Centre for Aerospace Design Technology, RMIT, for his encouragement, guidance, and support throughout my doctoral candidature including the committed effort in preparing my thesis through major comments on a daily basis over ten months;
- **Consultant** *A/Prof. Peter Hoffman*, School of Aerospace, Mechanical and Manufacturing Engineering for his encouragement and advice to achieve the research objectives; and
- My **parents** for their confidence and financial support throughout my doctoral candidature.

Ankush Rathore

17 March 2006

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

SUMMARY

The development and induction in-service of Unmanned Air Vehicles (UAV) systems in a variety of civil, paramilitary and military roles have proven valuable on high-risk missions. These UAVs based on fixed wing configuration concept have demonstrated their operational effectiveness in recent operations. New UAVs based on rotary wing configuration concept have received major attention worldwide, with major resources committed for its research and development.

In this thesis, the design process of a rotary-wing aircraft was re-visualised from an unmanned perspective to address the requirements of rotary-wing UAVs – Vertical Take-off UAVs (VTUAV). It investigates the conventional helicopter design methodology for application in UAV design. It further develops a modified design process for VTUAV addressing the requirements of unmanned missions by providing remote command-and-control capabilities. The modified design methodology is automated to address the complex design evaluations and optimisation process.

An illustration of the automated design process developed for VTUAVs is provided through a series of inputs of the requirements and specifications, resulting in an output of a proposed VTUAV design configuration for “design decision support”.

The VTUAV automated design process has been developed to pioneer an aerospace design tool for further detailed development and application as a – Design Decision Support System.

1 INTRODUCTION

Aerospace industries around the globe have invested major resources for the design and development of Unmanned Aerial Vehicles (UAV) as a viable alternative to the manned aircraft, on various high risk missions. UAVs are emerging as the next generation of airborne reconnaissance assets due to their ability to penetrate deep into enemy airspace, loiter on designated target areas, detect lethal weapons and conduct damage assessments. They are the ideal aerial platforms for emerging warfare concepts of 21st century. Presently, the major market of UAVs is military. It comprises of strategic and tactical UAVs. The civil market is still under development with research and development focussed on surveillance and communication tasks.

The major developments of UAVs to-date have been Fixed-Wing (FW) focussed. FW-UAVs have undergone major operational trials and are in various stages of induction and utilisation. The present inventory of FW-UAVs addresses the military (strategic & tactical) and civil (Relief, monitor, & research) operational needs. In addition to FW-UAVs the missions envisaged for UAVs require vertical take-off and landing capabilities – hence the requirement of a Rotary-Wing (RW) UAV. The development of RW-UAVs has gathered momentum to fill-in the operational gaps of FW-UAVs, and it is envisaged that it will play an increasingly important role in both military and civilian missions. This is due to their inherent take-off and landing capabilities that allow operation from unprepared sites, with no requirements of runways, net, launch rail or parachutes. Further, its ability to hover and manoeuvre in limited spaces makes them suitable for operation in restricted or urban terrain, beyond the operational capabilities of FW-UAVs. In addition, to their small size, the viability of detection and destruction is low. Industrial programs have been initiated for the research and development of RW-UAVs, also referred as “Vertical Takeoff UAVs” (VTUAVs). The present inventory of VTUAVs only addresses close-to-ground reconnaissance and surveillance requirements. Of late, the operational need for combat support is also being considered.

Various concepts, beyond the conventional helicopter (Fuselage, and main & tail rotor system) are being proposed and include rocket and magnetic propulsion take-off. Jet-lift principle is another concept for addressing vertical take-off issues. These concepts have not gathered much support; making the conventional helicopter concept as the base-line design concept for the development of VTUAVs.

The rotary-wing design methodology has been traditionally applied for the conceptual design of helicopters. The methodology is based on a series of estimations and evaluations of weight, power, speed, and drag for sizing the helicopter. The computations refers to past helicopter design data and statistics to provide the design benchmarks for the development of a variable conceptual configuration to meet the slated design specifications.

The design methodology is analytical and requires major iterations for the design optimisation; hence it is time intensive. To address the optimisation process more efficiently an automated design process is needed. The traditional design methodology needs to incorporate an autonomous component by the development of an expert system to generate the conceptual design configuration of the VTUAV from the slated mission requirements. The expert system automating the conceptual design methodology needs to display the conceptual design configuration on a “graphical user interface”.

In this thesis, a systems approach is adopted for an insight of the VTUAV design methodology from a systems perspective. It involves a re-visit to the helicopter design methodology from an autonomous perspective for the development of VTUAV system architecture followed by a design methodology. An expert system is developed to automate the design of the VTUAV. The research framework adopted in this thesis for the development of an expert system to address the automation of the VTUAV design methodology is illustrated in Fig 1.1.

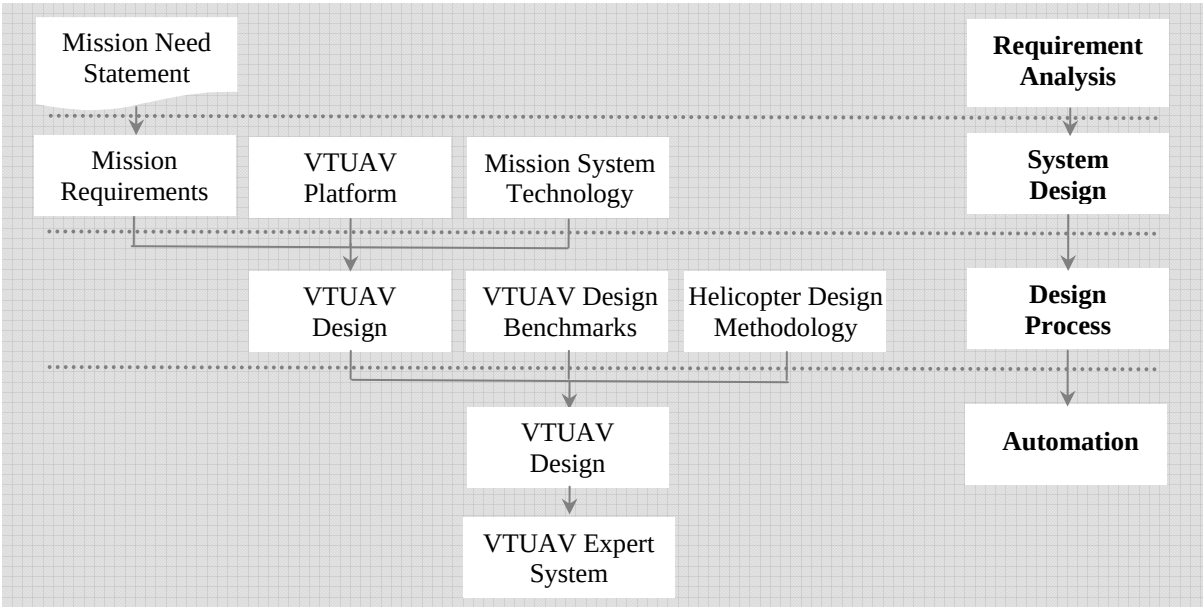


Figure 1.1: Research Framework to Develop an Expert System to address the Automation of the VTUAV Design Methodology

2 LITERATURE SURVEY

2.1 Systems

System is defined as “A purposeful collection of inter-related components working together towards some common objective” [1, 2]. It can also be defined as an assemblage or combination of elements or parts forming a complex or unitary whole. Systems are as pervasive as the universe in which we live. At one extreme they are as grand as the universe itself and at the other end they are as infinitesimal as an atom. Systems first appeared in ‘Natural Forms’, but now also exist in man-made forms [3]. A system may include software, mechanical, electrical and electronic hardware and may be operated by people [2]. In aircraft industry, the term ‘System’ is normally used for electrical systems, hydraulic systems, power-plant systems etc.

Systems have a structure that is defined by its parts (components) and processes [7]. Behaviour of system components is inextricably intermingled. The inter-dependence of the components governs the system properties [1]. The integration of the components into a system provides the base for its assessment and measurement. The system properties generated through this integration are known as emergent properties and depend on the relationships between system components. The overall weight of the system, its reliability and the usage are illustrations of systems emergent properties. These properties are classified as follows [2]:

- Functional: These properties are derived when all the parts of a system work together to achieve a slated objective; and
- Non-functional: These properties are governed by the behaviour of the system in its operational environment. They are often critical for computer-based systems.

The various parts of a system have functional as well as structural relationships between each other. The fact that functional relationship exists between the parts suggests the flow and transfer of some type of material. This involves the inputs and outputs of material (energy and/or matter) that is then processed causing it to change in some way. Systems often exchange materials beyond their defined boundary with the outside environment, and other systems, through various input and output processes as a system configuration [3].

2.1.1 System Composition

The system comprises of the following [3]:

- Elements: These are the operating parts of a system. Components may be atoms, molecules, or larger bodies of matter like sand grains, rain drops, plants, animals, etc;
- Attributes: These are the characteristics of the elements that may be perceived and measured like quantity, size, color, volume, temperature, and mass; and
- Relationships: These are the associations that occur between elements and their attributes. These are based on causes and effects. To identify the intra and inter relationships between the elements and attributes, a relationship matrix needs to be developed [4-6].

The system element may be a system in itself and every system may be a part of a larger system. This is known as a system hierarchy [3].

2.1.2 System Classification

Systems are classified based on their size and activity. The classification of a system is as follows [3, 7]:

- Simple system: A system with less number of elements, and few but organised interactions;
- Complex system: A system with more number of elements, and many but loose interactions;
- Static system: A system with a structure but no activity;
- Dynamic system: A system that combines structural components with activity;
- Isolated system: A system that has no interactions beyond its boundary;
- Closed system: A system that transfers energy, but not matter, across its boundary to the surrounding environment;
- Open system: A system in which both matter and energy can cross system boundary to the surrounding environment;
- Morphological system: A system where the relationships between elements and their attributes are in a indistinct sense based only on measured features or correlations, the processes of which are unknown;
- Cascading system: A system in which the flow of energy and/or matter from one element to another and the processes that cause this movement is known. In a cascading system the quantitative relationships that exist between elements related to the transfer of energy and/or matter are unknown;

- Process-Response system: A system that integrates the characteristics of both morphological and cascading systems;
- Control system: A system that can be intelligently manipulated by the action of humans; and
- Eco-system: A system that models relationships and interactions between the various biotic and abiotic components making up a community or organisms and their surrounding physical environment.

2.1.3 Thinking Modes: Pre-systems and Systems

As the system analysis needed widening to understand the complex behaviour of systems, a new mode of thinking emerged – System thinking [8]. It addressed complex problems and provided an understanding of its properties and functions. It identified that the systems had emergent properties peculiar to themselves that could not be derived directly from parts. These emergent properties emanated from the synergy of the individual parts working together. Pre-system thinking comprised of understanding the system as an aggregation of parts in which the whole is exactly equal to sum of individual parts.

System thinking has its foundation in the field of System dynamics, founded in 1956 by MIT Professor Jay Forrester [8]. Forrester recognised the need for a better way of testing new ideas about social systems, including engineering. System thinking ponders on an understanding of explicit systems and improves their understanding of principles and functions [8].

System thinking is fundamentally different from traditional/pre-system analysis. Traditional analysis focuses on separating the individual pieces for study. System thinking distinctly focuses on how the things being studied interact with the other constituents of the system of which it is a part. System thinking is effective in the analysis of complex problems [8, 9].

System thinking encompasses two pairs of core concerns [10]. The first is emergence and hierarchy. The characteristic of hierarchy reflects the way in which complex systems are organised from smaller, less complex ‘sub systems’. When all of its ‘sub systems’ are working together in an integrated coordinated fashion, the system exhibits emergent properties-synergy. The second pair is communication and control. Control is regulation of activities between subsystems. In all cases this regulation is based upon needs for the entire system, which pass up and down the system’s hierarchical structure. The interface is the point

at which control is conducted. Communication is the means by which controlling information is transmitted between subsystems [11].

Systems thinking comprises of various steps in analysing a system. The analysis is in two modes as follows: a) Analytic thinking - An 'Outside-in-thinking'; b) Synthetic thinking - An 'Inside-out-thinking'.

- Step 1: In analytic thinking the entity to be explained is taken apart and in synthetic thinking it is taken as part of a larger system;
- Step 2: In analytic thinking the contained parts are explained and in synthetic thinking the containing system is explained; and
- Step 3: In analytic thinking an effort is made to aggregate understanding of the parts into an understanding of the whole and in synthetic thinking the understanding of the containing whole is disaggregated to explain the parts by revealing their role or function in that whole.

Synthetic thinking fundamentally is based on two major ideologies: a) Holism; and b) Reductionism [12]. Holism propagates the idea that the whole is more than the sum of the parts. Reductionism can be defined as the belief that the behaviour of a whole or system is completely determined by the behaviour of the parts, elements or subsystems. Downward causation, a third ideology is a converse of reductionism to formulate a clear systemic stance without lapsing into the extremes of reductionism or holism [13].

2.1.4 Analytic & Systems Approach

Analytic thinking when applied to systems problems is called 'Analytic Approach' and synthetic thinking when applied to address systems problem is called 'Systems Approach'. The key characteristics of the two approaches are presented at Table 2.1.

Table 2.1: Analytic and Systems Approaches
(Rosnay, 1997)

Analytic Approach	Systems Approach
• Isolates and concentrates on various elements.	• Unifies and concentrates on the interaction between elements.
• Studies the nature of interaction	• Studies the effects of interactions
• Emphasizes detail precision	• Emphasizes global perception
• Modifies one variable at a time	• Modifies group of variables simultaneously
• Validates facts by means of experimental proof within the body of a theory	• Validates facts through comparison of the behavior of the model with reality
• Uses precise and detailed models that are less useful in actual operation	• Uses models that are insufficiently rigorous to be used as bases of knowledge but are useful in decision and action
• Has an efficient approach when interactions are linear and weak	• Has an efficient approach when interactions are nonlinear and strong
• Leads to discipline-oriented education	• Leads to multidisciplinary education

In software development applications, systems approach combines the elements of software quality assurance (testing, inspections, walkthroughs, measurements and process improvement) with the elements of operational environment (people, process and product) [3, 9, 14].

Based on 'systems thinking' there are several system approaches available to address system problems [15]. The various approaches and its characteristics are as follows:

- Systems dynamics: The structure and its processes are analysed to model real world systems. The approach is applicable for systems that are well-developed and are measurable;
- Viable system diagnosis: The organisation and the processes are analysed to model real world systems. The approach assumes organisational character of a system being evolutionary in nature. Modern organisations are the main clientele of this approach;
- Strategic assumption surfacing and testing: This approach is applicable for 'ill-structured' systems. It focuses on the relationships among the participants in a system rather than the structure of system;
- Interactive planning: This approach assists participants of an organisation in designing a joint future and creating ways to implement these designs. It is largely used as a planning function in an organisation and takes a cultural perspective of systems;
- Soft systems methodology: This pluralistic approach focuses on the values, interests and beliefs of participants in a system to address ill-defined problems. It is applicable on systems problems with a multidimensional culture;
- Critical systems heuristics: In this approach, power and coercion are key characteristics. It is used to identify and describe the underlying value assumptions that enter into planning and systems design. It is applicable for design of large systems; and
- Systems engineering: It is an interdisciplinary approach that addresses customer's needs throughout the system's life cycle [16]. The approach transfers the operational needs stated by the customer into system performance parameters and preferred system configuration by: a) Functional analysis; b) Optimisation; and c) Design, test and evaluation.

Flood and Jackson attempted to map these system approaches into a two dimensional space – 'Total Systems Intervention' (TSI) [17, 18]. The TSI is a problem solving procedure to encourage creative thinking about organisations and their issues. The TSI is open ended; and based on specific problem context it tabulates a group of system approaches as system

methodologies. The grouping involves the analysis of the system in the following categories: a) Simple - small number of elements, few interactions, predetermined attributes, and highly organised interactions amongst system elements; b) Complex - large number of elements, many interactions, not predetermined attributes and loosely organised interactions amongst system elements; c) Unitary - unified goal, and strong coalition amongst system elements; d) Pluralist - individual goal, and loose coalition amongst system elements; e) Coercive - individual goal, and no coalition amongst system elements. The grouping of the system approaches based on the system categories is presented in Table 2.2

Table 2.2: Grouping of System Methodologies based on Problem Contexts

(Flood and Jackson, 1991)

	Unitary	**Pluralist**	**Coercive**
Simple	Operations research Systems analysis Systems engineering Systems dynamics	Social systems design Strategic assumption surfacing & testing	Critical systems heuristics
Complex	Viable system diagnosis General system theory Socio-technical systems thinking Contingency theory	Interactive planning Soft systems methodology	-

System engineering is the most widely used system approach. The system engineering process involves an analysis of requirements & functions, synthesis, and verification of the design solution.

2.1.5 Systems Engineering Process

The systems engineering process (SEP) is built on an iterative, top-down, hierarchical decomposition of system requirements supported by trade studies that record the origin for significant decisions and considered preferences [19]. The iterative decomposition process levels are the following: a) System level; b) Subsystem level; c) Functional subsystem level; and d) Hardware/Software configuration level. At each level the activities involved are functional analysis, synthesis, and verification [20]. Depending on the system, it may be necessary to address special requirements. The sequence of activities is presented in Fig 2.1 and the details are as follows:

- Requirements analysis: A requirement is a statement of required performance or design constraint to which a product must confirm. The requirement analysis is an integral part of SEP. It involves appropriate allocation of functional requirements to the system elements that are verifiable. This analysis either verifies the validity of existing requirements or

develops new requirements which are more appropriate for the designated mission. It is to be reconfirmed by the customer;

- Functional analysis: A function is defined as a task, action or activity performed to achieve a desired outcome. Functional analysis establishes performance requirements, produces functional hazard assessment for certification, and defines limit of each function (time, scope and activities). When time is critically associated in establishing the performance, timeline analysis is also addressed;
- Synthesis process: Synthesis involves the design of the system architecture to address performance and design requirements [15, 20]. It involves the development of block and flow diagrams by appropriate engineering specialists. These diagrams represent the baseline design, with techniques for its test, support, and operation. It also identifies the interfaces and measures for traceability and changes [21];
- Verification process: Verification evaluates the degree to which the slated requirement specifications have been met. It involves the following [19]:
  - *System engineering validation*: It is a verification of requirements by inspection, analysis, demonstration, logical argument, modeling or simulation.
  - *Test engineering validation*: It involves hardware and software testing. Tests are conducted to verify requirements using final design hardware configuration and software configuration [22]. The testing of software configuration comprises of unit and functional testing and addresses it from a programmers and user's perspective respectively [23, 24]; and
- Speciality requirements: They address the intermittent customary requirements and are derived from system performances and constraints. These requirements warrant attention of engineering and operational specialists from a regulatory, environmental, and design perspective.

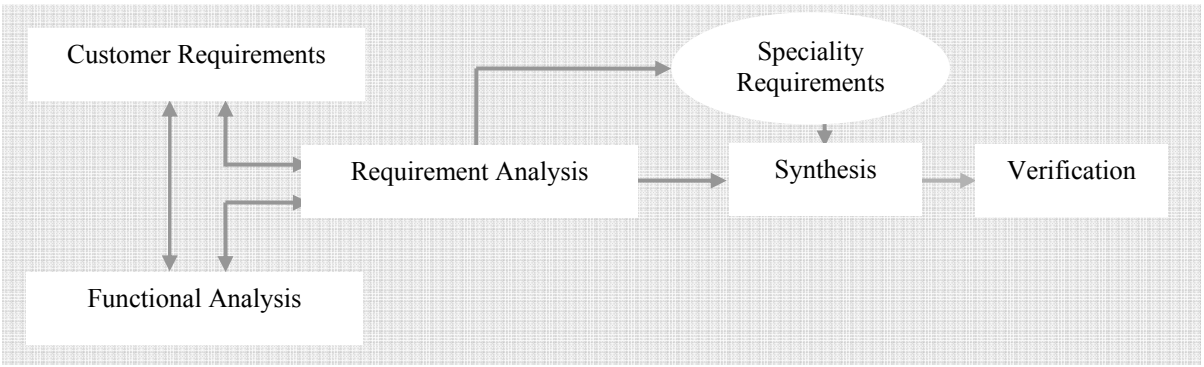


Figure 2.1: System Engineering Activities

Somerville [2] addresses the SEP from a design, development, installation and decommissioning perspective. The SEP thus comprises of the following (Fig 2.2):

- Requirement definition: It is an outcome of detailed requirement analysis. The properties of the requirements are classified as - main functional, emergent, and impermissible / undesirable;
- System design: It involves comprehensive system development. It begins with external system design and then proceeds to various internal sub-system levels. The major sequential activities are the following: a) Partition of requirements; b) Identification of sub-systems & assignment of individual requirements; and c) Determination of functions & assignment of limits;
- Sub-system development: This activity involves development of each sub-system, identified during system design. It may result in the application of SEP for each individual sub-system;
- System integration: It is an assimilation process of identified sub-systems into a main system. It comprises of the following: a) Big bang process involving assimilation of all sub-systems together; and b) Incremental process involving sequential assimilation of sub-systems;
- System installation: The process involves laying of developed system into its working environment; and
- System evolution and decommissioning: The process results in identification of limitations that needs to be addressed to evolve a 'legacy system' which is decommissioned at an appropriate time.

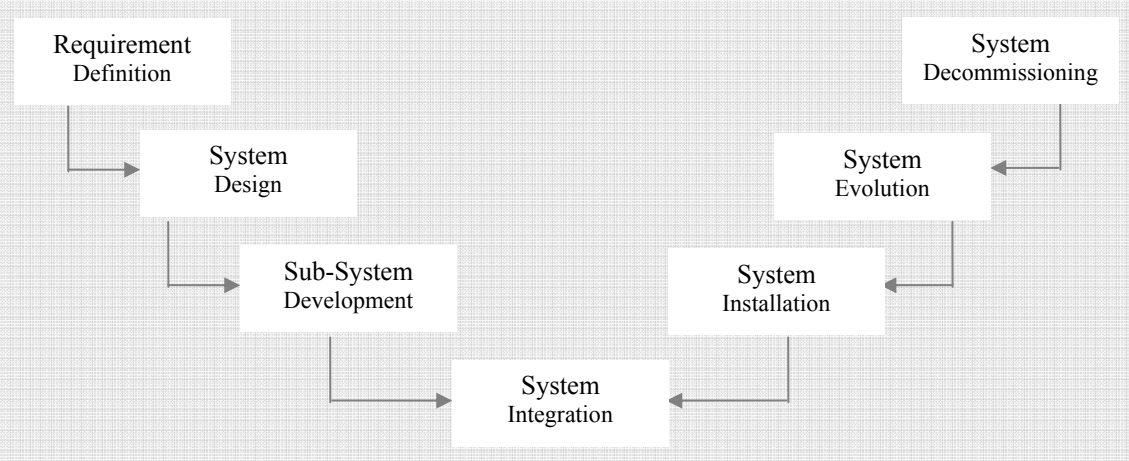


Figure 2.2: System Engineering Process

(Somerville, 1996)

There are several SEP standards and models that contain regulations for identification of vital processes and activities in conjunction with risk assessment. Some of the key SEP standards and models are the following [25-29]:

- ISO/IEC 15288, Systems Engineering-System Life Cycle Processes: The standard addresses life cycle of systems including development, operation, and support through a universal enterprise, project and various technical processes;
- ANSI/EIA 632, Processes for Engineering a System: The standard defines a system as constituent of operational and enabling products;
- IEEE 1220, Application and Management of the Systems Engineering Process: The standard defines the interdisciplinary tasks that are required throughout a system's life cycle to transform customer needs, requirements, and constraints into a system solution;
- EIA 731, Systems Engineering Capability Model: The standard is designed to measure and improve capability of an organisation's systems engineering process; and
- CMMI SWE/SE/IPPD/SS, Capability Maturity Model: The model integrates software engineering, systems engineering, integrated product & its development, and supplier sourcing.

2.1.6 System Life Cycle

The system life cycle [30, 31] comprises of two phases as follows: a) Acquisition; and b) Utilisation. The acquisition phase involves the design, manufacture and production of the system. The utilisation phase is the actual usage of the system and its logistic support while in service. The design, manufacture, production, usage and support are synergistically integrated [32]. The system-life-cycle is presented in Fig 2.3 [17].

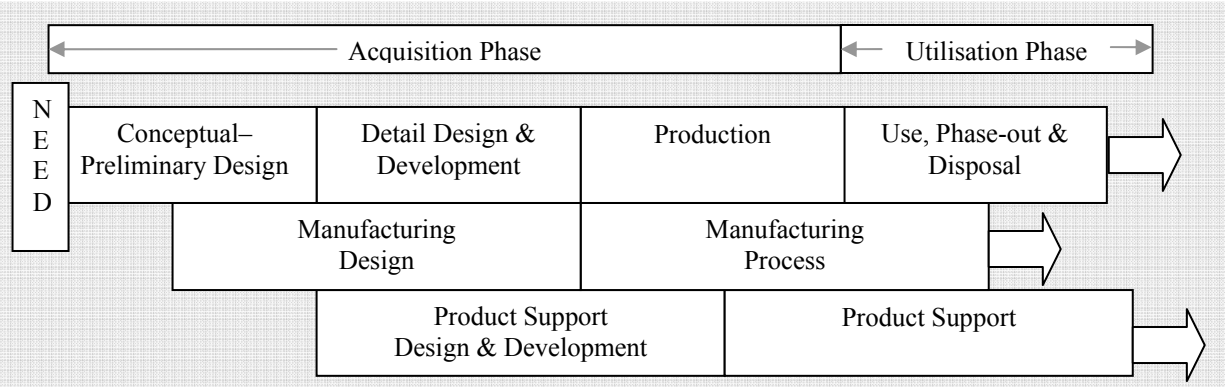


Figure 2.3: System Life Cycle

(Flood & Jackson, 1991)

2.1.7 System Engineering in Aircraft Design

Aircraft design comprises of the development of new, upgraded or derivative aircraft. Each of these designs applies system engineering process. The design of new aircraft involves a complete SEP starting from a new set of requirements. In upgraded design, additional requirements are developed and verified within specified constraints. The design of a derivative aircraft involves the re-development of the aircraft subsystems in coherence with the modified requirements [33].

Aircraft's lifecycle are a set of activities performed by the developer and users of the aircraft system and comprises of marketing & design development; manufacturing, assembling & certification; operation & sustenance; and decommissioning [34]. SEP for aircraft involves functional & requirement analysis, speciality requirements, synthesis and verification.

Functional analysis in an aircraft's life cycle covers the following: a) Vehicle function – Design activity including aerodynamics, propulsion, navigation and communication; b) Operational function – Pre-flight to post-flight including emergency; and c) System function – Maintenance and training activities.

Requirements analysis categorises aircraft's requirements as follows: a) Performance and support to address the acceptable operational, financial, & logistic needs; and b) Constraints that are driven by design regulations and design economics. Design regulations involve safety, functional, operational, performance, and maintenance certifications. Design economics is based on net projected life cycle costing.

Speciality aircraft requirements warrant attention of engineering and operational specialists to address critical regulatory and environmental requirements. Synthesis process documents the system architecture to specify the design requirements and performance for detailed design. Verification involves ground tests, simulations, analysis, examinations, and flight tests to validate the requirements.

2.2 Unmanned Aerial Vehicles

2.2.1 Chronological Development

The budget for research and development of UAV systems to meet the growing needs of future has increased worldwide [35-36]. The growth in dangerous aerial missions due to unconventional warfare has resulted in a concerted effort worldwide on the development of unmanned technology. Unmanned Aerial Vehicles (UAVs) were extensively used in Vietnam war (1959-1975) for combat reconnaissance. Firebee, the first reconnaissance UAV, was packed with cameras for spying and scouting armed flights during the war. During the Gulf war in early 90's UAV technology matured for key application in military operation to gain strategical and tactical advantage. At present, the world inventory comprises of approximately 225 UAVs of all types either in usage or under development. Global Hawk, a high altitude long endurance UAV, was first used in Kosovo operations to spot potential targets. Similarly Predator was deployed for reconnaissance and surveillance operations in Bosnia. Predator was also used extensively to support assault in Afghanistan. The Desert Hawk is presently used in Iraq and Afghanistan and the Pioneer UAV was deployed in a crucial role to spot battleships [37].

2.2.2 Future Developments

The UAV developments have mainly been with a military application focus. This trend will continue due to its inherent capabilities – a key requirement in futuristic warfare and includes tactical and strategic UAVs. The civil UAV market, due to restricted research funds, is still immature and the development has been for surveillance and telecommunication applications. Fixed-Wing Unmanned Aerial Vehicles (FWUAVs) have undergone major operational trials and are in various stages of induction and utilisation. Rotary-wing UAVs also referred as "Vertical Takeoff UAVs" (VTUAVs) are presently in advanced research and development stages, to fill the operational gaps of FWUAV deployment. The future technological developments will address several issues as follows [38]:

- Fuel cells supplantation of internal combustion engines to reduce audio signalling of UAVs;
- Inherent self-repairing & damage redundancy to enhance survivability;
- Efficient turbine engines to provide higher endurance for reconnaissance;
- Onboard mission systems providing GPS pseudo-satellites & airborne communications nodes for tactical application;
- Provision for precise terrain mapping to support cruise missile targeting; and

- Combat capability through weapon payload, secure communication, high manoeuvrability and usage of special materials to reduce combat damage.

2.2.3 Vertical Takeoff UAV

Vertical Takeoff UAVs (VTUAVs) have a distinct advantage over FWUAVs due to their inherent takeoff and landing capabilities from any terrain. They can operate from unprepared sites and ships [39-41]. No runways, nets, launch rails, or parachutes are required for the launch or recovery of VTUAVs; hence they are ideal for urban warfare. The envisaged roles for VTUAVs are reconnaissance, surveillance, targeting and intelligence gathering. In future it may encompass search & rescue, data relay, and battle damage assessment including its usage as force multipliers [42].

Sinha et al. [32] analysed comprehensively the envisaged roles for the VTUAVs from the perspective of the potential customers and their operational needs. The analysis resulted in slating the mission requirements of each of the operational needs and classification of the UAV technology based on the altitude of operation and its endurance. This was further categorised platform wise – Fixed wing or rotary wing. The environment of operation was also analysed by Sinha [34] and comprised of categorising it in terrain, weather, time, situation and threat. The operational needs & mission requirements and the operational environment are presented at Tables 2.3 and 2.4 respectively.

Table 2.3: Operations Needs and Mission Requirements of UAV Systems

(Sinha, 2001)

Operational Needs	Customers	Mission Requirements	Technology Classification
Global	Military	Distance & large theater surveillance and reconnaissance; Monitor hostile emission; Destroy mobile missile launcher & time-critical targets; Communication with covert formation; Ballistic missile defence; and Airborne early warning.	Support UAV HALE HALO MAME Combat UAVs UCAVs (HL) UCAVs (MM)
Global	Civil	Meteorological monitoring; Disaster monitoring; Atmospheric telecommunication relay platform.	Support UAVs HALE HALO MAME
Regional	Military	Precise targeting & terminal guidance; Force protection & battle management; Attack & suppress enemy ground targets; Combat Intelligence, Reconnaissance & Surveillance; Battle damage assessment; Communication relay; In-situ chemical/biological agent detection; Electronic & information warfare; and Immediate vicinity surveillance & reconnaissance	Support UAVs: MAME LALE VTUAVs (LE) VTUAVs (ME) VLAVLE (micro) Combat UAVs UCAV (MM) UCAV (LL)

Table 2.4: Mission Environments

(Sinha, 2001)

Operational Needs		**Military missions capabilities**		
(Roles)		Armed sub-mission	Attack sub-mission	Utility sub-mission
		Mission requirements		
	All terrain	• Desert • Jungle • Mountain	• Desert • Jungle • Mountain	• Desert • Jungle • Mountain
Natural	Adverse weather	• Hot • Wintry • Cyclonic • Rainy	• Hot • Wintry • Cyclonic • Rainy	• Hot • Wintry • Cyclonic • Rainy
	All times	• Day & night	• Day & night	• Day & night
	Situation	• High and hot • Hot and wet	• High and hot • Hot and wet	• High and hot • Hot and wet
Natural / Manmade	Threat	• Hostile • Non-hostile	• Hostile • Non-hostile	• Hostile • Non-hostile

VTUAV Concept of Employment (COE) analysed holistically several warfare missions from the perspective of mission success and identified supplementary tasks for each mission [41]. The supplementary tasks addressed intelligence, munition, management, defence and struggle. Each supplementary task was analysed from the logistics & mission perspective and crucial aide missions were identified as follows: a) Deploy & conduct manoeuvre; b) Develop intelligence; c) Employ firepower; d) Perform logistics & combat service support; e) Exercise command & control; and f) Protect force. COE analysed aide missions from operational perspective and identified the environment to be war-focused, littoral-based and variable in size.

Haskins [42] identified several civil sectors in which the usage of VTUAV will be mission and cost effective. It was based on an analysis of production and supplies. The sectors of effectiveness are the following: a) Forestry and agriculture; b) Power; c) Disaster and relief; d) Weather and environmental; and e) Scientific and geological research. The future of VTUAV design and development will be based on these sectors and manufacturing trends. It is envisaged that the design and development will need to address the following: a) Combat search and rescue; b) Operations in caves and buildings; c) Autonomous UAV swarms; and d) Medivac and Skytote.

The major VTUAVs in various stages of development and deployment are the following [43-46]:

- Eagle Eye – Bell Helicopter Textron: It is based on the pioneering tilt-rotor technology and provides both modes of flight;
- Fire Scout – Northrop Grumman Corporation: It is based on a 4-seater Schweizer 330SP helicopter;
- Cypher – Sikorsky Aircraft: It generates lift for hovering mode by employing a ducted fan consisting of two four-blade coaxial rotors and a conventional wing attached to the annular fuselage to provide lift in forward flight;
- Canard – The Boeing Company: It is characterised by the reaction-drive wing/rotor system. The configuration provides high speed and long range of a FWUAV with the flexibility of RWUAV capabilities and eliminates need for an anti-torque system, drive train & transmission;
- Guardian – Bombardier Aerospace: It is an improvised version of CL-227 Sentinel and is particularly designed for maritime applications;
- Vigilante – Science Applications International Corporation: It was initially designed to operate in manned, remote pilot and autopilot modes. It is now primarily used as a UAV for reconnaissance & surveillance roles;
- Camcopter – The Schiebel Group: It is programmed to fly along preselected routes and is fully autonomous and does not require any launching or recovery equipment;
- Seamos – EADS Dornier: It is characterised by two counter rotating rotors and has twin landing skids. It is primarily designed for sea reconnaissance [47]; and
- D'Gopher – Aircraft Research Institute: It is characterised by twin counter rotating rotors and offers higher aerodynamic efficiency with high manoeuvrability [48].

The detailed information specific to these and other VTUAVs in various stages of development and deployment is collated by Anon [49-51]. The collation also provides information about the various power-plants used in deployed VTUAVs.

Various concepts, beyond the conventional helicopter (fuselage and main & tail rotor system), are being proposed for VTUAVs, and includes rocket propulsion and magnetic propulsion take-off. Jet-Lift principle is another concept for addressing vertical take off issues. These concepts haven't gathered much support, making the conventional helicopter concept as the base-line design concept for the development of VTUAVs [52-54].

Brasel categorised VTUAV flight based on the operational mode as follows [54]:

- Pre-launch – It involves connecting air-vehicle to ground control system;

- Launch – It is usually based on rotary technology;
- Flight – It operates in nap-of-earth, low-level, peripheral, high-level and contour flight modes [31]; and
- Recovery – It is automatic and usually based on rotary technology.

The complete VTUAV system comprises of the following components [33, 41]:

- Air-vehicle – It comprises of the airframe and onboard systems. Onboard systems include fixed components and modular mission payloads. The fixed components comprises of following systems:
  - *Power-plant system*: It includes engine and transmission system responsible for propulsion.
  - *Rotor system*: It includes main and tail rotor systems constituting of hub and blades attached to it.
  - *Avionics system*: It includes air-data, attitude measurement and Fly-By-Wire (FBW) components. Air-data components provide airspeed, altitude and rate of climb & descent measurements by sensing air pressure through combination of static and pitot probes. Attitude measurement components (Artificial horizon) provide pitch, roll and yaw information. FBW components use multiple lanes of redundancy in the various computing channels and are used with air-data and attitude measurement components to execute control laws.
  - *Other systems*: It includes electrical, hydraulic and flight safety systems;
- Ground station – The air-vehicle is controlled by the Ground Control Station (GCS), which includes operator workstations and communicates with the air-vehicle through a data link;
- Remote data terminals – The Remote Data Terminal (RDT) provides capability to receive annotated (geo-location parameters etc.) video imagery from the air vehicle at remote locations from the GCS. The receive-only capability is primarily used to support scouting, targeting, intelligence dissemination, and similar activities;
- Data-links – The VTUAV data link supports video, data, and telemetry communications between the air-vehicle, GCS, and RDT. The Data-link includes the following: a) Air-vehicle component - Air Data Terminal (ADT); and b) Ground component - Ground Data Terminal (GDT). The VTUAV commonly uses Tactical Common Data Link (TCDL), a digital data link operating in the Ku frequency band (Frequency = 10.7-18.4 GHz and Wavelength = 2.5-1.67 cm);

- Launch and recovery equipment – An automatic takeoff and recovery is inherent in the VTUAV autopilot. A manual interface device is incorporated into the system to enable takeoff aborts and recovery wave-offs in event of emergency;
- Tactical communications equipment – Tactical communications are provided by VHF/UHF radios located at GCSs. The air-vehicle has an embedded VHF/UHF radio relay capability that augments system's tactical communications by extending its range and assisting in overcoming line-of-sight (LOS) issues;
- Activity support equipment – The Activity Support Equipment (ASE) includes necessary gear for vehicle assembly/disassembly, storage, shipment, and maintenance, repair/replacement of components;
- Transportation equipment – Aboard ship, VTUAV is stowed and transported using existing ship equipment. VTUAV systems used ashore require multiple High Mobility Multi-Wheeled Vehicles (HMMWVs); and
- Mission system technology – It includes systems that provide the required mission capabilities. These systems are sensors, communication links, survivability and weapon delivery system.

2.2.4 Mission System Technology

Mission systems technology comprises of systems that meet the mission requirements and is required for the design of the mission payload. These mission systems may be categorised from an operational or a capability perspective. The categorisation based on the capabilities it provides is as follows [33]:

- Observation system: It comprises of sensors that provide imagery information;
- Survivability system: It comprises mainly of electronic warfare systems for electronic information and electronic counter action;
- Communication system: It comprises of systems that communicates information to operational base for interpretation; and
- Weapon delivery system: It comprises of weapons and support equipments.

Sinha [33] further discusses the mission capabilities achieved by these mission systems and their components.

The categorisation from an operational perspective is as follows [55]:

- Information gathering: It obtains information through active and passive sensors and comprises of the following systems:
  - *Acoustic sensor*: It provides means of detecting and tracking the passage of objects.
  - *Attack/Surveillance radar*: It provides information on hostile and friendly targets.
  - *Camera*: It records high resolution images for intelligence purposes.
  - *Electro-optical sensor*: It provides passive target surveillance.
  - *Electronic support measure*: It provides emitter information, range and bearing of hostile transmitters.
  - *Electronic warfare system*: It detects and identifies enemy emitters to collect and record traffic and, if necessary, provides the means for jamming transmission;
- Intelligence development: The information is communicated to processing unit for transformation into intelligence by the following systems:
  - *Data link*: It provides transmission and reception of messages under encrypted communication using data-over-voice protocol.
  - *Mission computing unit*: It collates sensor information to provide a fused data picture to mission crew stations.
  - *Station keeping unit*: It provides station safety means; and
- Retaliatory action: The intelligence is used for further suitable action, that are provided by the following systems:
  - *Defensive aid unit*: It provides means for detecting and tracking missile attack and deploying countermeasures.
  - *Magnetic anomaly detector*: It confirms presence of large metallic objects prior to attack.
  - *Weapons system*: It arms, directs and releases weapons from the aircraft.

Mission system technology specific to VTUAVs collated by Anon [49-51] is presented at Appx A. Further detailed information of mission technology is at Appx B.

2.2.5 UAV Design Regulation

Civil Aviation Safety Regulation (CASR) Part 101 is the recommended safety guideline to be considered for unmanned aircraft design in Australian conditions [56]. It is based on the airworthiness standards established for UAVs in Federal Aviation Regulation (FAR) Part 101 as follows:

- UAVs are classified as micro (Weight less than 100 gm), small (Weight more than 100 gm and less than 100 kg), and large (Weight more than 100 kg). However, universally micro UAVs are exempted upto maximum weight of 8 kg;

- Micro UAVs are exempt from regulations, small UAVs have to follow regulations but do not require certification, and large UAVs must be regulated, certified and registered;
- Detailed procedures laid for the of UAV components;
- Flying height restrictions imposed at 400ft AGL; and
- State and territory laws should be complied in parallel.

2.3 Helicopter Design

The design of an air-vehicle is an exercise of compromise of design parameters to meet an optimised design configuration. The design parameters on broader terms cover mission capabilities, flight performance, reliability, maintainability, and cost. The design process of helicopters like any air-vehicle involves a three-tier process as follows: a) Operation; b) Design; and c) Manufacturing, usage & support [57]. The design concept covers preliminary and detailed design. The prime objective of a design process is to meet both the parametric requirements and constraints. The goal of the helicopter design process is to economise the size, weight and cost [31].

2.3.1 Design and Development Stages

The details of design and development stages of helicopters as stated above comprises of the following [58-59]:

- Conceptual and preliminary design: It comprises of an analysis of the customer needs for investigation of the feasible alternative solutions. The feasibility is governed by the technical viability and resource availability, including financial support;
- Detailed and complete design: This phase involves expertise from the disciplines to develop a technical viable configuration. The two guiding documents for this phase are the following: a) System Specification – It provides the operational performance, and safety capabilities to be met; and b) Design Criteria – It provides the guidelines for the detailed design;
- Assembly, manufacture and mass production: It comprises of the development of prototypes as follows: a) Cumulative mock-up - It is a temporary prototype, supplemented by various partial mock-ups, used for structural testing; and b) System integration fixture – It is a permanent prototype and contains all fuselage structural elements. Laboratory testing succeeds prototype development and qualifies each component as being airworthy. Before mass production, all components are operated coherently and inter-operability issues are identified and addressed; and

- Usage, refinement and disposal: It comprises of identification of components performance issues that need to be refined. The refinement phase is a continuous process. Depending on the service life or technological advancements, the aircraft is phased out.

2.3.2 Conceptual Design Methodology

Sinha [31] developed a methodology for the preliminary design of helicopter. The methodology was developed based on helicopter aerodynamics, traditional design processes and issues presented by Prouty [57-60]. The methodology involves a series of sequential computations and iterations, and refers to past helicopter design data and statistics. The computation involves a series of weight, aerodynamic parameters estimations, and power & speed evaluations for the establishment of physical dimensions. The computations provide the base for the development of a conceptual configuration to meet the slated design specifications.

The conceptual design process commences with a set of specific requirements derived from the operational needs and established by the potential customer. There are several requirements which influence the design. As an illustration, a set of critical design parameters that govern the conceptual design is presented in Fig 2.4.

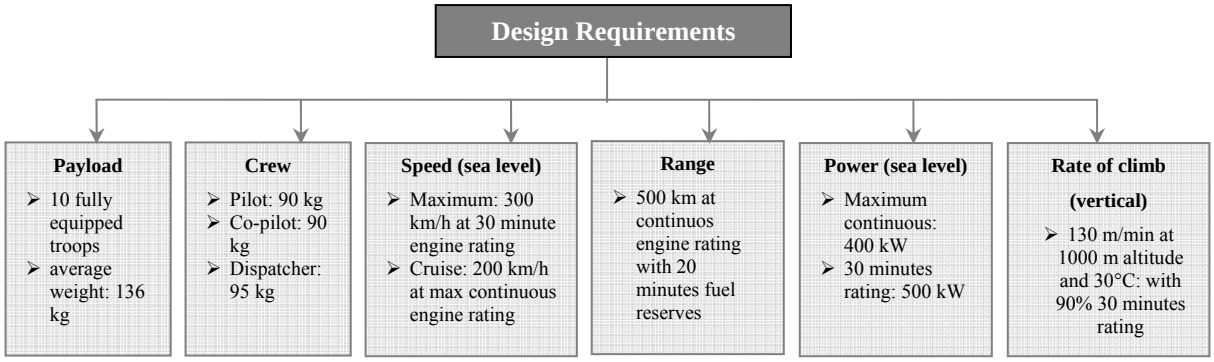


Figure 2.4: Design Requirements

(Sinha, 2000)

Additionally, there are several design constraints placed on the designer to meet the operational and safety standards. Constraints are also governed by state-of-the-art technology and support infrastructure. The conceptual design that emanates from the design process to meet the parametric requirements also addresses the imposed design constraints. An illustration of the design constraints placed on the designer is presented in Fig 2.5

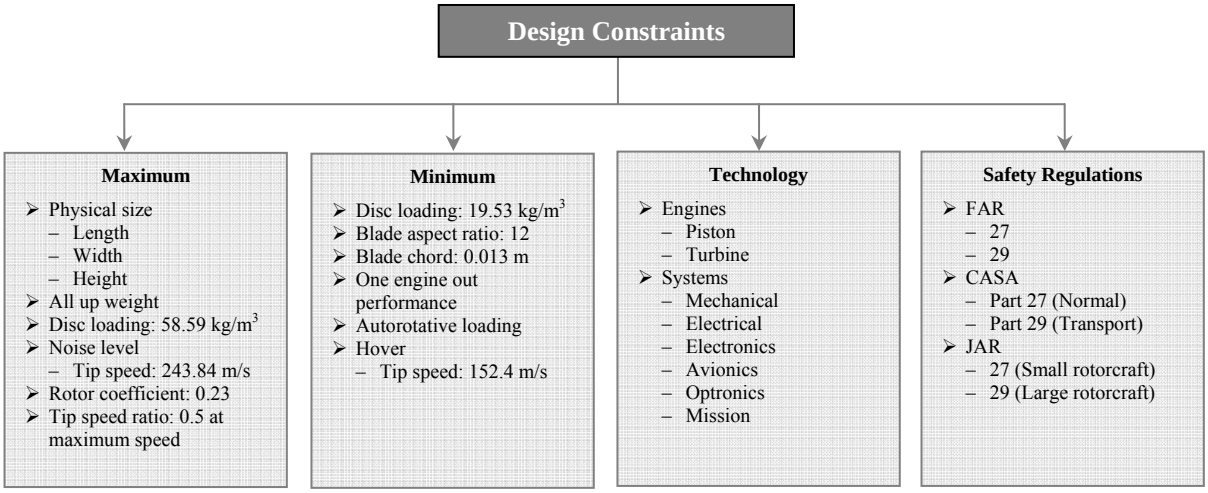


Figure 2.5: Design Constraints

(Sinha, 2000)

Civil Aviation Safety Regulation (CASR) Part 27 is the recommended safety guideline to be considered for helicopter design. It was developed to specify the airworthiness standards for helicopters with a Maximum Take-Off Weight (MTOW) of 3175 kg and nine passenger seat configurations. The airworthiness standards for rotorcraft in the normal category are the airworthiness standards set out in Part 27 of the Federal Aviation Regulations (FAR). CASR Part 29 was developed to specify the airworthiness standards for transport category rotorcraft with MTOW of 9090 kg and has been based on foreign legislation, specifically the USA's Federal Aviation Regulations Part 29. CASR Part 27 and Part 29 address the following: a) Design rotorcraft to be used in the Australian aviation industry; or b) Design modifications and repairs for rotorcraft [61-62].

The series of computation, estimation and iteration stages for the preliminary design of a helicopter, as formulated by Sinha [31], are as follows:

- Stage I - Weight and Power: The analysis covers the design requirements & operational environment established by the customer; constraints dictated by safety regulations; and limitations of the technology. Based on helicopter of similar class, the weight and power is estimated.

- Stage II - Fuel: The fuel is estimated from the mission profile of the aircraft and determined from installed engine power and its specific fuel combustion. The fuel required is estimated as follows:

$$W_f = sfc \times P_i \times T_m \qquad \textbf{Eqn 2.1}$$

Where,

W_f = fuel weight; sfc = specific fuel consumption;

P_i = installed power; and T_m = mission time.

- Stage III - Weight: The gross weight is obtained from aircraft's useful weight and the W_u/W_g factor. The useful weight is evaluated as follows:

$$W_u = \sum W_c + \sum W_{ms\,(int)} + \sum W_{ms\,(ext)} + \sum W_p + W_f \qquad \textbf{Eqn 2.2}$$

Where,

W_u = useful weight; $W_{ms(int)}$ = internal mission system weight;

W_c = crew weight; $W_{ms(ext)}$ = external mission system weight;

W_p = personnel weight; and W_f = fuel weight.

The W_u/W_g factor is read-off from graph at Fig 2.6 and the gross weight is evaluated as follows:

$$W_g = \frac{W_u}{W_u / W_g} \qquad \textbf{Eqn 2.3}$$

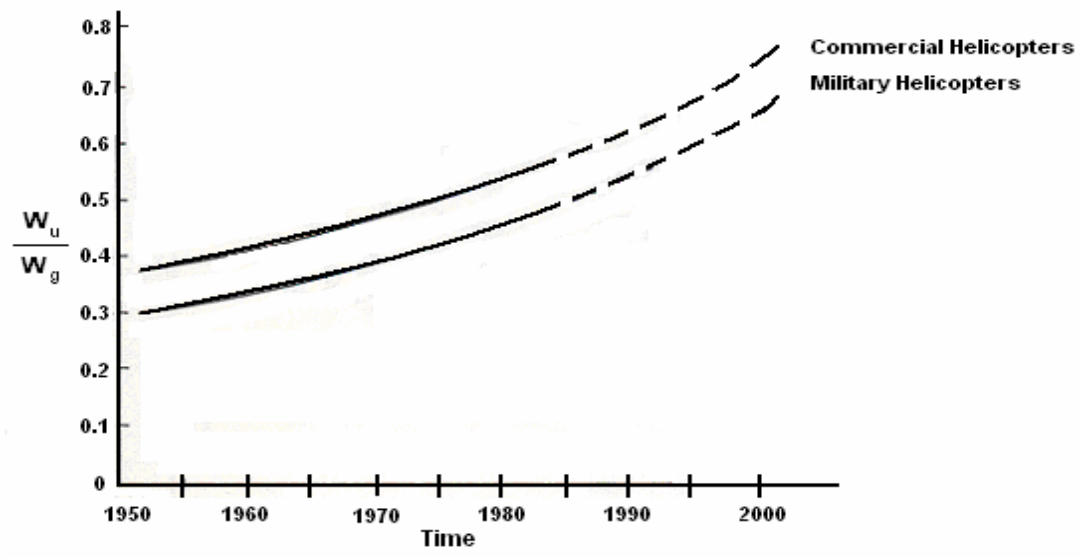


Figure 2.6: Chronological Ratio of Useful Weight to Gross Weight

(Prouty, 1986)

If the variance between evaluated gross weight and the recorded weight at Stage I is significant, then the recorded power at Stage I and the fuel is iterated, based on the associated sfc; to arrive at a new compatible gross weight.

- Stage IV - Vertical Drag: It is a measure of the downwash on the helicopter and is calculated from the disc loading and aircraft's projected area in the wake. The projected area is evaluated by developing the plan view of the fuselage from the layout. The vertical drag is evaluated from the following expression:

$$D_v = 0.3 \times D_L \times A_p \qquad \textbf{Eqn 2.4}$$

Where,

D_v = vertical drag; D_L = disk loading; and

A_p = projected area of all components in the remote wake.

- Stage V - Main Rotor System: The following parameters of the main rotor system dimensions are determined in this stage: a) Rotor size; b) Tip speed; c) Solidity; and d) Number of blades. Tip shape, taper, and twist are also addressed in this stage.

The main rotor diameter is evaluated from the disc area by the following expression:

$$D_L = \frac{W_g + D_v}{A_D} \qquad \textbf{Eqn 2.5}$$

Where,

D_L = disk loading; W_g = gross weight;

D_v = vertical drag; and A_D = disk area.

The tip speed is computed from the design requirements of maximum speed and the evaluated rotor diameter for a tip speed ratio of 0.5 from the following expressions:

$$\mu = \frac{V_f}{\Omega R} \qquad \textbf{Eqn 2.6}$$

$$V_T = \Omega R \qquad \textbf{Eqn 2.7}$$

Where,

μ = tip speed ratio; V_F = Forward speed;

R = rotor radius; Ω = rotational speed; and

V_T = tip speed.

The number of blades is evaluated from the "thrust coefficient"-to-"solidity ratio". The ratio ranges from 0.17 at sea level and varies with altitude to a maximum of 0.23. Considering sea level conditions and aspect ratio of 12 with a chord of 0.127; the number of blades is evaluated from the following expressions:

$$\frac{C_T}{\sigma} = \frac{T}{\rho\sigma A_D (\Omega R)^2} = \frac{W_g + D_v}{\rho\sigma A_D (\Omega R)^2} \quad \textbf{Eqn 2.8}$$

$$\sigma = \frac{A_b}{A_D} = \frac{bc}{\pi R} \quad \textbf{Eqn 2.9}$$

Where,

C_T = coefficient of thrust; σ = solidity;

T = thrust; W_g = gross weight;

D_v = vertical drag; ρ = air density;

A_D = disc area; ΩR = tip speed;

b = blade; c = chord; and

R = rotor radius.

The airfoil of blades is an exercise of compromise to meet the aerodynamics requirements of maximum lift coefficient and high drag divergence Mach number. The selection of a tip shape depends on design considerations to delay compressibility, reduce noise and introduce favourable dynamic aero-elastic twist. Aerodynamically, taper is required to improve hover performance by unloading the tips to achieve a uniform velocity distribution across the disc. The requirement of twist is high for good hover performance and to delay retreating blade stall. The twist is to be low, to reduce vibration and blade loads in forward flight.

- Stage VI - Drag in forward flight: In forward flight, the various drags that need to be estimated are as follows: a) Parasite drag from the non-lifting portions of the aircraft; b) Profile drag due to frictional resistance of the blades passing through the air. The parasite and profile drag coefficients in forward flight are determined in this design stage.

The parasite drag of the helicopter in forward flight is evaluated as the aggregate of the parasite drags of clean helicopter and the external loads, from the following expression:

$$C_{dp} = C_{dp(clean)} + C_{dp(ext)} \quad \textbf{Eqn 2.10}$$

Where,

C_{dp} = coefficient of parasite drag;

$C_{dp(clean)}$ = coefficient of parasite drag of clean helicopter with internal useful load; and

$C_{dp(ext)}$ = coefficient of parasite drag external.

The coefficient of parasite drag of the clean helicopter with internal useful load is estimated from the gross weight by the following expression.

$$C_{dp(clean)} = k\,((W_g - W_{ms(ext)}) / W_g)^{2/3} \quad \textbf{Eqn 2.11}$$

Where,

W_g = gross weight of helicopter; $W_{ms(ext)}$ = mission system weight external;

K = parasite drag-gross weight constant = 8.18 x 10^{-3}; and

$C_{dp(clean)}$ = coefficient of parasite drag of clean helicopter with internal useful load;

The coefficient of parasite drag for external loads is evaluated from the following expression:

$$C_{dp(ext)} = 1.15 \times \lambda_f C_f \quad \textbf{Eqn 2.12}$$

Where,

λ_f = form factor; $C_{dp(ext)}$ = coefficient of parasite drag external; and

C_f = coefficient of friction.

The coefficient of profile drag is evaluated from the average angle of attack by the following expression:

$$C_{do} = \delta_o + \delta_1 \alpha_{av} + \delta_2 \alpha_{av}^{\,2} \quad \textbf{Eqn 2.13}$$

For conventional airfoil, the value of δ_0, δ_1, and δ_2 are 0.0087, 0.0126, and 0.400, respectively.

Alternatively, profile drag coefficient can also obtained by assuming a parabolic profile drag curve of the form:

$$C_{do} = \delta + \Sigma \alpha_{av}^{\,2} \quad \textbf{Eqn 2.14}$$

Where, the median full scale values of Σ and δ is 0.3 and 0.010 respectively.

- <u>Stage VII - Power required in forward flight</u>: The power required to sustain forward flight is evaluated from the following expression:

$$P_f = \frac{W_g^{\,2}}{2A_d \rho V_f} + \frac{\sigma C_{d_o} \rho A_d (\Omega R)^2 (1 + 4.65\mu^2)}{8} + \frac{\rho C_{d_p} A_d V_f^{\,3}}{2} + P_m \quad \textbf{Eqn 2.15}$$

Where,

W_g = gross weight; A_D = area of disk;

ρ = air density at specific altitude; V_f = forward velocity;

σ = solidity of main rotor; C_{do} = coefficient of profile drag;

Ω = angular velocity; R = main rotor radius;

μ = tip speed ratio; and P_m = miscellaneous power.

If the calculated power is significantly different from the estimated power, then a new engine is selected and values of weight and drag coefficient are iterated.

- Stage VIII - Refinement of gross weight: The weights of each section / component of the aircraft is evaluated based on the mathematical expressions presented by Prouty [2, 3, 7, 8]. These equations undergo updates based on the usage of composite materials and state-of-the-art systems, and are expressed mathematically as follows:

  - Body (fuselage) = 6.9 * pow((Gross_Weight/1000),0.49) * pow(Length,0.61) * pow(Projected_Area,0.25) **Eqn 2.16**
  - Main rotor blades = 2 * 0.026 * pow(2*Rotor_Radius,0.66) * 0.4265 * pow(Rotor_Radius,1.3) * pow(Tip_Speed,0.67) **Eqn 2.17**
  - Nacelles = 0.041 * pow(Engine_Weight,1.1) + 0.33*pow(Projected_Area,1.3) **Eqn 2.18**
  - Dry system = 13.6 * pow((Non-Useful_Weight),0.82) * pow(5.5, 0.37) / pow((Tip_Speed / Rotor_Radius),0.64) **Eqn 2.19**
  - Propulsion system = 2 * pow(Engine_Weight,0.59) **Eqn 2.20**
  - Electrical system = 9.6 * pow((Non-Useful_Weight),0.65) / pow((Gross_Weight/1000),0.4) **Eqn 2.21**
  - Air conditioning = 8 * (Gross_Weight / 1000) **Eqn 2.22**
  - Instruments = 3.5 * pow((Gross_Weight / 1000),1.3) **Eqn 2.23**
  - Cockpit controls = 11.5 * pow((Gross_Weight / 1000),0.40) **Eqn 2.24**
  - Engine installation = (Installed Wt. per Engine) * (No. of Engine) **Eqn 2.25**
  - Manufacturing variation = 4*(Gross_Weight / 1000) **Eqn 2.26**

  Where,
  pow(x, y) = x^y

  Once the weights are refined the weight and balance is addressed. The longitudinal cg is evaluated from the static moments about a ‘datum line’ slightly ahead of the helicopter nose, for all components including mission systems & crew, and group weights, by the following expression:

$$cg = \frac{\sum M_{gp} + \sum M_{comp}}{\sum W_{gp} + \sum W_{comp}} \qquad \textbf{Eqn 2.27}$$

Where,

cg = centre-of-gravity; W_{gp} = group weight;

M_{gp} = moment of group weights; M_{comp} = moment of crew mission systems; and

W_{comp} = missions systems and crew weight,

- Stage IX - Trade-Off analysis: This last stage of the design involves trade-off analysis of the design parameters to achieve a compromise amongst gross weight, disk-loading, solidity, power, and tip speeds.

2.4 Design Automation

Technological advancements in computer software and hardware have led to their application in several fields including engineering. Automation [63] provides a cost-effective mode of enhancing productivity and product designs. The scope of automation in engineering is wide and covers from design, development to manufacture, operation and support. Automation is broadly classified as follows [64]: a) Fixed automation – It is a custom engineered application/equipment to automate a fixed sequence of operations that have high production rates and inflexible product design; b) Programmable automation – It is an application/equipment designed to accommodate a specific class of product changes that are batch produced; and c) Flexible automation – It is an application/equipment designed to manufacture a variety of products or parts with varying design.

Expert System (ES) is a computer system which emulates the decision-making ability of a human expert [65]. It is the most popular developmental approach of artificial intelligence due to following reasons: a) It preserves knowledge by building corporate memory; b) It assists when expertise is scarce, expensive, or unavailable; c) It guides under time & pressure constraints; and d) It trains new employees and improves work productivity. A holistic 'Expert System' design involves the following essential components [65]:

- Graphical user interface: A Graphical User Interface (GUI) is a human-computer interface, to provide manual inputs through a input device (mouse, keyboard etc) [66]. GUIs, in contrast to Command Line Interfaces (CLIs) that are text based and accessed solely by a keyboard, uses windows, icons and menus for display. A window is a

rectangular portion of the monitor screen that can display its contents independent of the rest of the display screen. An icon is a small picture or symbol in a GUI that represents a program, a file, a directory or a device (such as a hard disk or floppy). A menu is an annotated text display that represents a program command. The advantage of GUIs is that it provides user with an immediate, visual feedback about the effect of their action. GUIs allow users to take full advantage of the powerful multitasking capabilities of modern operating systems by displaying several instances simultaneously. Advanced GUIs provide user with the illusion of navigating through a three-dimensional space [66].

- Application service processing: It involves implementation of supervisory and developmental logic in a programming environment that is robust with powerful programming language and debugging & implementation tools. Programming language is a set of syntactic and semantic rules used to define computer instructions. The classification of programming languages are [67, 68]:
  - *Statically typed languages*: These languages use pre-defined, static and fixed data types for programming and perform type-checking during compilation time. Mainstream statically-typed languages include C, C++, Java, and C#. These are the most widely used programming languages and are primarily used in system programming;
  - *Dynamically typed languages*: These languages do not require explicit variable declaration before its usage and perform type-checking during run time. Prime examples of these are Objective-C, Lisp, JavaScript, Prolog, Python, and Ruby;
  - *Strongly typed languages*: These languages detect incorrect type usage, either at runtime for dynamically typed languages, or at compile time for statically typed languages. ADA and Java are examples of strongly typed languages; and
  - *Weakly typed languages*: These languages do not strictly enforce type rules or have an explicit type-violation mechanism. Assembly language and Tcl are examples of weakly typed languages.
- Database management system: A database is a collection of systematic information stored in a computer. The software used to manage and query a database is known as a Data-Base Management System (DBMS). DBMS range from simple to highly complex systems and are broadly classified into five major types as follows [69]: a) Navigational DBMS – It is founded on manual navigation of a linked dataset; b) Distributed DBMS – It distributes the actual database widely and connects them by a computer network; c) Object-oriented DBMS - It uses ‘objects’ to store data; d) Relational DBMS – It uses a

data model based on predicate logic & set theory, and stores data in a ‘record’; and e) Object-relational DBMS - It uses ‘objects’ and ‘records’ to store data.

2.4.1 Off-the-Shelf Software

Off-the-Shelf refers to products that are not custom made and can be directly purchased by the client [70]. The application offers generic usage and may need customisation specific to a need. An exhaustive list of off-the-shelf design softwares is discussed as below:

- AutoCAD: It is a three-dimensional design and a two-dimensional drafting and detailing tool, used by designers worldwide. It includes productivity tools, presentation graphics, Computer Aided Design (CAD) standards, and a user friendly GUI [71];
- CATIA: It is an integrated suite of CAD, Computer Aided Engineering (CAE), and Computer Aided Manufacturing (CAM) applications for digital product definition and simulation. It is used in mechanical design, product synthesis, system engineering, and machining & application development. It has wide application in aerospace design [72];
- SolidWorks: It is three-dimensional mechanical design software and provides a 3D CAD solution to manage product data more efficiently. Its provides speed & flexibility in managing large assemblies, enhanced design with advanced shapes, automation of repetitive tasks, and creation of automatic holistic design from individual parts [73]; and
- MATLAB, Simulink and Stateflow: MATLAB is a commercial "Matrix Laboratory" package which operates in an interactive programming environment. It enables visualisation of complex mathematical functions. Simulink is an interactive environment for modelling offline simulations with user-friendly block diagrams. Tightly integrated with Simulink and MATLAB, Stateflow provides an elegant solution for designing embedded systems that contains supervisory logic [74].

2.4.2 Software Development

Software development is a set of procedures that involves design and development of instructions based on the slated requirements. Programming paradigm is established and it comprises of the foundation application software and the following [75]: a) Procedural software – It is a sequential written code that involves code redundancy and overlapping functionality; b) Modular software – It comprises of structured modules that offer exclusive functionality; c) Object oriented software – It comprises of independent structured classes that promote reusability; and d) Distributed and concurrent software – It comprises of distributed application code over the network that promotes high security and faster execution. Object

oriented software has a wider scope of application and has inbuilt flexibility for future development.

The next step in the software development is selection of the type of database. The database needs to be compatible and robust for recording transactions including receiving critical inputs and updating storage. Relational Data-Base Management System (RDBMS) provides the required compatibility and robustness for the object oriented software [69].

Standalone applications cannot be upgraded to World Wide Web (WWW) or other intra-networks in future, unless they are programmed in .NET (Dot Network) environment. The .NET compatibility is offered by Microsoft across various platforms.

Finally, the connection between GUI and database has to be established for successful program execution. The connection can be established through a cross platform connection string that stores the physical location of database [76].

Software development tools that are available off-the-shelf and provide the above discussed functionalities are as follows:

- Microsoft Visual C++ .NET 2003: C++ is the most popular systems-level language, and Visual C++ .NET 2003 provides software developers a powerful tool to build application software [77]. It provides a dynamic development environment for creating microsoft windows & .NET–based applications, dynamic web applications and eXtended Markup Language (XML) web services. Visual C++ .NET includes the industry-standard Active Template Library (ATL) and Microsoft Foundation Class (MFC) libraries, advanced language extensions, and powerful Integrated Development Environment (IDE) features that enable developers to edit and debug source code efficiently [77]. With advanced template features, low-level platform access and an optimising compiler, Visual C++ .NET delivers superior functionality for generating robust applications and components;
- Microsoft SQL Server 2000: Microsoft SQL Server (MSSQL) is a Relational Data-Base Management and Analysis System (RDBMAS) for e-commerce, and data warehousing solutions. It offers a variety of administrative tools for database development, maintenance, and administration as follows [78]: a) Enterprise manager – It provides a management console to define, create, register, execute and administer new and existing servers; b) Query analyser – It creates queries and other SQL scripts and executes them against the database; c) SQL profiler – It allows system administrators to monitor

performance events by identifying slow executing queries; d) Service manager – It is used to start, stop and pause database services; and e) Data transformation services – It provides features for saving data lineage information to the database. MSSQL offers the following [78]: (a) Rich XML support; (b) Integration with windows server system; (c) Web-enabled analysis; (d) Web-access to data; (e) Multiple application hosting; (f) Robust application security through file and network encryption; (g) Indexed views; and (h) Full text search facility;

- Microsoft .NET Framework 1.1: The .NET framework has two main components as follows [79]: a) Common language runtime; and b) Class library. The Common language runtime is the foundation of the .NET framework, and class library is a comprehensive collection of reusable objects that are used to develop applications ranging from command-line to GUI. The .NET framework is designed to fulfill the following objectives [79]:
  - A consistent object-oriented programming environment.
  - A code-execution environment that minimises software deployment and version conflicts, promotes safe execution of code, and eliminates performance problems of scripted or interpreted environments.
  - It is consistent across windows-based and web-based applications.
  - It builds all communication on industry standards to ensure that code based on the .NET framework can integrate with any other code; and
- File DSN: Open Data-Base Connectivity (ODBC) technology is a standard programming language interface used to connect to all major data sources like SQL server and oracle. Data Source Name (DSN) is an ODBC term for the collection of information used to connect an application to a particular data source. The ODBC driver manager uses this information to create a connection to the database. A DSN can be stored in a file (File DSN) or in windows registry making the project portability easy [80].

2.4.3 Analytical Tool for Software Development

The design and development of the database application involves analysis of the referred statistical data for generation of representative expressions. The representative expressions are expressed as mathematical equations, and are derived from various statistical techniques. Regression is a prime statistical technique that involves sophisticated equations for analysis and translation of large datasets into line or curve coordinates [81]. The different types of regression include [81]: a) Linear regression (Linearisation) – It generates the slope of line based on a single dataset; b) Non-linear regression – It generates a logarithmic, polynomial,

exponential, power, or a moving average curve based on a non-linear dataset; c) Multiple regression – It generates a realistic projection from the analysis of multiple datasets by applying linear and non-linear regression techniques.

Microsoft Excel is a prime spreadsheet application with in-built regression functions. The TREND [82] regression function extrapolates future data and plots a linear line or non-linear curve, tended on data pattern. The R-squared value (R^2) determines the probability of pattern. A numerical value of 0.70 or more finalises the generated mathematical expression. The FORECAST [83] function incorporates TREND functionality but returns a single point rather than an array of points. The GROWTH [84] function is also based on TREND function but works with non-linear curves only.

3 REQUIREMENT ANALYSIS

The design of a VTUAV needs to be visualised from systems perspective for a holistic analysis of the operational needs & environment, technology and design processes. System approach is needed to transfer the operational needs and environment to mission requirements and the development of systems structure. The analysis needs to address the operational and technological issues for the design and development of the VTUAV.

The present and envisaged operational needs of VTUAVs were discussed in Sec 2.2.3, and comprises of global and regional requirements for the military and civil customers. The operational environment of VTUAVs includes varying terrain, weather, time, situation and threat (Sec 2.2.3). The design technology of VTUAVs, as discussed in Sec 2.2.2, covers crucial redundancy, advanced communication and efficient propulsion. To address the operational, environmental, and technological issues of VTUAVs, the requirement analysis needs to be subdivided as follows: a) Operational analysis – To cover the operational and environmental issues based on customers need statement; and b) Mission analysis – To cover the technological issues based on operational need.

3.1 Mission Need Statement

The requirement analysis commences with a customers "Mission Need Statement" that describes the objectives of the design. The need statement comprises of a brief of the operational environment, current & planned capabilities including their shortfalls, developmental timeframe, and the fiscal allocation. To develop a generic design process for a VTUAV a customised need statement was developed from the study of present and envisaged needs of the UAV technology (Sec 2.2.1 to 2.2.3). The need statement is listed as below:

"The present global scenario necessitates the need to provide a responsive capability to conduct wide-area and near-real-time reconnaissance, surveillance, and target acquisition, command and control, signals intelligence, electronic warfare, and special strategic & tactical missions during peace and war times against defended/denied areas over long durations. The evolution of the hostile surface-to-air and air-to-air threat and their collective effectiveness against manned aircraft and satellites will generate unacceptably high attrition and strike-loss rates. The current systems cannot perform these missions in a timely, responsive manner in an integrated hostile air defence environment without high risk to

personnel and costly systems. There is a need for a capability which can be employed in areas where enemy air defences have not been adequately suppressed, in heavily defended areas, in open ocean environments, and in contaminated environments. Nuclear survivability is required to perform missions in a nuclear contaminated environment.

The technological cost justifies the growing demand of unmanned technology in the civilian sectors of agriculture, forest, power and research. The mission activities range from aid and monitorship, to vigilance and examination over the designated areas for long durations and in varied climatic conditions. The current systems cannot perform these activities without high risk to workforce. There is a need for automated systems that can arrive at mission area in short span during emergencies. Resource and fiscal allocations are forfeited for the product robustness that ensures maximum functionality with minimum resources."

3.2 Operational Analysis

The operational analysis investigates operational and environmental issues that emanate from the customised need statement.

The objectives of the design and the expected capabilities of the VTUAV were analysed to formulate a set of operations for the VTUAV including the environment. The analysis is discussed as follows:

- Operational needs: The mission-need-statement stipulates global and regional needs of the VTUAV to address civil and military clientele. In broader terms, it covers reconnaissance & surveillance, and offensive capabilities including communications. The operational needs are categorised as follows for further detailed analysis:
  - Civilian needs: To address commercial requirements; and
  - Military needs: To address defence requirements.
- Operational environment: The mission-need-statement provides a broad insight into the external environment in which the VTUAV will operate. It comprises of natural and manmade environment as follows:
  - Terrain;
  - Weather;
  - Time;
  - Situation;
  - Threat; and
  - Littoral

3.3 Mission Analysis

The mission analysis investigates the operational needs from the perspective of mission features to identify technological requirements.

The mission investigation of various VTUAVs transformed the operational needs into following mission categories based on their execution schedule: a) General missions – These missions address common civil and military needs; b) Special missions – These are customised civil and military missions. Both the mission categories are discussed in depth as follows:

- **General missions:** The operational analysis of the need statement resulted in the segregation of most widely usage of VTUAVs. These were categorised as "General missions". These missions represented the UAV technology development and deployment on both civil and military missions. The operational analysis further provided an insight into the environmental issues. The mission environment covers varying terrain, weather, time, and situation. The various identified common missions and their features are as follows:
  - Reconnaissance and surveillance: The mission involves armed and unarmed secret vigilance over a designated territory. It comprises of passive observation at varied levels of altitude and speed, including penetration in the forward line of enemy troops when target information is incomplete or the enemy's flank is unprotected. This is to provide information about enemy deployments, movements and selection of routes;
  - High altitude scientific research: The UAVs provide high altitude imaging to monitor topography (which out-limits the satellite wavelength bandwidth). It covers mapping of peripheral and earthy textures, including airy constituents; and
  - Fire detection and monitoring: It involves the usage of multiple temperature and smoke sensitisers (heat resistant and impermeable components) for detecting and following-up fire breakouts over an operational area.
- **Special missions:** These are customised VTUAV missions that demonstrate the versatility of the technology. The operational environment of special missions is the degree of threat and is situation dependent, in addition to varying terrain, weather and time. The missions categorisation is as follows:
  - *Civilian missions:* These are unarmed missions. The potential usage of VTUAVs is in agriculture, forest, power, research and management applications. The operational environment is situation, weather and terrain focussed. Civilian mission classification is discussed as follows:

- Relief and monitoring: These missions involve relief and support in situation based environment, and are as follows:
  - Agricultural aide and monitorship: It will involve spraying of seeds, fertilisers, and insecticides/pesticides and monitoring of weather changes to detect locust and rodents movement;
  - Disaster and relief assistance: It will involve the delivery of relief aid to affected areas and the notification of unaided topography;
  - Forestry assistance and vigilance: It is to provide rescue operations and customised guidance about topography and damage. It includes monitoring to detect troubled areas and report their location; and
  - Power monitorship: It involves monitoring pipelines and electrical lines, and implementation of recovery and reporting procedures.
- Research: These missions cover topographical exploration in weather and terrain based environment and are as follows:
  - Scientific and geological research aide: The mission involves customised assistance to cover experimentation and analysis.

o *Military missions:* These are armed missions for an offensive or defensive role and are categorised as strategic and tactical missions. In addition to the operational environment as discussed above, it includes littorals. The missions are as follows:

- Strategic military missions: These missions are planned and executed at the higher level and have a strategic implication on the operation. A higher threat level is associated with the operational environment. The key strategic missions are as follows:
  - Detect electronic emissions: It involves detection of data-loss, data-redundancy and data-surge during transmission and reception, and its reporting;
  - Detect nuclear, biological and chemical emissions: The mission involves usage of multiple nuclear, biological and chemical sensors and equipments to detect and report emissions;
  - Deploy forces and conduct manoeuvres: It involves providing assistance to armed forces by navigating and deploying them into safety zones;
  - Develop intelligence: It involves collection, analysis and documentation of war-time intelligence inputs;
  - Detect, identify and geo-locate communications/non-communication channel: It involves reporting of communication and non-communication channels between airborne vehicles and ground stations by detecting the data-link,

classifying it, identifying the source/sink, and finally plotting the source/sink location;

- Illuminate, range, and designate location: It involves illuminating target area, and updating the control station with its location coordinates;
- Protect force: It involves enhancing military survivability and supplementing the rescue & recovery; and
- Weapon delivery assignments: It involves the weapon delivery with minimal ingress and egress time.

- Tactical military missions: These missions are implemented at field-level. A reduced threat level is associated with the operational environment. The missions comprises of the following:
  - Dispense supply: It involves dispensing of incapacitating agents, logistical supplies, pamphlets and sonobuoys by using ground, meteorological and nuclear, biological, chemical sensors;
  - Command and control: It involves a quick situational assessment through command and control procedures by acquiring, analysing and communicating information to the ground station;
  - Electronic attack and protection: It involves hijacking communication frequency of airborne vehicles and decoying their airflight path by broadcasting deceptive frequency or by jamming the channels. Protection covers re-encryption and closure of the communication frequency;
  - Firepower: It involves engaging targets through detection, classification and tracking;
  - Relay: It involves audio and imagery data relay from airborne object to ground platform through relay sensors and communication network;
  - Imagery and geo location: It involves processing of still & video imagery, and updating the ground station with air-vehicle coordinates by using infrared frequency of a synthetic aperture radar; and
  - Target-return data: It involves updating ground control station with target data obtained from multi-spectral radar through a surface search algorithm.

The results of the mission analysis are summarised at Table 3.1 to systematically present the following: a) Operational needs; b) Mission categories; c) Mission requirements; d) Environment.

Table 3.1: Summarised Results of Mission Analysis

Operational needs	Mission clientele	Mission categories	Mission requirements	Environment
General	Military & Civil	Generic	• Reconnaissance & surveillance • High altitude scientific research • Fire detection & monitoring	Terrain, weather, time & situation
	Civil	Relief & monitoring	• Agricultural aide & monitorship • Disaster & relief assistance • Forestry assistance & vigilance • Power monitorship	Terrain, weather, time & situation
		Research	• Scientific & geological research aide	Terrain & weather
Special	Military	Strategic	• Dispense supply • Command & control • Electronic attack & protection • Firepower • Relay • Imagery & geo-location • Target-return data	Weather, time & threat
		Tactical	• Detect electronic emissions • Detect nuclear, biological & chemical emissions • Deploy forces & conduct manoeuvres • Develop intelligence • Detect, identify & geo-locate communications/non-communication channel • Illuminate, range & designate location • Protect force • Weapon delivery assignments	Terrain, weather, time, situation & threat

3.4 Mission Profile

Mission requirements are further investigated from an operational perspective to determine the mission phases for the development of the mission profile. Mission profile is defined as a sequential representation of various mission phases from an operational perspective. It provides the flight mode and estimates the time required to fulfill the slated mission requirements as an aggregate of time period for various phases. The flight mode, as discussed in Sec 2.2.3, for rotary wing UAV missions comprises of nap-of-earth, low-level, peripheral, high-level, and contour. A generic rotary wing UAV mission profile comprises of following phases:

- Take-off: It is either concealed or open, and includes taxing and take-off time;
- Transit (Source to destination): It involves an outbound transition to destination through a series of way points;
- Ingress: It involves the entrance in threat/non-threat zone;

- Engage: It is execution of the mission;
- Disengage: It is accomplishment/abort of mission;
- Egress: It involves exit from threat/non-threat zone;
- Transit (Destination to source): It involves an inbound transition to base through a series of way points; and
- Landing: It covers both landing and taxing time.

Two mission profiles are developed considering the above generic phases to address the operational needs of reconnaissance & surveillance and weapon delivery missions. It provides a series of time periods to evaluate the total mission time. The mission profiles are presented in Fig 3.1 and 3.2.

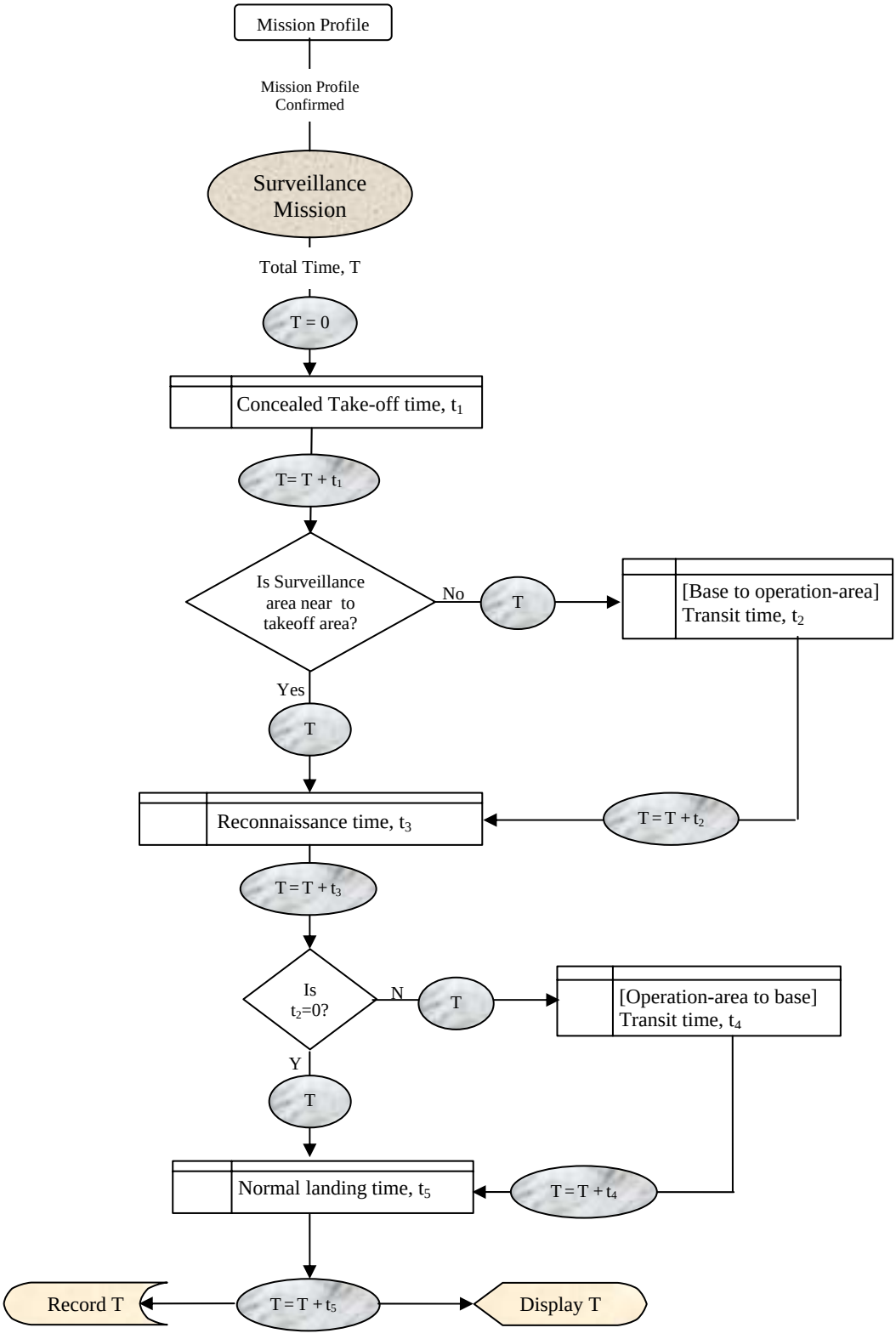


Figure 3.1: Flowchart to Calculate Time of Reconnaissance & Surveillance Mission

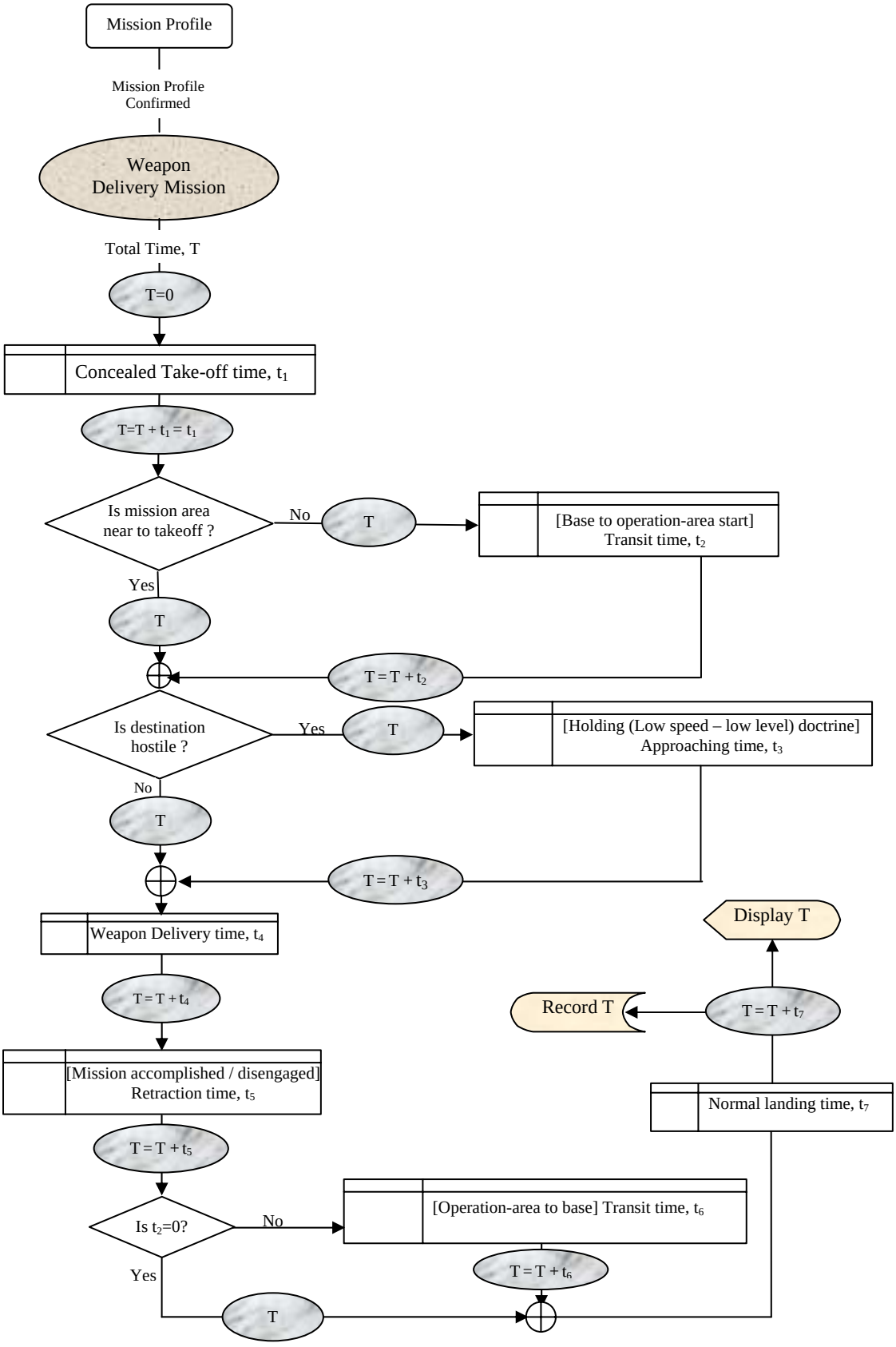


Figure 3.2: Flowchart to Calculate Time of Weapon Delivery Mission

4 SYSTEM DESIGN

The system engineering process is applied to design the system architecture based on the mission requirements generated from the requirement analysis (Sec 3).

The system architecture design involves comprehensive system development by identifying participating system that collectively meets the slated requirements (Sec 2.1.5). Each identified system is analysed as a set of components with specific attributes (Functional characteristics) and then further investigated for intra and inter relationships between components and attributes (Sec 2.1.1). The operational environment of the system is identified including the system input from the environment and the system output to the environment. The system productivity is represented as system output. The system design at the various levels comprises of functional analysis and synthesis (Sec 2.1.5).

4.1 Functional Analysis

Functional analysis of an aircraft involves investigation of vehicle function (Design analysis including aerodynamics, propulsion, navigation and communication), operational function (Pre-flight to post-flight analysis including emergency), and system function (Maintenance and training activities) (Sec 2.1.7).

The complete VTUAV system needs to comprise of the following three components (Sec 2.2.2 to 2.2.4) with desired functions to meet the slated mission requirements (Table 3.1) derived from the requirement analysis (Sec 3):

- Mission system: To provide the mission functional characteristics – attributes (Sec 2.2.4 & 3.3);
- Vehicle system (Helicopter): To provide the flight functional characteristics (Sec 2.2.3 & 2.3.1-2.3.2); and
- Autonomous system: To provide the required automation characteristics (Sec 2.2.3 & 2.4).

These components together with intra and inter relationships will form the generic VTUAV system (Sec 2.1.1).

4.1.1 Mission System

Mission system comprises of systems that provide the required mission capabilities and are vital for mission accomplishment. The design features of the mission system constitute a set

of attributes which represent the functional characteristics of the system. These characteristics address the mission requirements. Hence the attributes of the mission system technology needs to be investigated to identify the functional characteristics that will meet the mission requirements. This will assist in the design of an appropriate mission payload.

The types and functions of mission system technology (Sec 2.2.4) presented at Appx A, and its details at Appx B were re-examined for appropriate grouping to design the mission system component as part of the total UAV system. This was needed to provide a broader functionality that holistically addressed the varied functional characteristics needed to meet the mission requirements. This was achieved by a two-stage process comprising of Level-I classification and Level-II classification. The grouping is presented in Appx C.

The regrouping resulted in reclassification of mission system technology in seven components representing generic functional characteristics. The components of the mission system technology and its functional characteristics are as follows (Table 4.1):

- Acoustic sensor: It detects objects and reports data based on sound patterns.
- Communication unit: The functional characteristics are as follows:
  - Wireless monitoring and control of an incoming data signal;
  - Propagation of an electromagnetic signal;
  - Interaction over a secure data channel; and
  - Detection and decryption of hostile data signals.
- Electro-optical unit: It provides the functional characteristics of passive target surveillance by imaging undercover, processing images as intelligence inputs, and displaying them for further action.
- Electronic warfare unit: It includes electronic support measure, electronic counter measure, and electronic counter counter measure components that detect and identify enemy emitters to collect and record traffic. It also provides means for jamming transmission by using modular surveillance payload, decoys and jammers.
- Forward looking & thermal imaging unit: It assesses battle damages for support measures, and comprehends active target surveillance coupled with retaliatory measures.
- Integrated unit: It includes on-board and on-ground components that are used in tandem to achieve real time data processing and tactical control.
- RADAR: The radio detection and ranging component detects, ranges, and maps an object for further action.

Table 4.1: Grouping of Mission System Technology for Systematic Analysis of Components, its Elements & Functions

Mission system technology grouping (Component)	Elements	Functional Characteristics
Acoustic Sensor	• Sensor array	• Object detection based on varied sound patterns • Data reporting
Communication unit	• Transponder • Transmitter • Terminal • Interceptor	• Wireless monitoring & control of incoming signal • Propagation of electromagnetic signal • Interaction over secure data channel • Detection and decryption of hostile data signal
Electro-optical unit	• Imaging processor with sensor system • Display and observation unit • Camera and modular accessories	• Imaging undercover • Processing images as intelligence inputs • Displaying intelligence inputs
Electronic warfare unit	• Modular payload • Radar warning receiver with electronic surveillance module • Radar warning receiver with signal intelligence component • Radar decoys • Jammers	• Detection and identification of enemy emitters • Recording aerial traffic • Hacking enemy transmission
Forward looking & thermal imaging unit	• Imaging camera • Imaging component • Imaging module	• Assessment of battle damages • Aiding support measures • Comprehending active target surveillance • Comprehending retaliatory measures
Integrated unit	• Onboard UAV component • Base component	• Processing real-time data • Achieving tactical control
RADAR	• Radio detection and ranging component	• Object detection • Object ranging • Object mapping

The hierarchy (Sec 2.1.1) of the mission systems technology is developed by a block representation of the seven components and their elements identified in Table 4.1. The mission system hierarchy is presented at Fig 4.1.

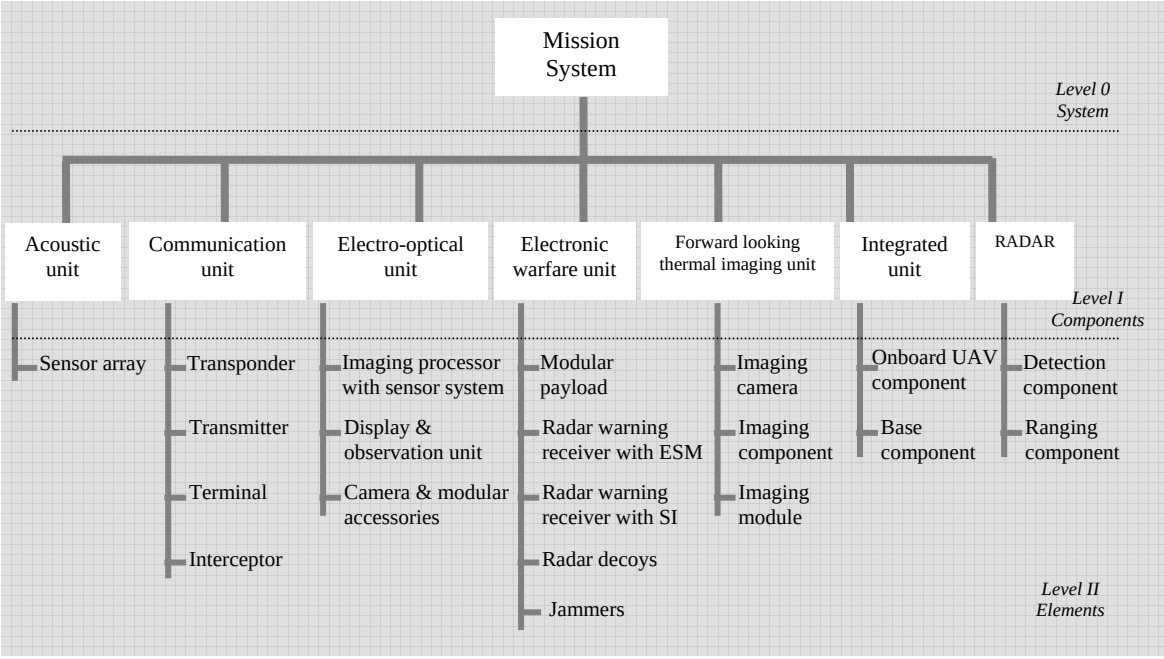


Figure 4.1: Mission Systems Technology Hierarchy Representing the Components & Elements

The mission analysis (Sec 3.3) identified the mission requirements of the VTUAV. These requirements are now studied vis-à-vis the mission system technology and its components & functional characteristics (Table 4.1). This analysis is for identification of the mission systems that meet the slated mission requirements of the VTUAV to be designed. The analysis resulted in the identification of the following: a) Base components; and b) Additional components. Base components are the mission systems that meet mission requirements and provide the desired capabilities. Additional components enhance the capabilities of the base components. The identified mission systems that met the mission requirements are presented at Table 4.2.

Table 4.2: Identification of Mission Systems that meet Mission Requirements

Mission requirements (Sec 3.3)	Mission system technology (Base component)	Mission system technology (Additional component)
Reconnaissance & surveillance	• Electro optical unit • Forward looking & thermal imaging unit	• Acoustic sensor
High altitude scientific research	• Electro optical unit • Communication unit	• Integrated unit
Fire detection & monitoring	• Electro optical unit	• Forward looking & thermal imaging unit
Agricultural aide & monitorship	• Integrated unit	• Electro optical unit
Disaster & relief assistance	• Integrated unit • Electro optical unit	• Forward looking & thermal imaging unit
Forestry assistance & vigilance	• Integrated unit • Electro optical unit	• Forward looking & thermal imaging unit
Power monitorship	• Electro optical unit • RADAR	• Forward looking & thermal imaging unit
Scientific & geological research aide	• Electro optical unit • Communication unit	• Integrated unit
Dispense supply	• Integrated unit • Electro optical unit	• Forward looking & thermal imaging unit
Command & control	• Communication unit • Electronic warfare unit	• Integrated unit
Electronic attack & protection	• Electronic warfare unit • Forward looking & thermal imaging unit	• Integrated unit
Firepower	• Electronic warfare unit • Forward looking & thermal imaging unit	• Integrated unit
Relay	• Communication unit	• Electro optical unit
Imagery & geo-location	• Electro optical unit • Communication unit	• Forward looking & thermal imaging unit
Target-return data	• Communication unit	• Acoustic sensor
Detect electronic emissions	• Electro optical unit	• Acoustic sensor
Detect nuclear, biological & chemical emissions	• Forward looking & thermal imaging unit • Electronic warfare unit	• Integrated unit
Deploy forces & conduct manoeuvres	• Electronic warfare unit • Communication unit	• Integrated unit
Develop intelligence	• Electro optical unit • Electronic warfare unit	• Communication unit
Detect, identify & geo-locate communications/non-communication channel	• Electronic warfare unit • Communication unit	• Integrated unit
Illuminate, range & designate location	• Forward looking & thermal imaging unit • Electro optical unit	• RADAR
Protect force	• Forward looking & thermal imaging unit • Electro optical unit	• Integrated unit
Weapon delivery assignments	• Electronic warfare unit • Forward looking & thermal imaging unit	• Communication unit

The functional characteristics of the additional and base components (Table 4.2) represented in the mission system technology hierarchy (Fig 4.1), that meet mission requirements are now analysed to stipulate their attributes (Sec 2.1). These attributes address mission systems from

an operational perspective for an understanding of the mission capabilities it provides. The components of the mission systems and its attributes comprises of the following:

- Acoustic sensor: The functional characteristics involve detection and reporting of the object from varied sound patterns. The detection and reporting of the object is based on constant observation. Hence the attribute of acoustic sensor is – observation;
- Communication unit: The functional characteristics involve detection, decryption, monitoring, and control of incoming signals. The observed signal is then communicated to the base-station or targeted-receptor. Hence the attributes of communication unit are – communication and observation;
- Electro-optical unit: The functional characteristics involve spied imaging based on continuous observation, and processing images into recorded intelligence inputs. The intelligence inputs may be communicated to base station or stored onboard. Hence the attributes of electro-optical unit are – observation; survivability; and communication;
- Electronic warfare unit: The functional characteristics involve detection and identification of enemy emitters and recording enemy transmissions by an endurable unit, which is navigated into enemy territory after safely avoiding detected obstacles. Hence the attributes of electronic warfare unit are – survivability; navigation; and obstacle detection and avoidance;
- Forward looking & thermal imaging unit: The functional characteristics involve assessment and observation of battle damages for providing support and retaliatory measures by a unit that should be safely navigated into the battlefield. Hence the attributes of forward looking & thermal imaging units are – observation; survivability; and navigation;
- Integrated unit: The functional characteristics involve manual operation of on-board and ground components that when used together achieve real time data processing and tactical control. Hence the attribute of integrated unit is – operator activity; and
- RADAR: The functional characteristics involve object observation including detection, ranging, and mapping. Hence the attribute of radio detection and ranging component is – observation.

The subsystems (components) and the attributes of mission system that meet slated mission requirements are presented in Table 4.3.

Table 4.3: Subsystems and Attributes of Mission System

Mission Subsystems	Attributes
Acoustic	▪ Observation
Communication	▪ Communication ▪ Observation
Electro-optical	▪ Observation ▪ Survivability ▪ Communication
Electronic warfare	▪ Survivability ▪ Navigation ▪ Obstacle detection & avoidance
Forward looking and thermal imaging	▪ Observation ▪ Survivability ▪ Navigation
Integrated	▪ Operator activity
RADAR	▪ Observation

4.1.2 Vehicle System - Helicopter

The vehicle system comprises of subsystems (components) which provide flight functional characteristics to the VTUAV. These components need to be identified by the systematic analysis of the VTUAV as a system in its conceptual design process (Sec 2.2.3). The helicopter system design process (Sec 2.3.2) comprises of various stages of evaluations of the design parameters to meet the design specifications and constraints of the various components. The design process governs the attributes of the subsystems being designed.

The subsystems of the vehicle system being considered in the design process provide the avenues to identify the components and its attributes of the VTUAV system that addresses the flight functional characteristics. The design process comprises of nine stages (Sec 2.3.2) and the subsystems considered in the conceptual design covers the following: a) Airframe b) Power-plant; and c) Rotor.

The design process of these subsystems (components) is investigated in detail to identify its attributes as follows (Sec 2.3.2):

- Airframe – The airframe subsystem when considered in the design process involves the following: a) Evaluation of vertical drag - Stage IV; b) Calculation of parasite drag - Stage VI; c) Refinement of gross weight - Stage VIII; and d) Development and refinement of external profile including the sizing - Stage IV & IX. Hence the key characteristics of the airframe subsystems conceptual design involves mathematical evaluations and optimisation of the evaluations through refinement or design trade-offs;
- Power-plant – The power-plant subsystem when considered in the design process involves the following: a) Approximation of power - Stage I; b) Calculation of fuel - Stage II; c) Recalculation of gross weight - Stage III; d) Calculation of power - Stage VII; and e) Trade-off analysis of design parameters including power - Stage IX. Hence the key

characteristics of the power-plant subsystems conceptual design involves mathematical calculations and re-calculations of the approximations through design trade-offs; and

- Rotor – The rotor subsystem when considered in the design process involves the following: a) Evaluation of rotor diameter, tip speed and number of blades - Stage V; b) Calculation of profile drag - Stage VI; and c) Trade-off analysis of design parameters including rotor tip speed - Stage IX. Hence the key characteristics of the rotor subsystem conceptual design involves mathematical evaluations and calculations through design trade-offs.

The subsystems and its key characteristics (Attributes) of the vehicle system addressed during various stages of helicopter design process are presented in Table 4.4

Table 4.4: Subsystems & Attributes of Vehicle System addressed during various Stages of Helicopter Design Process

Vehicle subsystem	Helicopter design process stages	Attributes
Airframe	• Stage IV - Vertical drag calculation • Stage VI - Drag calculation • Stage VIII - Gross weight refinement • Stage IX - Trade-off analysis	• Evaluation • Calculation • Refinement • Trade-offs
Power-plant	• Stage I - Weight & power approximation • Stage II - Fuel calculation • Stage III - Gross weight recalculation • Stage VII - Power calculation • Stage IX - Trade-off analysis	• Approximation • Calculation • Recalculation • Calculation • Trade-offs
Rotor	• Stage V - Main rotor system design • Stage VI - Drag calculation • Stage IX - Trade-off analysis	• Evaluation • Calculation • Trade-offs

The above investigation of the design of subsystems identified the attributes of vehicle system. These attributes needs to be further refined for the development of the system structure (Sec 2.1) subsequently. The design stages at which the subsystems are considered are following:

- Stage I (Power-plant): In this stage the weight and power values is estimated based on the helicopters of similar class. Hence, the attribute of Stage I is – estimation;
- Stage II (Power-plant): In this stage the fuel is computed based on the mission profile of the aircraft and the specific fuel combustion. Hence, the attribute of Stage II is – computation;
- Stage III (Power-plant): In this stage the useful weight is computed to estimate the gross weight based on chronological data. If the variance between estimated gross weight and recorded weight at Stage I is significant, then the estimated power and computed fuel

values are iterated. Hence, the attributes of Stage III are – computation; estimation; and iteration;

- Stage IV (Airframe): In this stage the vertical drag is computed at maximum disc loading from the estimated drag coefficient and computed projected area of external profile. Hence, the attributes of Stage IV are – computation and estimation;
- Stage V (Rotor): In this stage the rotor diameter, tip speed and number of blades are computed from estimated disk loading, estimated tip speed ratio, maximum speed, estimated aspect ratio, and chord value. Hence, the attributes of Stage V are – computation and estimation;
- Stage VI (Airframe & Rotor): In this stage the parasite drag coefficient (Airframe) is computed from estimated gross weight, computed form factor and coefficient of friction. The profile drag coefficient (Rotor) is computed from the computed average angle-of-attack and Mach number. Hence, the attributes of the stage VI are – computation and estimation;
- Stage VII (Power-plant): In this stage the power required to sustain forward flight at maximum speed is computed from the computed parasite and profile drag coefficients. Hence, the attribute of Stage VII is – computation;
- Stage VIII (Airframe): In this stage the weights of each aircraft section and components are optimised based on the used advanced materials and mission systems. Hence, the attribute of Stage VIII is – optimisation; and
- Stage IX (Airframe, Power-plant and Rotor): In this stage trade-off analysis of design parameters including estimated disc-loading, computed solidity, computed power (Power-plant), and computed tip speeds (Rotor) is involved to reduce the estimated gross weight (Airframe). Hence, the attribute of Stage IX are – trade-off; estimation; and computation.

The refined attributes of the vehicle system identified at various stages of the helicopter design process are presented in Table 4.5.

Table 4.5: Refined Attributes of Vehicle System

Helicopter design process stages (Vehicle Subsystems)	Attributes
Stage I (Power-plant)	• Estimation
Stage II (Power-plant)	• Computation
Stage III (Power-plant)	• Computation • Estimation • Iteration
Stage IV (Airframe)	• Computation • Estimation
Stage V (Rotor)	• Computation • Estimation
Stage VI (Airframe & Rotor)	• Computation • Estimation
Stage VII (Power-plant)	• Computation
Stage VIII (Airframe)	• Optimisation
Stage IX (Airframe, Power-plant & Rotor)	• Trade-off • Estimation • Computation

4.1.3 Autonomous System

The autonomous system comprises of onboard and ground subsystems that provide autonomous characteristics (Sec 2.2.3), vital for the VTUAV operation. The autonomous characteristics need to cover the command and control of the flight and mission. The status of the automation technology provides the degree of automated flight and mission control. The subsystems that contribute to automation were identified in Sec 2.2.3 and cover the following subsystems: a) Onboard avionics; b) Remote data terminal; c) Data-link; and d) Launch & recovery. A further detailed investigation is needed to identify the attributes of the autonomous subsystem

The investigation of these categorised autonomous subsystems (components) identified critical attributes that represent functional characteristics of automation as follows (Sec 2.2.3):

- Onboard avionics: The vehicle's onboard systems comprises of mission systems (Sec 4.1.1) and fixed components including avionics. Avionics components comprises of the following: a) Air-data – It senses total and static air pressure; b) Artificial horizon / Attitude indicator – It provides pitch, roll & yaw information; and c) Fly-by-wire – It uses multiple lanes of redundancy in computing channels. These avionics components are used together to execute control laws for flight control. Hence, the attribute of onboard avionic subsystem is - flight control;
- Remote data terminal: The remote data terminal is situated away from the ground control station and receives annotated video imagery from the VTUAV for supporting scouting, targeting and intelligence dissemination. The automated transmission and reception of video imagery is controlled remotely by the remote data terminal. Hence, the attribute of remote data terminal subsystem is - remote control;
- Data-link: The data-link of a VTUAV provides video, data, and telemetry communications between the air-vehicle, ground terminal station and remote data terminal. The communication is critical for the aircraft manoeuvrability (Flight control) and mission success of the VTUAV. Hence, the attributes of data-link subsystem are - flight control and mission success; and
- Launch and recovery: The launch and recovery component supports automatic takeoff & recovery of the VTUAV and operates on the safety procedures, presented at Sec 2.2.5. It provides VTUAV with a degree of autopilot and contributes a section of mission success. Hence, the attributes of launch and recovery subsystem are - autopilot and mission success.

The subsystems and the identified attributes of the autonomous system are presented in Table 4.6

Table 4.6: Subsystems and Attributes of a Autonomous System

Autonomous subsystems	Attributes
Onboard avionic	• Flight control
Remote data terminal	• Remote control
Data link	• Flight control • Mission success
Launch & recovery	• Autopilot • Mission success

4.2 Synthesis

Synthesis involves the design of the system architecture to address performance and design requirements. It involves the development of block and flow diagrams representing the baseline design. Synthesis also identifies issues in the formulated architecture and suggests corrective measures (Sec 2.1.5).

The functional analysis (4.1) identified mission, vehicle and autonomous systems as the three subsystems of the VTUAV system. The functional characteristics that met the slated mission requirements (Table 3.1) derived from the requirement analysis were also identified. The three subsystems (Sec 4.1) need to be structured as a system to develop the VTUAV design architecture. The architecture will assist in providing a system perspective of the components, attributes, relationships and environment.

4.2.1 VTUAV Design Architecture

The functional analysis (Sec 4.1) resulted in the identification of the subsystems of VTUAV and its attributes (functional characteristics). The subsystems, the elements and their attributes were presented in Tables 4.3, 4.5 and 4.6. The intra and inter relationships between elements and attributes, including elements & elements, attributes & attributes, elements & attributes needs to be determined. The requirement analysis (Sec 3) resulted in the identification of operational environment (Table 3.1) and the operational analysis (Sec 3.2) resulted in the identification of the operational needs that are to be the key inputs for the design and development of the VTUAV which included civil and military needs. The key output of the VTUAV design architecture is a VTUAV that provides the slated mission capabilities (Sec 2.2.4). The summary of the results of the requirement and functional analysis to be used for developing the VTUAV design architecture is presented in Table 4.7.

Table 4.7: Subsystems, Elements, & Attributes of a VTUAV Design System

Input	Environment	Subsystems/ Components	Elements	Attributes	Output
• Civil needs • Military needs	• Terrain • Threat • Time • Littoral • Situation • Weather	Mission	Acoustic	• Observation	VTUAV that provides mission capabilities
			Communication	• Communication • Observation	
			Electro-optical	• Observation • Survivability • Communication	
			Electronic warfare	• Survivability • Navigation • Obstacle detection & avoidance	
			Forward looking & thermal imaging	• Observation • Survivability • Navigation	
			Integrated	• Operator activity	
			RADAR	• Observation	
		Vehicle	Airframe	• Computation • Estimation • Optimisation • Trade-off	
			Power-plant	• Estimation • Computation • Iteration • Trade-off	
			Rotor	• Computation • Estimation • Trade-off	
		Autonomous	Onboard avionic	• Flight control	
			Remote data terminal	• Remote control	
			Data link	• Flight control • Mission success	
			Launch & recovery	• Autopilot • Mission success	

With the identification of the subsystems, elements, its attributes, inputs, outputs and the operational environment, the design architecture of VTUAV is now developed as an input-process-output system configuration (Sec 2.1). The design architecture is presented in Fig 4.2.

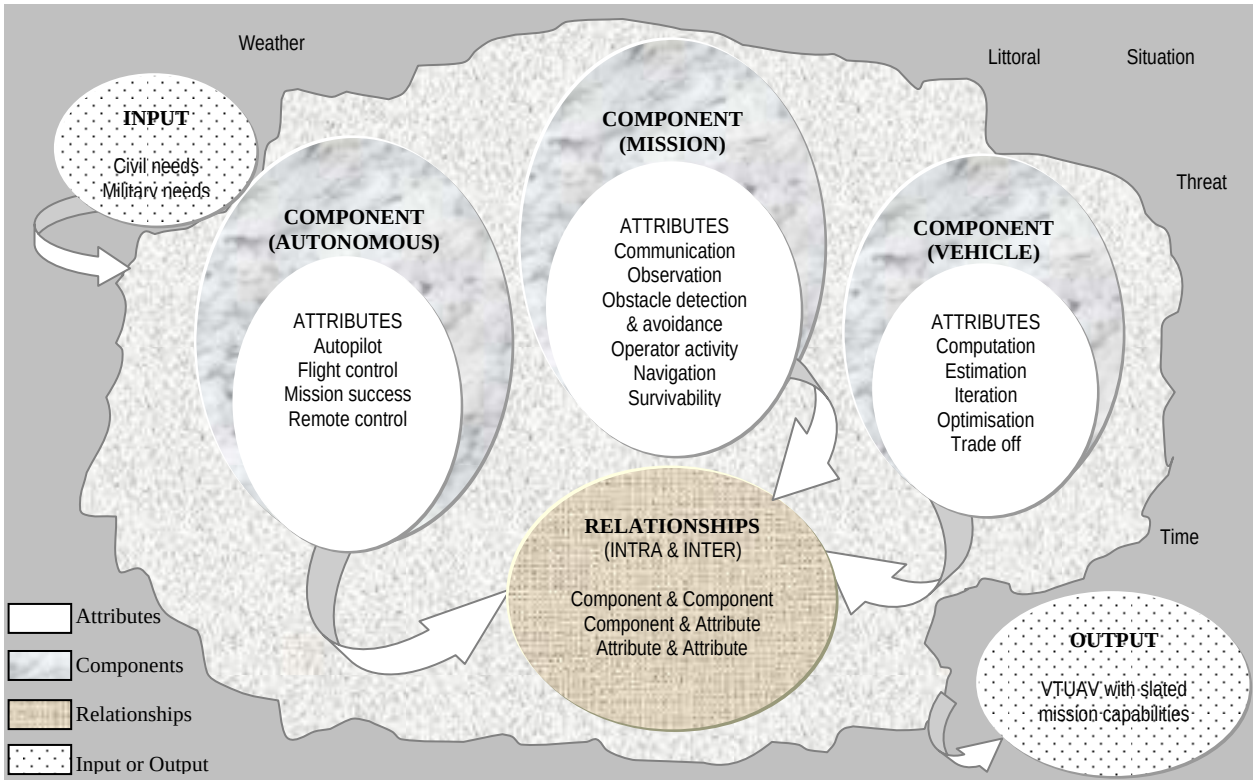


Figure 4.2: VTUAV Design Architecture in an Input-Process-Output Configuration

4.2.2 Refinement of VTUAV Design Architecture

In the VTUAV design architecture developed, three relationships (Sec 4.2.1) have been identified for further detailed analysis. A relationship matrix (Sec 2.1.1) needs to be developed on a matrix format to identify the intra and inter relationships between the components and attributes. The analysis of relationship is addressed by considering the operational requirements and environment conditions from the perspective of mission commonality. The relationship matrix developed for the analysis is presented at Appx D.

The matrix identified relationships between components and attributes of the autonomous, and vehicle & mission systems. The autonomous system attributes of autopilot, flight and remote control have an operational relationship with the vehicle system. There functional characteristics needs to be inherent to the vehicle system. Similarly the autonomous system attribute of mission success needs to be inherent to the mission system that provides the mission capabilities for success.

The VTUAV design architecture (Fig 4.2) can now be refined by considering the system structure with only two components and additional attributes – vehicle and mission system. The refined VTUAV design architecture is presented in Fig 4.3.

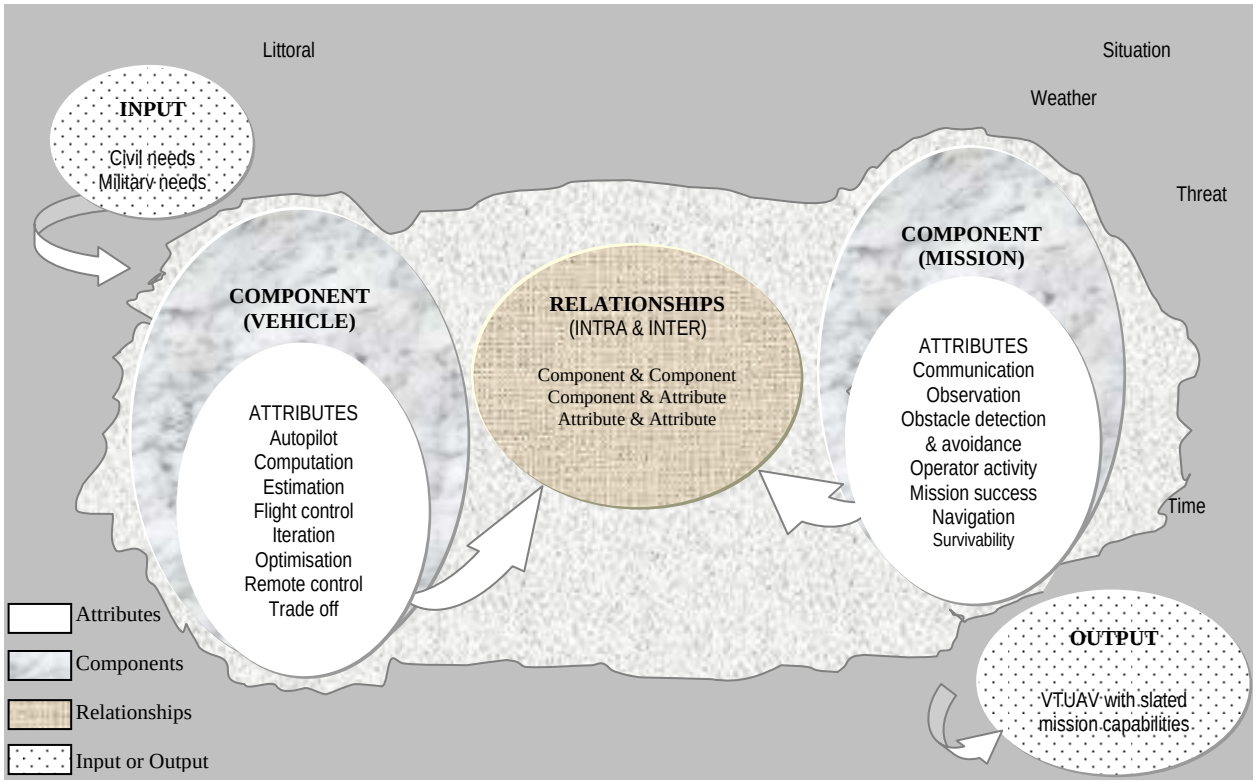


Figure 4.3: Refined VTUAV Design Architecture in an Input-Process-Output Configuration

5 DESIGN PROCESS

The refined design architecture (Sec 4.2.2) provides the base structure for the development of a design process for the VTUAV. The design process needs to consider the identified components and attributes to define the design parameters and constraints (Sec 2.3.2) for the VTUAV to be designed. The stages of the design process needs to comprise of inputs for design analysis and a database (Sec 2.4) for statistical comparison with VTUAVs under development and in-service. The statistical comparison is to establish the benchmarks of design parameters and constraints. Additionally, an updated analytical conceptual design process for the VTUAV will be required. Thus, the design process of a VTUAV will involve the following three stages:

- Design analysis: The attributes of the components of the refined design architecture will be investigated to identify the design parameters that represent its functional characteristics. These will serve as inputs in the design process. The investigation will comprise of following:
  - The attributes of the vehicle system (Fig 4.3) will be analysed to identify the design parameters that represent the flight and autonomous functional characteristics.
  - The attributes of the mission system (Fig 4.3) will be analysed to identify the design parameters that represent the functional characteristics of the mission;
- Design collation: The design process of a VTUAV will involve a series of estimations that refer to past VTUAV design data and statistics. A comprehensive database of VTUAVs in development and in-service needs to be collated to establish the benchmarks of the design parameters for reference during various stages of design; and
- Design development: The traditional rotary-wing analytical design methodology (Sec 2.3.2) will be updated to include the autonomous component in the design.

5.1 Design Analysis

This stage of design process needs to provide the inputs (Fig 4.3) to the design architecture that are to be identified from the attributes of vehicle and mission components (Sec 4.2.2) presented at Table 5.1. These inputs are to be the design parameters that represent the functional characteristics of the vehicle and mission components.

Table 5.1: Attributes of Vehicle and Mission Components identified in the Refined Design Architecture of VTUAVs

Attributes (Vehicle Component)	Attributes (Mission Component)
• Estimation • Computation • Iteration • Optimisation • Trade off • Flight control • Remote control • Autopilot	• Communication • Observation • Obstacle detection & avoidance • Operator activity • Navigation • Survivability • Mission success

- Vehicle attribute analysis: The vehicle attributes are to provide the flight and autonomous functional characteristics as follows:
  - *Estimation:* It is an attribute of power-plant, airframe and rotor subsystems. The various inputs as initial estimations (Sec 2.3.2 & Sec 4.1.2) in the design of these subsystems based on the past design data (Data collation) are - power; gross weight; speed; range; ceiling; and endurance;
  - *Computation:* It is an attribute of power-plant, airframe and rotor subsystems. The various inputs required to refine the estimated values through computation (Sec 2.3.2 & Sec 4.1.2) are - gross weight; speed; range; ceiling; endurance; and rotor diameter;
  - *Iteration:* It is an attribute of power-plant subsystem that needs to be iterated (Sec 2.3.2 & Sec 4.1.2) to refine the estimated value and includes - power;
  - *Optimisation:* It is an attribute of airframe subsystem that finally refines the estimated value through optimisation (Sec 2.3.2 & Sec 4.1.2) and includes - gross weight;
  - *Trade-off:* It is an attribute of power-plant, airframe and rotor subsystems. The various inputs required to perform tradeoff (Sec 2.3.2 & Sec 4.1.2) analysis between the estimated, computed & iterated values are - power; speed; range; ceiling; endurance; fuselage size; and rotor diameter; and
  - *Flight control, remote control and autopilot:* These are the attributes of onboard avionics, data-link, remote data terminal, and launch & recovery subsystems. The input required to control the flight remotely or by an autopilot (Sec 4.2.2) is - operational ceiling.
- Mission attribute analysis: The mission attributes are to provide the functional characteristics of the mission (Sec 4.2.2) as follows:
  - *Communication, observation, obstacle detection & avoidance, operator activity, navigation, and survivability:* These are integrated attributes of acoustics, electro-

optical, electronic warfare, forward looking & thermal imaging and radar subsystems/components (Sec 4.2.2 & Table 4.3) that form the mission payload (Fig 4.1). Its selection as inputs is determined by the mission capabilities it provides and its weight & dimensions, including the requirement of stub-wings for external mounted payload.; and

- *Mission success:* This is also an integrated attribute of all the mission system components. It includes an additional input of mission-time computed from the mission profile (Sec 3.4).

Additionally, the design of vehicle and mission components and its subsequent operation needs to meet the safety regulations of UAVs (Sec 2.2.5). Accordingly the input needs to be the relevant sections of Civil Aviation Safety Regulation (CASR) part 101 or Federal Aviation Regulation (FAR) part 101 as-an-whens developed.

The design parameters that represent the functional characteristics of the vehicle and mission components and are the inputs in the design process are presented in Table 5.2.

Table 5.2: Design Parameters (Inputs) identified from the Analysis of Attributes of Vehicle & Mission Components

Vehicle component	Attributes	Estimation	Computation	Iteration	Optimisation	Trade off	Flight control	Remote control	Autopilot
	Design Parameters (Inputs)	• *Power* • *Gross weight* • *Speed* • *Range* • *Ceiling* • *Endurance*	• *Gross weight* • *Speed* • *Range* • *Ceiling* • *Endurance* • *Rotor diameter*	• *Power*	• *Gross weight*	• *Power* • *Speed* • *Range* • *Ceiling* • *Endurance* • *Fuselage size* • *Rotor diameter*	• *Ceiling*	• *Ceiling*	• *Ceiling*
Mission component	Attributes	Survivability	Observation	Navigation	Communication	Obstacle detection & avoidance		Operator activity	Mission success
	Design Parameters (Inputs)	• *Payload weight* • *Payload dimensions* • *Stub-wing requirement*	• *Payload weight* • *Payload dimensions* • *Stub-wing requirement*	• *Payload weight* • *Payload dimensions*	• *Payload weight* • *Payload dimensions*	• *Payload weight* • *Payload dimensions* • *Stub-wing requirement*		• *Payload weight* • *Payload dimensions*	• *Mission time*
Vehicle & Mission components	Attributes	Regulations							
	Design Parameters (Inputs)	• *CASR part 101* • *FAR part 101 (as-an-whens developed)*							

5.2 Design Collation

A comprehensive database of VTUAVs in-development and in-service needs to be prepared in accordance to the identified design parameters (Table 5.2). The database will establish the design benchmarks for reference during various design stages. The database of the design parameters (specifications) of various VTUAVs is provided at Appx E and that of power-plants is at Appx F (Sec 2.2.3).

5.2.1 Validation of Design Parameters

The design parameters provide inputs to the refined design architecture of the VTUAV. These design parameters needs to be validated against statistically derived benchmarks prior to being used as inputs. A statistical investigation of design specifications of VTUAVs is required to establish equations that compute the benchmarks. The process developed and its results are as follows:

- Process: The process involves regression analysis (Sec 2.4.3) of VTUAV design specification on a spreadsheet application – Microsoft Excel. The regression analysis of the specification generates linear or non-linear polynomial plots on which an array of points is plotted automatically by the TREND function (Sec 2.4.3) to develop the mathematical equations. The validity of the equation is determined from the R-squared (R^2) value of 0.70 or more. A corrective factor may also be associated to increase the R^2 value to refine the results.
- Results: The above process resulted in the establishment of following equations between various design parameters:
  - *Payload & maximum take-off weight:* To generate a set of mathematical equations that computes payload weight from maximum take-off weight, several records of maximum takeoff, payload weight, and empty weight were collated (Appx G) based on the CASR UAV categorisation (Sec 2.2.5) of micro, small, and large UAVs. The regression analysis of collation records (Appx G) resulted in non-linear polynomial plots for micro, small and large UAVs validated by the acceptable R^2 value of 0.9397, 0.8096, and 0.7181 respectively. A refinement factor of 1.064, 1.235 and 1.393 was associated with the expressions to generate acceptable patterns. The mathematical expressions for micro, small and large UAVs classification are as follows and their respective graphical representation is at Fig 5.1, 5.2 and 5.3.
    - Micro UAV

      $$\text{Payload}_{max} = 1.064\ \{0.0177(\text{MTOW})^2 + 0.1132(\text{MTOW}) + 0.085\} \qquad \textbf{Eqn 5.01}$$

- Small UAV

$$\text{Payload}_{max} = 1.235\ \{0.0025(\text{MTOW})^2 + 0.0184(\text{MTOW}) + 3.684\} \quad \textbf{Eqn 5.02}$$

- Large UAV

$$\text{Payload}_{max} = 1.393\ \{1\text{E-}05(\text{MTOW})^2 + 0.1264(\text{MTOW}) + 22.25\} \quad \textbf{Eqn 5.03}$$

Where,

MTOW = Maximum take off weight; and max = maximum.

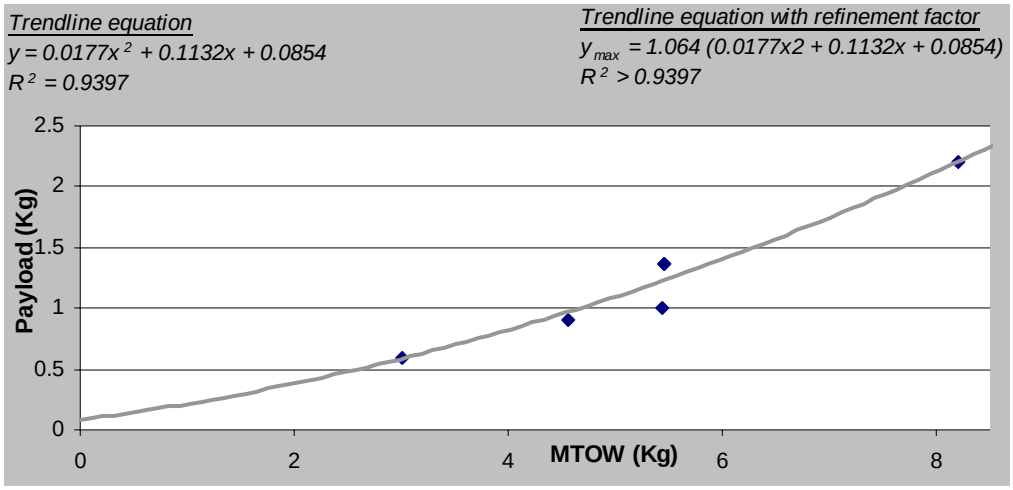


Figure 5.1: Micro UAV – Trendline

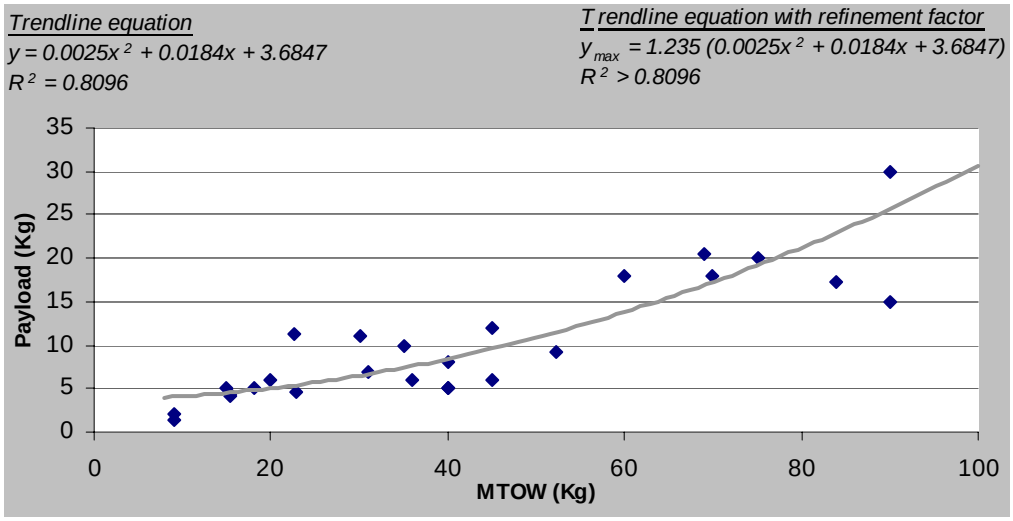


Figure 5.2: Small UAV - Trendline

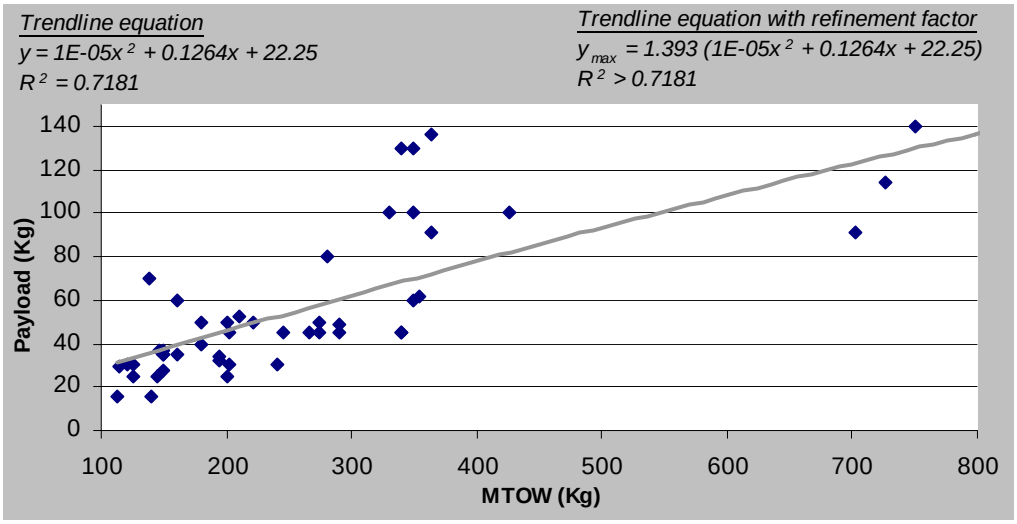


Figure 5.3: Large UAV – Trendline

- *Speed, endurance, payload & maximum take-off weight:* Similar investigations for relationships between other parameters (speed, endurance, payload & maximum take-off weight) based on the design specifications (Appx E) resulted in the following mathematical expressions:
  - Micro, small and large UAVs

    $\{(\text{Speed} \times \text{Endurance} / \text{Range})_{min}\} / \{(\text{Speed} \times \text{Endurance} / \text{Range})_{max}\} = 0.25$

    Eqn 5.04

  - Micro, small and large UAVs

    $\{(\text{MTOW} / \text{Power})_{min}\} / \{(\text{MTOW} / \text{Power})_{max}\} = 0.353$

    Eqn 5.05

    Where,

    MTOW = Maximum take-off weight; min = minimum; and

    max = maximum.

The above equations establish relationships between minimum and maximum limits of endurance and range at maximum forward speed, and of MTOW & power. These relationship values will limit the variance of input parameters during various stages of design process with-in acceptable design limits.

5.3 Design Development

The traditional rotary-wing analytical design methodology (Sec 2.3.2) needs to be updated to incorporate an autonomous component for application in VTUAV design. It will involve a series of sequential estimations, computations, and iterations with reference to past VTUAV design data and statistics as benchmark values.

The design methodology of VTUAV needs to incorporate a set of inputs derived from the operational needs (Sec 3.2) established by the potential customer. These inputs are represented by the design parameters (Table 5.2) to be subsequently classified as follows:

- Design requirements: These are design specifications derived from the operational need; and
- Design constraints: These are the operational & design limitations, and safety standard constraints as stipulated by the design benchmarks, regulations, and past design data.

The classification of the design parameters as requirements and constraints is presented in Table 5.3

Table 5.3: Requirement and Constraint Parameters

		Parameters	Derived from / Stipulated by
Design parameters	**Design requirements**	• P :Power	Operational need
		• W_{MTO} :Maximum take-off weight	
		• W_p :Payload weight	
		• W_e :Empty weight	
		• V_f :Forward speed	
		• R_{cruise} :Cruise range	
		• t_E :Endurance time	
		• Z_{abs} :Altitude (Absolute Ceiling)	
		• T_m :Mission time	
	Design constraints	• L_f :Fuselage length	Design specification
		• B_f :Fuselage breadth (Width)	
		• D :Rotor diameter	
		• L_P :Payload length	
		• B_P :Payload breadth	
		• $CASR_{101}$:Civil aviation safety regulation	Safety standard
		• FAR_{101} :Federal aviation safety regulation	
		• L_{sw} :Stub-wing length	Operational need
		• D_L :Disc loading = 19.53 to 58.59 kg/m^2	Past design data
		• C_D :Drag coefficient = 0.3	
		C_T/σ :Coefficient of thrust to solidity ratio = 0.17 to 0.23 m	
		• R/C :Radius-to-chord ratio = 8.0 to 24.0	
		• c :Blade chord = 0.0127 m	
		• μ :Tip speed ratio = 0.5	

With the establishment of the design requirements and constraints, the traditional rotary-wing analytical design methodology (Sec 2.3.2) needs to be updated for the development of

conceptual design configuration of the VTUAV. The design configuration is to meet the slated mission requirements (Table 3.1). The series of estimations, computation, and iteration stages for the conceptual design of a VTUAV are as follows:

- Stage 1 – Mission time computation: The mission time is estimated as an aggregate of time periods for various mission phases, which are identified from the slated mission requirements and represented by the mission profile (Sec 3.4);
- Stage 2 – Payload selection: The mission system technology (Sec 4.1.1) that meets slated mission requirements is selected from the systems collated at Appx A to form the payload. This is based on the functional characteristics of the mission system. The individual weights of selected mission systems are aggregated to compute payload weight. The dimensions of the mission systems are also recorded to evaluate the payload size;
- Stage 3 – Database search and power estimation: Based on the design requirements, VTUAVs of similar characteristics are sorted on weight, speed, endurance, range and altitude to estimate the power (P) required. The model and number of the engines required is also identified;
- Stage 4 – Fuel computation: The weight of total fuel required for the mission is computed from Eqn 2.1 based on the specific fuel combustion (sfc), power and mission time. Reserve fuel of 10% and trapped fuel in pipeline of 4% is also included in the computation;
- Stage 5 – MTOW breakdown: The engine weight is estimated at 15% of the required MTOW of the class of vehicle to be designed based on the relative weight analysis of powerplant weight and MTOW of existing VTUAVs (Appx F). The balance weight (structural) comprises of various airframe sections of the VTUAV. The weight of the various sections is evaluated subsequently;
- Stage 6 – Vertical drag calculation: The vertical drag is computed from Eqn 2.4 based on the disc loading of similar class of VTUAV and the projected area (A_p) of the VTUAV in remote wake. The projected area is evaluated by developing the outboard profile of the fuselage from the inboard profile (layout of the payload) and the configuration (conventional/coaxial) selected;
- Stage 7 – Main rotor design: It involves the evaluation of the number of main rotor blades required based on the rotor radius, chord (constraint) and solidity (Eqn 2.9). The rotor radius is evaluated from the disc area that is computed from vertical drag and disc loading by Eqn 2.5. Solidity is computed from 'thrust coefficient to solidity' ratio (constraint) at sea level (Eqn 2.8). The tip-speed is evaluated based on the forward speed specification & tip-speed ratio constraint (Eqn 2.6 & 2.7). The airfoil selection is also addressed;

- Stage 8 – Drag in forward flight: The coefficient of parasite drag of a VTUAV with internal and external loads is evaluated from MTOW, coefficient-of-friction, form factor, and size of external load (Eqn 2.11 & 2.12). The coefficient of profile drag is evaluated from the average angle-of-attack (constraint) by Eqn 2.14;
- Stage 9 – Power required to sustain forward flight: The power required to sustain forward flight at design maximum speed and at a specific altitude is computed by evaluating the parasite, profile and induced power (Eqn 2.15). The miscellaneous power required to compensate for mechanical losses is considered as 15% of the total power required; and
- Stage 10 – Weight & balance and refinement of gross weight: The balance weight (structural) of the airframe sections - rotor blades, propulsion, electrical, avionics, optional nacelles and dry systems (Sec 2.2.3) of the VTUAV is computed (Eqn 2.17 to Eqn 2.23). This is followed by the weight and balance exercise. Further iteration was not addressed.

6 AUTOMATION

The design methodology developed (Sec 5.3) needs to be automated to design an expert system based on ‘flexible automation’ principle (Sec 2.4) to generate the conceptual design configuration of the VTUAV from the slated mission requirements (Table 3.1). The expert system to be developed to address automation of the conceptual design methodology needs to display the conceptual design configuration on a Graphical User Interface (GUI).

The design of the expert system requires the selection of a suitable programming language and the database in a programming environment to implement the supervisory and developmental logic of the design methodology. This selection is to be based on inter-compatibility as follows (Sec 2.4.2):

- Programming environment: Microsoft Visual Studio .Net was selected to provide the dynamic environment for implementing the design methodology developed due to its modular and object-oriented base;
- Programming language: Microsoft Visual C++ was selected from the various inbuilt languages of Microsoft Visual Studio .Net to program required modules of the design methodology as ‘reusable classes’ due to its system and web interaction ability; and
- Database: Microsoft SQL Server was selected to be the relational database management system for its ability to interact with Microsoft Visual C++ to analyse and store exchanges through its administrative tools of database analysis, maintenance, and administration. The connectivity between Microsoft Visual C++ and Microsoft SQL Server was established through a customised portable file-connection string. The application consistency across windows and web-based platforms was maintained with Microsoft .NET Framework 1.1, an inherent feature of Microsoft Visual Studio .Net environment (Sec 2.4.2).

6.1 Expert System - Design

The design methodology of VTUAVs needs to be analysed to identify various reusable classes that provide computational details of the design stages (Sec 5.3). These classes, their variables and functional methods will provide the base design for the expert system. Accordingly the classes need to be categorised.

6.1.1 Base Category

The various stages of the design methodology for VTUAV design were analysed to identify the classes for the base design category of the expert system. The investigation established the functionality of the various design stages to be distinctive, and derived from common inputs. Thus, each stage of the design methodology can be represented by a "variable-dependent" and "method-independent" class. The investigation also established the class of design-input (requirements and constraints) needs at Stage 1, and defined design-output at Stage 10 of the design methodology (Sec 5.3). Thus, the various classes, their corresponding variables and methods based on base category and identified by the analysis of the design methodology stages are as follows:

- 'Design-input' class: The class provides validated inputs to the VTUAV design methodology. The class variables are the design requirements and constraints (Sec 5.1) identified from the attributes of the components of the refined design architecture (Fig 4.3). The class methods comprise of acquiring design requirements & constraints, and validating specific design parameters including payload & maximum take-off weight; speed, endurance & range; and power & maximum take-off weight against the established relative benchmarks (Sec 5.2.1);
- 'Mission-time' class: The class estimates and validates the time required to complete the slated mission. The class variables are the generic mission phases - take-off, transit, ingress, engage, disengage, egress, transit back, and landing (Sec 3.4). The class methods include an aggregate of time periods of the mission phases to evaluate the mission time, and the validation of mission time against the slated endurance time (design input). Since the class involves mission requirements (Table 3.1), there are 23 independent "mission time" classes required to compute mission time based on the mission requirements. The variables of all these "mission time" classes are similar but for 'engage' and 'disengage' class variables that are mission specific;
- 'Payload' class: The class involves the selection of mission systems that meet slated mission requirements. The class variables are the various mission systems collated at Appx A. The class methods involve the addition of the selected mission systems weights to compute the payload weight. Since the class involves the analysis of the mission requirements that are the functional characteristics of the components of mission systems (Table 4.1), seven classes are to be designed based on the elements of the components of the mission system. The variables of these classes are the elements of the components of the mission system (Fig 4.1);

- ‘Database-I’ class: The class involves identification of VTUAVs based on design requirements for the estimation of power. The class variables are the design requirements comprising of power, and also include the make/model and number of the engines. The class methods include matching of the design requirements with database for similar VTUAVs, and selection of engine based on the model and number of engines onboard these VTUAVs;
- ‘Fuel’ class: The class involves estimation of mission fuel. The class variables are power, specific fuel combustion (sfc), mission time, and mission fuel. The class method includes computation of weight of total fuel required for the mission including the reserve and trapped fuel in lines;
- ‘Structural weight’ class: The class involves evaluation of the structural weight. The class variables are MTOW, payload and engine weight. The class method includes the evaluation of remainder weight of various airframe sections of the VTUAV based on engine weight as 15% of the MTOW;
- ‘Vertical drag’ class: The class calculates the vertical drag and involves development of inboard and external profile of the VTUAV based on mission systems load stations, governed by its functional characteristics. The class variables are drag coefficient, disc loading, projected area in remote wake, and dimensions of selected mission systems. The class methods comprise of calculation of the vertical drag based on the computation of projected area in remote wake from the external profile developed of the VTUAV;
- ‘Rotor system-I’ class: The class involves the computation of the size of main rotor system and the number of rotor blades based on the coaxial configuration; other configurations have not been addressed being an illustration. The class variables are MTOW, vertical drag, disc loading, forward speed, rotational speed, & tip speed, and also include the air density. The class methods comprise of computation of the rotor radius from disc area, and computation of solidity to determine the number of rotor blades;
- ‘Rotor system-II’ class: The class involves the selection of blade airfoil. The class variables in airfoil design include the type (symmetrical/cambered) & thickness of airfoil section, and the positioning of maximum camber. The class method comprises of selection of the airfoil section;
- ‘Drag’ class: The class involves evaluation of parasite and profile drag for further computation of power required to sustain forward flight at maximum speed. The class variables include MTOW, payload weight, dialog factor, and coefficient of friction. The class methods comprises of evaluation of the coefficients of parasite and profile drag;

- ‘Power’ class: The class involves the evaluation of power required in forward flight for maximum speed. The class variables include MTOW, air density, solidity, rotor radius & disc area, forward speed, coefficient of profile drag, angular velocity, and tip speed. The class method comprises of power evaluation at maximum speed in forward flight;
- ‘Database-II’ class: The class involves the reselection of engine based on significant variance between the evaluated power at previous class and estimated power at ‘Database-I’ class. The class variables include evaluated and estimated power. The class methods comprise of comparison of evaluated power with the estimated power and reselection of the engine from the database;
- ‘Airframe-section weight’ class: The class involves computation of the structural weight of the airframe sections including rotor blades, propulsion, electrical, avionics, optional nacelles and dry systems of VTUAV. The class variables include MTOW, fuselage length & projected area, and engine weight. The class methods involve computation of structural weights of airframe sections of VTUAV for their comparison with the evaluated structural weight at ‘Structural weight’ class;
- ‘Balancing’ class: The class involves the ‘weight and balance’ exercise to compute the longitudinal cg. The class variables include the load stations of airframe sections and mission systems. The class methods comprise of evaluation of longitudinal cg; and
- ‘Design-output’ class: The class involves the refinement of inboard and external profile of the VTUAV. The class variables include the selected mission systems and the airframe sections. The class method comprises of refinement of inboard and external profiles of the VTUAV.

6.1.2 Database Category

The design of the expert system involves another category of classes; the databases of the design parameters (Table 5.2) of various VTUAVs in-development/in-service (Appx E) and their power-plants (Appx F). The databases are used in the design methodology to refer to past VTUAV design data for estimations. It is also used for information collection, and comparisons between various VTUAVs. These databases are represented by editable and non-editable classes as follows and comprises of ‘search’ functionality for the analysis of collected information:

- ‘Non-editable database’ class: The class provides ‘read-only’ facilities of the design parameters of various VTUAVs. The class variables include design parameters and the class methods comprise of ‘search and sort’ function as required by the designer; and

- ‘Editable database’ class: The class provides the option to update the collated VTUAV design parameters. The class variables include design parameters and the class methods comprise of update of edited design parameters.

6.1.3 Utility Category

Additional category of classes is identified based on the need to execute and record interim computations involved in the design methodology of VTUAVs. These classes based on utility category comprise of operating-system classes including “Calculator”, “Notepad”, and “WordPad” to execute interim computations. To record these interim computations, classes that mail and print the design-configurations of the VTUAV are identified. The operating system class of “Send-by-mail” provides mailing functions, but an independent class is designed as follows to provide printing functions:

- ‘Print’ class: The class involves print of the database and the design configuration of VTUAV. The class variables include page-setup, page-copies, and page-orientation. The class method comprises of printing of the design-configuration/database of VTUAVs.

6.1.4 Navigation Category

There is a need for a category of classes to address the navigation between ‘mission-time’ classes. Hence, classes based on navigation category are designed on the mission requirements (Table 3.1) as follows:

- ‘Mission categories’ class: The class provides the option to select the desired category of mission requirements based on the operational needs. The class variables are various mission categories including generic, relief & monitoring, research, strategic, and tactical. The class method comprises of navigation to mission requirement based on the selected mission category; and
- ‘Mission requirements’ class: The class involves various mission requirements based on the selected mission category. The class variables are mission requirements and the class method comprises of navigation to mission phases of selected mission requirement for computation of mission time.

6.1.5 Framework Category

The development of the application framework in .NET (Sec 2.4.2) programming environment will comprise of another category of classes. These classes are based on the internal framework of Microsoft Visual C++ involving “application”, “view”, and

"document" classes. The class variables include global/private variables that are used in 'base classes'. The class methods comprises of user-application interactions as follows:

- 'Application' class: It responds to menu and taskbar interactions;
- 'View' class: It responds to interactions involving input devices including mouse and keyboard; and
- 'Document' class: It involves database usage vis-à-vis configured connection string.

A total of 27 classes were identified based on the five categories of base, database, utility, navigation, and framework for the automation of the design methodology. The various classes and its corresponding variables and methods are presented at Table 6.1.

Table 6.1: Classes, Variables & Methods for the Design of the Expert System based on the Design Methodology of VTUAVs

Category	Class	Definition	Variables		Methods
Base	*Design-input*	Validation of inputs to VTUAV design methodology	• Design requirements	• Design constraints	• Acquire • Validate
	Mission-time	Estimation & validation of time required to complete slated mission	• Mission phases		• Acquire • Aggregate • Validate
	Payload	Aggregation of selected mission systems weights to compute the payload weight	• Mission systems		• Select • Aggregate
	Database-I	Identification of VTUAVs of similar characteristics as design requirements for estimation of power	• Design requirements		• Match • Select
	Fuel	Estimation of mission fuel	• Power • sfc	• Mission time • Mission fuel	• Compute
	Structural weight	Evaluation of structural weight	• MTOW • Payload weight	• Engine weight	• Evaluate
	Vertical drag	Calculation of vertical drag involving development of inboard & external profile of VTUAV based on the mission systems load stations	• Drag coefficient • Disc loading	• Projected area • Mission system dimensions	• Compute • Develop
	Rotor system-I	Computation of size of main rotor system and number of rotor blades based on coaxial configuration	• MTOW • Vertical drag • Disc loading • Forward speed	• Rotational speed • Tip speed • Air density • Rotor configuration type	• Compute • Evaluate
	Rotor system-II	Selection of blade airfoil	• Section type • Section thickness	• Camber • Camber positioning	• Select
	Drag	Evaluation of parasite & profile drag for further computation of power required to sustain forward flight at maximum speed	• MTOW • Payload weight	• Dialog factor • Coefficient of friction	• Evaluate
	Power	Evaluation of power required in forward flight for maximum speed	• MTOW • Air density • Solidity • Rotor radius • Disc-area	• Forward speed • Coefficient of profile drag • Angular velocity • Tip speed	• Evaluate
	Database-II	Variance based reselection of engine	• Evaluated power	• Estimated power	• Compare • Match • Select

	Airframe-section weight	Computation of balance weight (structural) of airframe sections - rotor blades, propulsion, electrical, avionics, optional nacelles and dry systems of VTUAV	• MTOW • Fuselage length	• Projected area • Engine weight	• Compute • Compare
	Balancing	Computation of longitudinal cg from 'Weight & balance' exercise	• Airframe sections	• Mission systems	• Evaluate
	Design-output	Refinement of inboard & external profile of VTUAV based on repositioning of mission systems	• Airframe sections	• Mission systems	• Refine
Database	*Non-editable database*	Display of 'read-only' facilities of design parameters of various VTUAVs	• Design parameters		• Match
	Editable database	Update of collated VTUAV design parameters	• Design parameters		• Update
Utility	*Calculator*	Execution of supplementary computations	• User-based		• Calculate
	Notepad	Archive of small observations	• User-based		• Record
	WordPad	Archive of large observations	• User-based		• Record
	Send–by-mail	Accessibility of output for reference & archiving	• User-based		• Access • Archive
	Print	Printing of design parameters of VTUAVs database & resulted design configuration of VTUAV	• Page-setup • Page-copies	• Page-orientation	• Print
Navigation	*Mission categories*	Selection of category of mission requirements based on operational needs	• Generic • Research • Tactical	• Relief & monitoring • Strategic	• Navigate
	Mission requirements	Display of various mission requirements based on selected mission category	• Mission requirements		• Navigate
Framework	*Application*	Interactions of user with menu-bar & task-bar	• Global variables		• Interact
	View	Interactions of user with input devices including mouse & keyboard	• Global variables		• Interact
	Document	Usage of database & development of connection string	• Global variables		• Usage • Develop

6.2 Expert System - Dialog

The various reusable classes obtained from the analysis of computational details of the design stages (Sec 5.3), provide the base design for the expert system. These reusable classes including those based on database, utility, navigation, and framework category are programmed as a dialog to achieve the desired functionality.

The dialogs to be developed will need to comprise of various controls including buttons, sliders, option-boxes, check-boxes, text-boxes, and data-grids to implement class methods based on the class variables. The requirements of controls for the various dialogs are as follows:

- ‘Design input’ dialog: Option-boxes and text-boxes to validate the acquired design requirements and constraints;
- ‘Mission-time’ dialog: Text-boxes to compute the time required to complete slated mission requirements;
- ‘Payload’ dialog: Check-boxes to aggregate the selected mission systems weight to compute the payload weight;
- ‘Database-I’ dialog: Data-grids and text-boxes to identify VTUAVs based on the design requirements;
- ‘Fuel’ dialog: Text-boxes to evaluate the mission fuel;
- ‘Structural weight’ dialog: Text-boxes to evaluate the structural weight;
- ‘Vertical drag’ dialog: Text-boxes to evaluate the vertical drag;
- ‘Rotor system-I’ dialog: Text-boxes to compute the number of rotor blades based on coaxial configuration;
- ‘Rotor system-II’ dialog: Text-boxes and sliders to select the blade airfoil;
- ‘Drag’ dialog: Text-boxes to evaluate the coefficients of parasite and profile drag;
- ‘Power’ dialog: Text-boxes to evaluate the power required in forward flight for maximum speed;
- ‘Database-II’ dialog: Text-boxes to reselect the engine;
- ‘Airframe-section weight’ dialog: Text-boxes to compute the structural weight of various airframe sections;
- ‘Balancing’ dialog: Text-boxes to implement the ‘Weight and balance’ exercise to place the cg within the acceptable limits;
- ‘Design-output’ dialog: Text-boxes to generate the inboard and external profile of the VTUAV;

- ‘Non-editable database’ dialog: Data-grids and text-boxes to provide the read-only facilities of the design parameters of various VTUAVs in-development/in-service and their power-plants;
- ‘Editable database’ dialog: Data-grids and text-boxes to provide the update facilities of the design parameters of various VTUAVs in-development/in-service and their power-plants;
- ‘Print’ dialog: Text-boxes to print the database and design configuration of the VTUAVs;
- ‘Mission categories’ dialog: Buttons to select the desired category of mission requirements; and
- ‘Mission requirements’ dialog: Buttons to select the mission requirements of the selected mission category.

6.2.1 ‘Design input’ Class

The function of the expert system commences with the validation of the acquired design requirements and constraints by the ‘Design-input’ class. The class output is a dialog that provides text and numerical fields for typing in data based on the data-type (number, alphabet, and alphanumeric) of design requirements (Weight, speed, endurance, ceiling, and altitude) and constraints (Physical dimensions, stub-wing requirement, and CASR part 101 safety regulation). The dialog also validates the data against the relative established benchmarks (Eqn 5.01 to 5.05).

The dialog comprises of several buttons for analysis/information gathering as follows:

- Next: It validates the typed design requirements against the established benchmarks, and navigates to the output dialog of ‘Mission time’ class;
- File: It specifies the established benchmarks between payload, MTOW, speed, endurance, and range involved during the validation;
- Home: It navigates to the ‘welcome screen’ that provides information about the expert system; and
- More Info: It specifies CASA and other safety & design standards (Sec 2.2.5) to the user for further selection.

The dialog and its corresponding buttons as an output of the ‘Design-input’ class are presented in Fig 6.1. The program code of the ‘Design-input’ class involving design requirements and constraints as class variables, and acquire & validate as class methods (Table 6.1) is presented at Appx H.1.

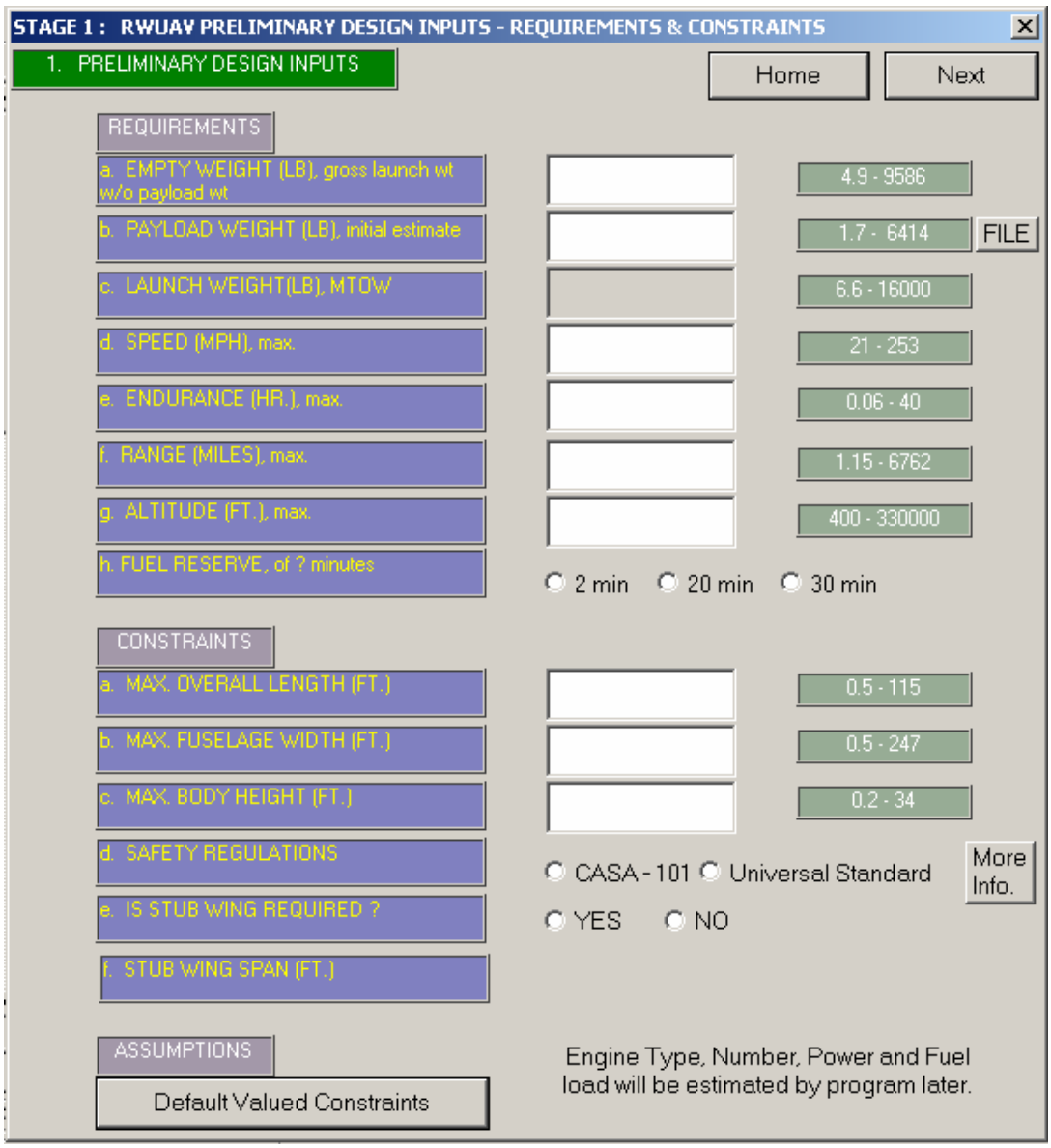


Figure 6.1: The Output Dialog of the 'Design input' Class

6.2.2 'Mission time' Class

With the validation of the design requirements and constraints, the time required to complete slated mission requirements is estimated by the 'Mission-time' class. A total of 23 independent 'Mission-time' classes are programmed based on the various identified mission requirements (Table 3.1). The output of the 'Mission time' class based on the reconnaissance and surveillance mission is discussed below as an illustration.

The dialog comprises of several buttons to integrate the mission phase time and validate these as follows:

- Next: It aggregates the time acquired from the designated numerical fields of various mission phases of reconnaissance and surveillance mission, and then navigates to the output dialog of 'Payload' class; and
- Prev: It opens the output dialog of 'Design input' class.

The dialog and its corresponding buttons as an output of the 'Mission time' class are presented in Fig 6.2. The program code of the 'Mission time' class involving mission phases as class variables, and acquire, aggregate, & validate as class methods (Table 6.1) is presented at Appx H.2.

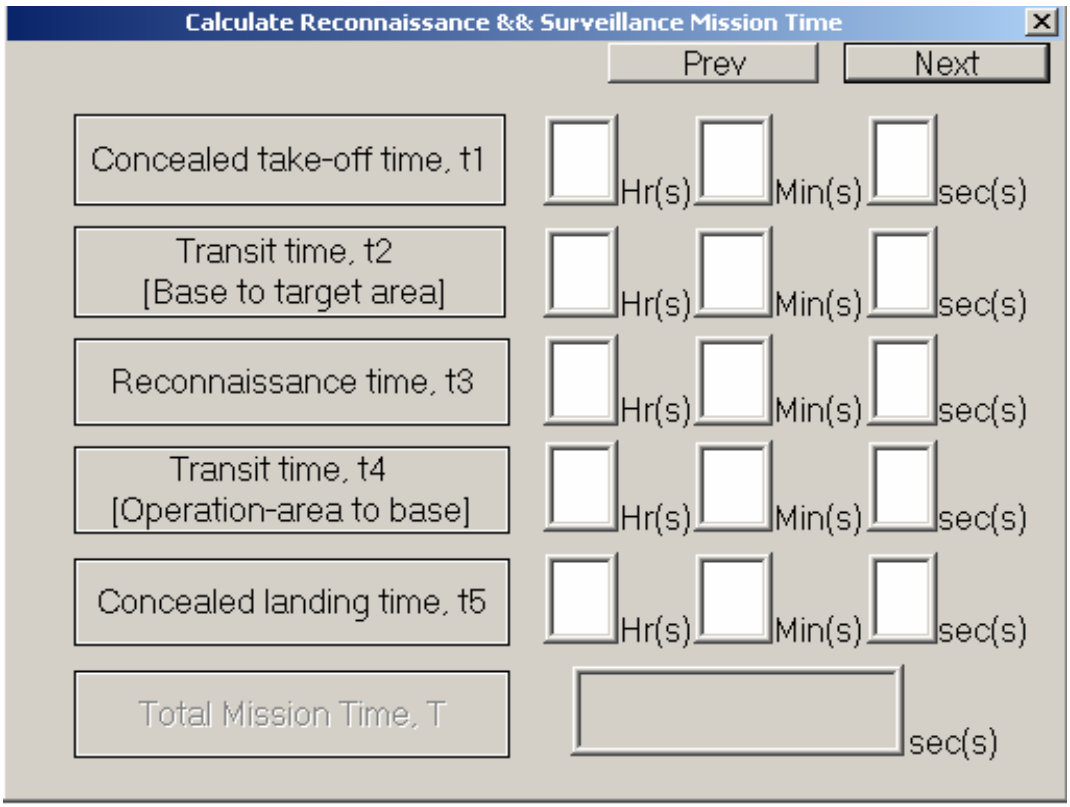


Figure 6.2: The Output Dialog of 'Mission-time' Class based on Reconnaissance and Surveillance Mission

6.2.3 'Payload' Class

The 'Payload' class involves addition of selected mission systems weights to compute the payload weight. Six payload classes are designed instead of proposed seven (Sec 6.1.1) due to single mission system with the acoustic sensor component – Sensor array element (Appx A). The output of 'Payload' class based on the acoustic-sensor and communication components of mission systems is discussed below as an illustration.

The dialog comprises of ‘checkboxes’ to select the mission systems based on the analysis of the slated mission requirements. It also includes several buttons to integrate mission system weight as follows:

- Next: It aggregates the weights of the selected mission systems and navigates to the output dialog of ‘Database-I’ class; and
- Prev: It navigates to the output dialog of ‘Mission time’ class.

The dialog as an output of the ‘Payload’ class based on acoustic-sensor and communication components of mission systems is presented in Fig 6.3. The corresponding program code involving mission systems of acoustic-sensor and communication components as class variables, and select, & aggregate as class methods (Table 6.1) is presented at Appx H.3.

The dialogs of other components of mission system including electro-optical, electronic warfare, forward looking thermal imaging, integrated, and RADAR are presented in Fig 6.4 to 6.8.

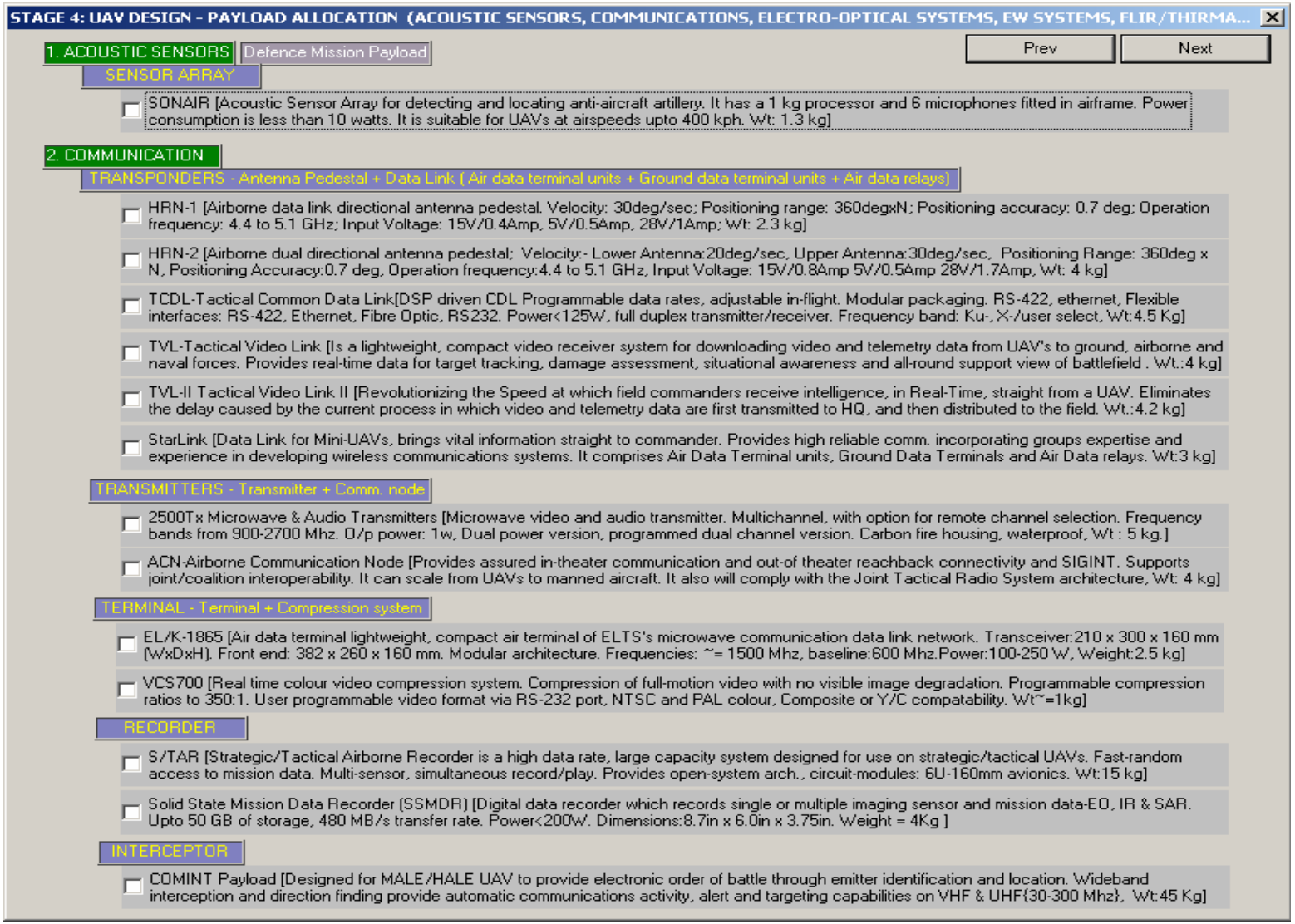


Figure 6.3: The Output Dialog of 'Payload' Class of Mission Systems – Acoustic Sensors and Communication Components

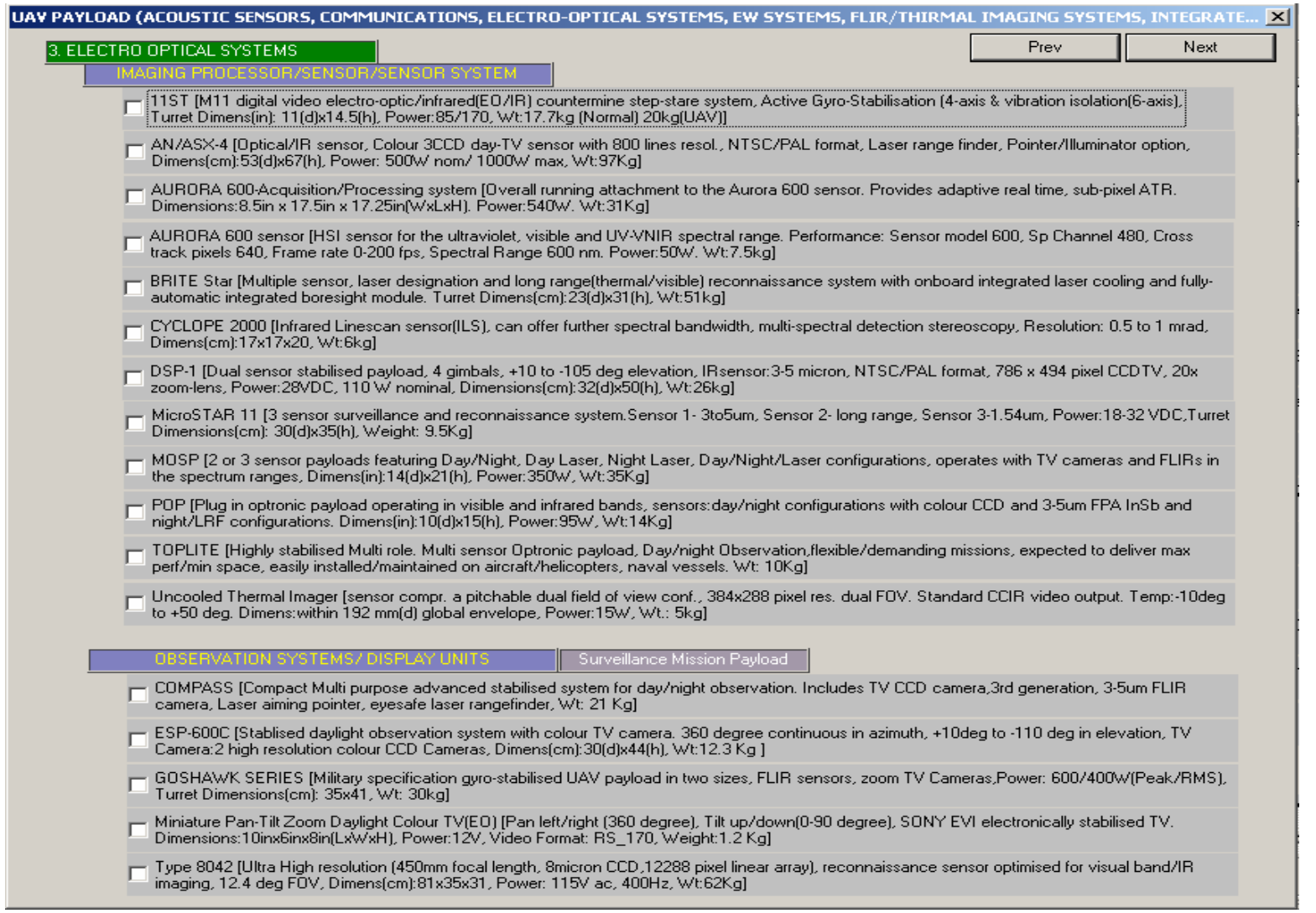


Figure 6.4: The Output Dialog of 'Payload' Class of Mission Systems – Elements of Electro-optical Component (Imaging Processor with Sensor System and Display & Observation Unit)

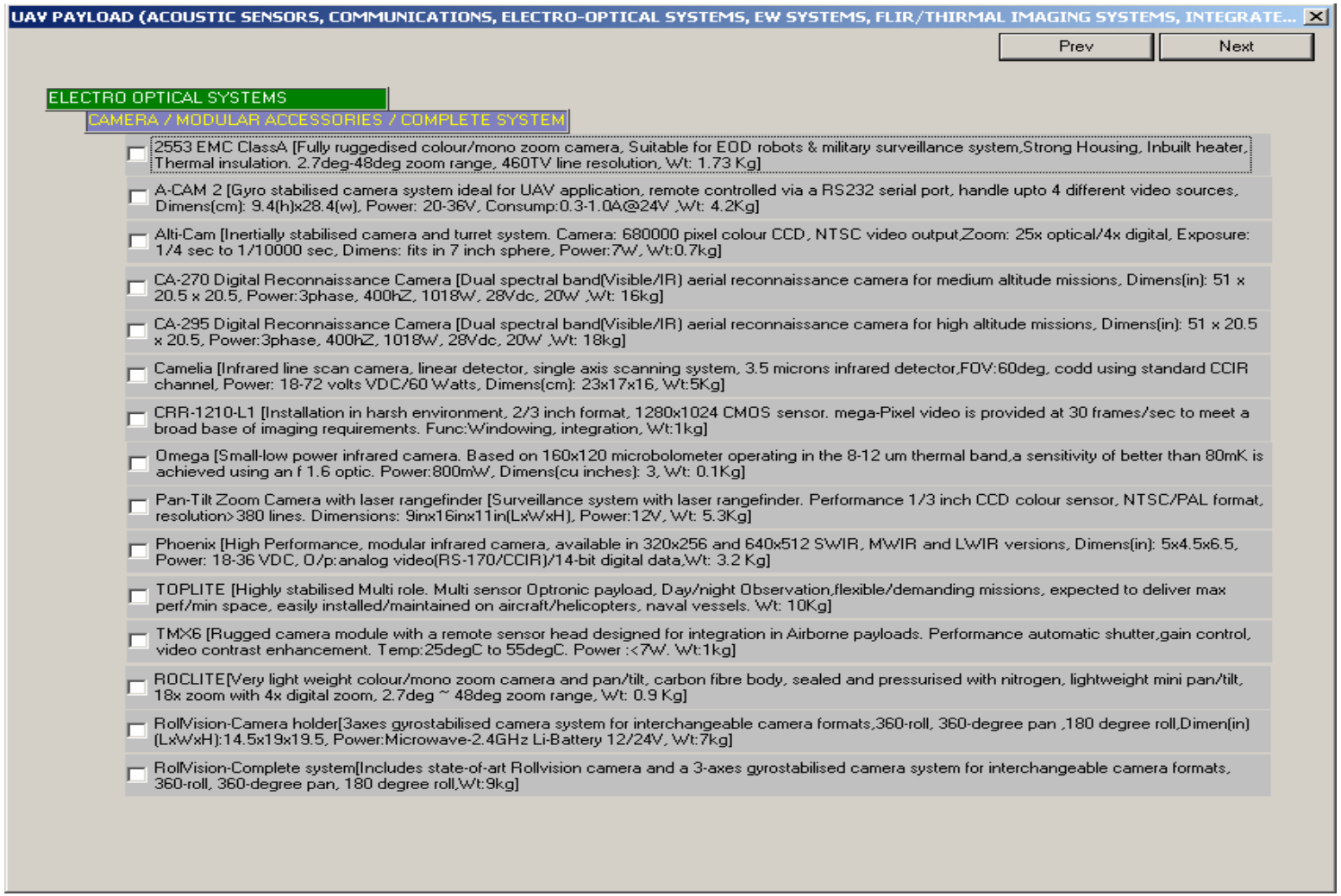


Figure 6.5: The Output Dialog of 'Payload' Class of Mission Systems – Element of Electro-optical Component (Camera & Modular Accessories)

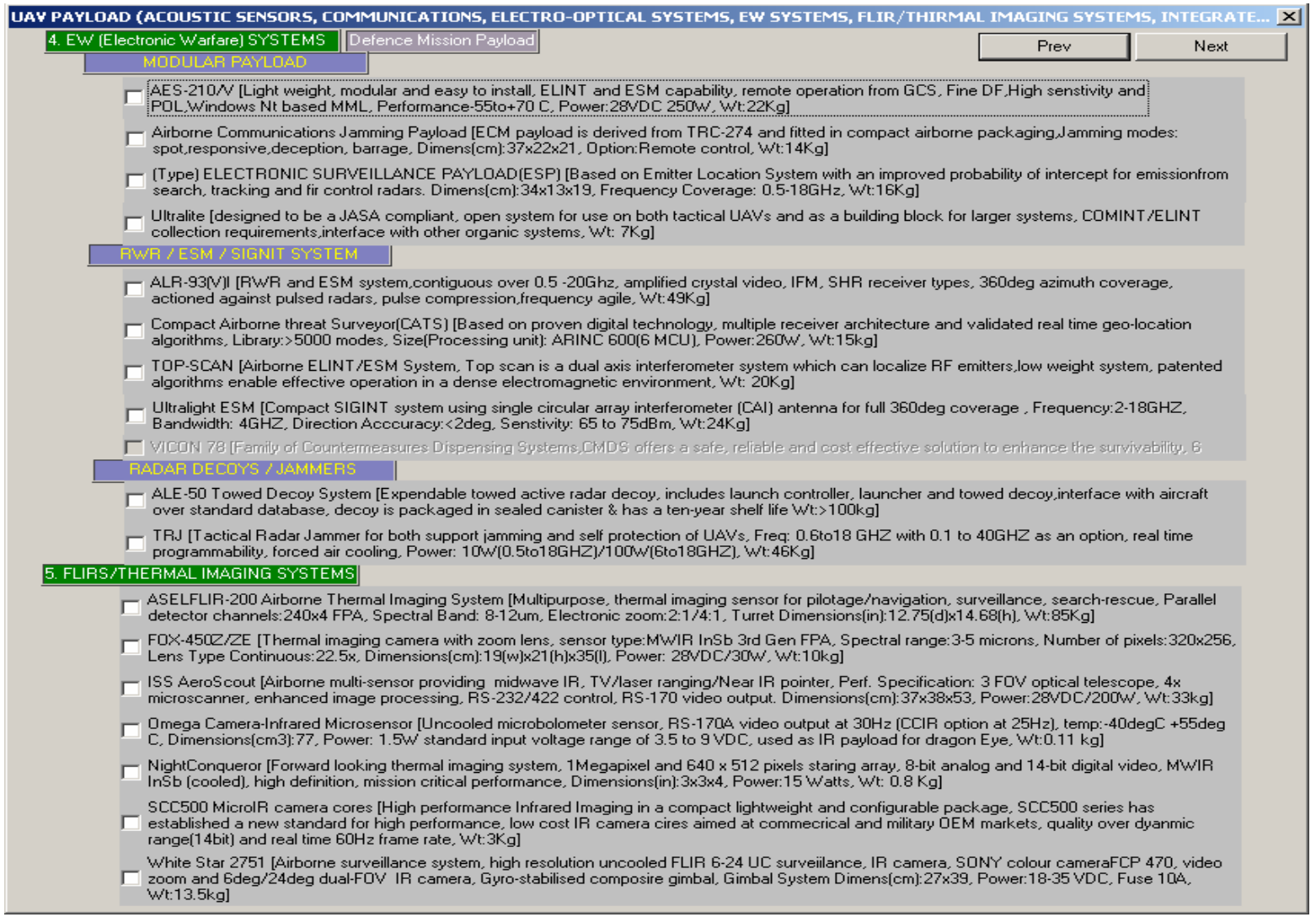


Figure 6.6: The Output Dialog of 'Payload' Class of Mission Systems - Electronic Warfare and Forward Looking Thermal Imaging Components

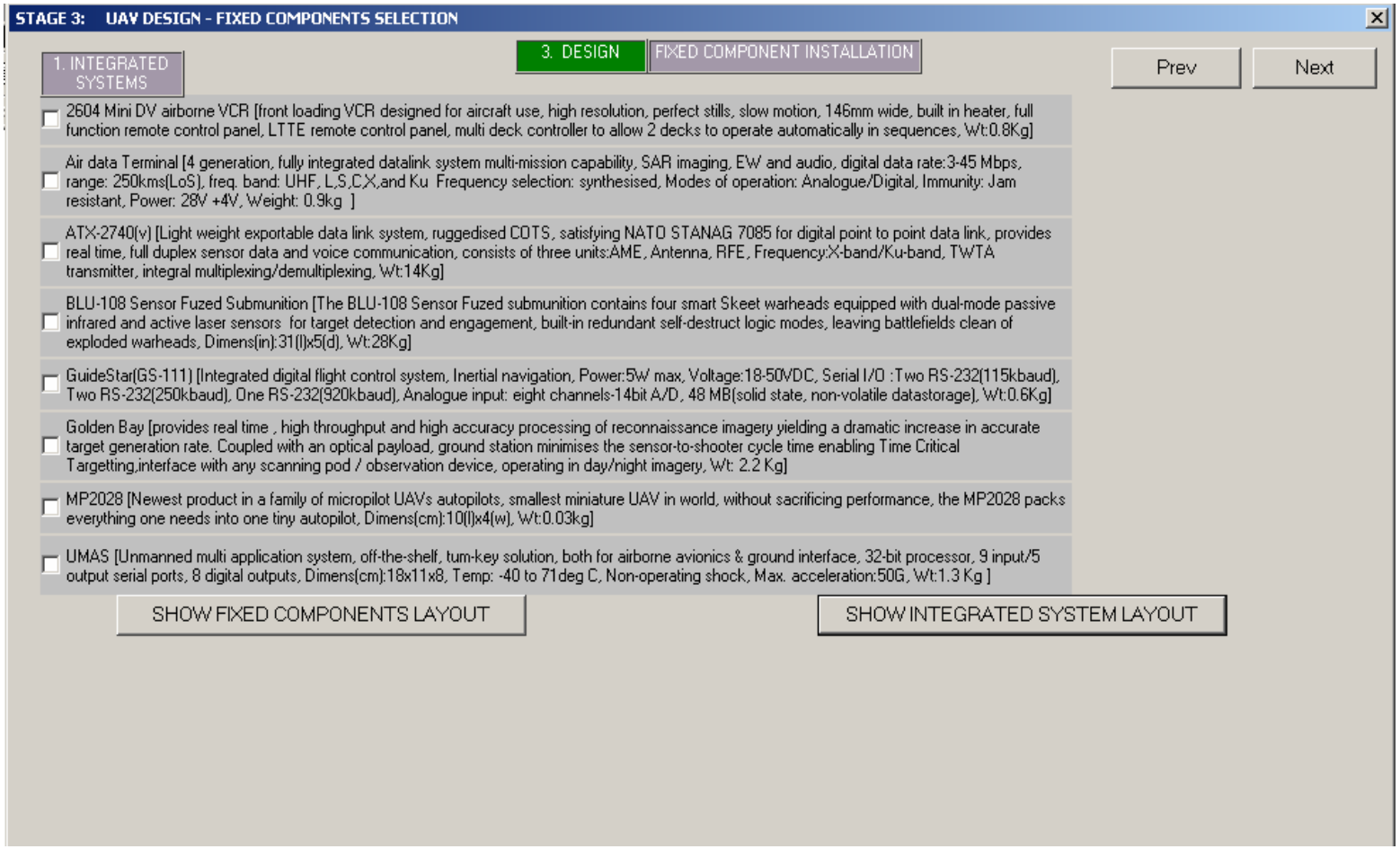


Figure 6.7: The Output Dialog of 'Payload' Class of Mission Systems - Integrated Component

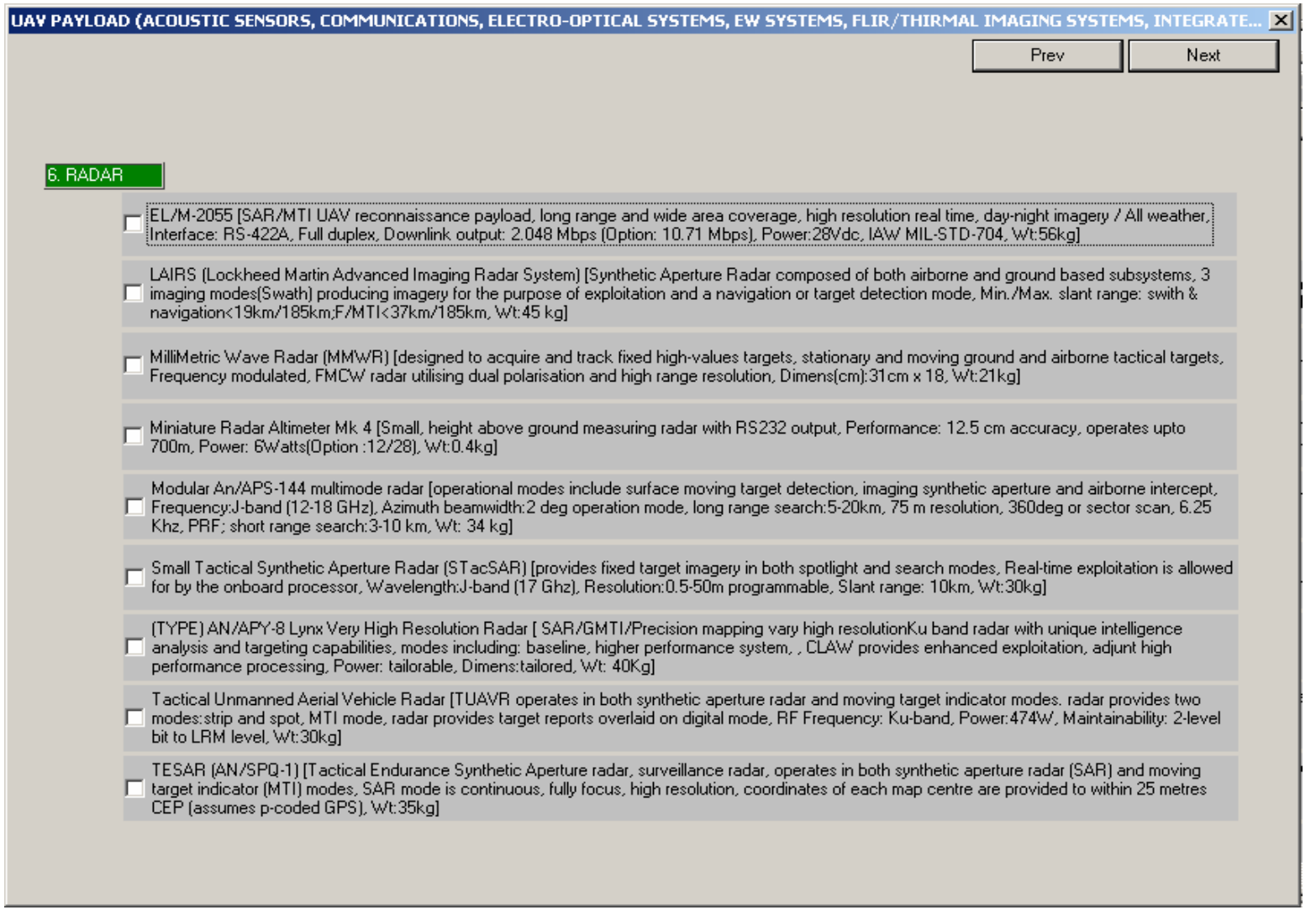


Figure 6.8: The Output Dialog of 'Payload' Class of Mission Systems - RADAR Component

6.2.4 'Database-I' Class

The 'Database-I' class involves identification of VTUAVs based on the specified design requirements. The class output is a dialog that provides the 'search and select' facility.

The dialog comprises of buttons for search/selection as follows:

- Search: It identifies the VTUAVs based on the specified design requirements (Weight, speed, endurance, ceiling, & altitude) and constraints (Physical dimensions, stub-wing requirement, and CASR part 101 safety regulation), presented in Fig 6.1;
- Select and Proceed: It selects a VTUAV from the search result, displayed in a data-grid and navigates to the output dialog of 'Fuel' class; and
- Back: It navigates to the last output dialog of 'Payload' class which is based on the RADAR component of mission system.

The dialog as an output of the 'Database-I' class is presented in Fig 6.9. The program code of the 'Database-I' class involving design requirements as class variables, and match, & select as class methods (Table 6.1) is presented at Appx H.4.

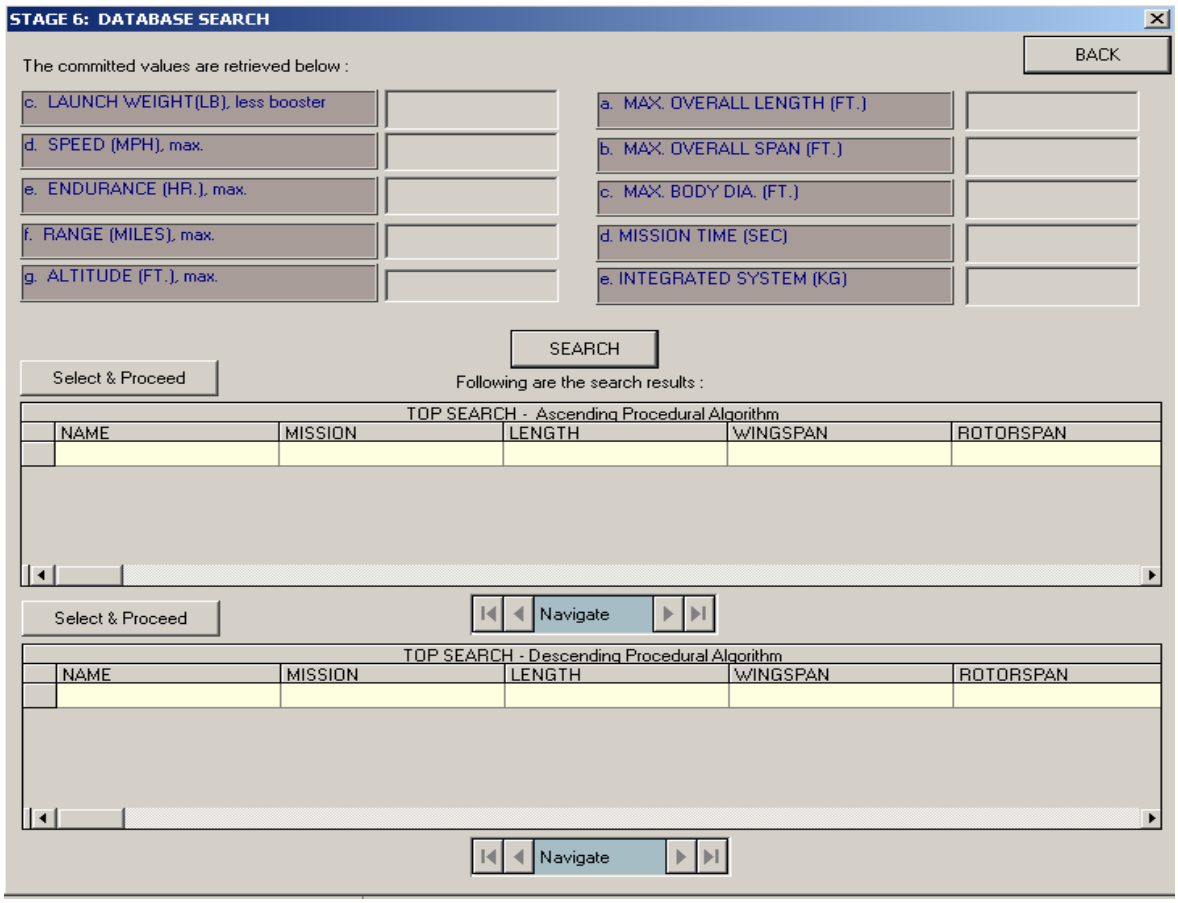


Figure 6.9: The Output Dialog of 'Database I' Class based on Design Requirements

6.2.5 ‘Fuel’, ‘Structural weight’, ‘Vertical drag’, & ‘Rotor system-I’ Classes

The ‘Fuel’, ‘Structural weight’, ‘Vertical drag’, and ‘Rotor system-I’ classes involve the evaluation of fuel, structural weight, vertical drag, and number of rotor blades based on the coaxial configuration. The output of the classes is combined in one dialog to provide maximum information.

The dialog comprises of buttons for navigation as follows:

- Next: It navigates to the output dialog of ‘Rotor system-II’ class; and
- Back: It navigates to the output dialog of ‘Database-I’ class.

The ‘Onload’ sub-routine that is executed automatically on the activation of the dialog is used for various estimations, evaluations and computations involved in the class as follows:

- It computes the weight of mission fuel;
- It evaluates the weight of various airframe sections of VTUAV;
- It computes the vertical drag; and
- It evaluates projected area of the fuselage in remote wake to compute the size of main rotor system and the number of rotor blades.

The dialog as an output of the ‘Fuel’, ‘Structural weight’, ‘Vertical drag’, and ‘Rotor system’ classes is presented in Fig 6.10, and the corresponding program code involved is presented at Appx H.5.

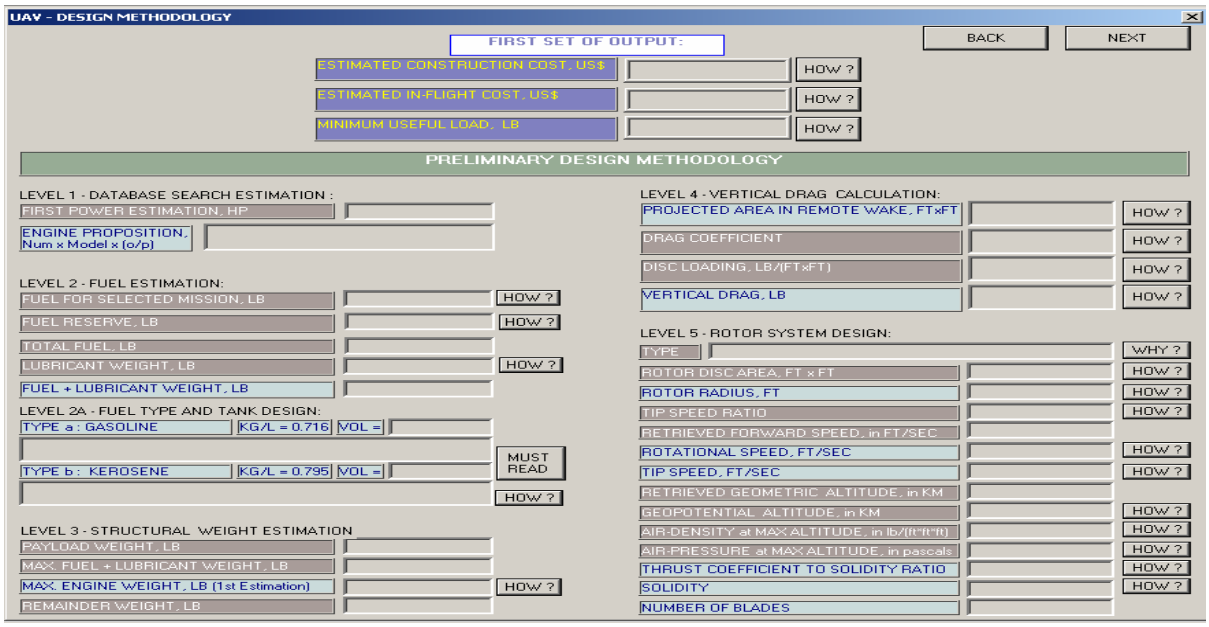


Figure 6.10: The Output Dialog of ‘Fuel’, ‘Structural weight’, ‘Vertical drag’, and ‘Rotor system-I’ Classes

6.2.6 ‘Rotor system-II’ Class

The ‘Rotor system-II’ class involves the selection of blade airfoil. The class output is a dialog that provides the select facility based on the type of airfoil.

The dialog comprises of “Slider” controls for varying the thickness of airfoil section and the involved camber including its maximum positioning. It also includes “Option boxes” for selecting the type of airfoil (symmetrical/cambered). Additionally, the dialog comprises of several buttons for information gathering / navigation as follows:

- Next: It navigates to the output dialog of ‘Drag’ class;
- More: It explains the detailed design of the airfoil; and
- Back: It navigates to the output dialog of ‘Fuel’, ‘Structural weight’, ‘Vertical drag’, and ‘Rotor system’ classes.

The dialog as an output of the ‘Rotor system-II’ class is presented in Fig 6.11. The corresponding program code of the ‘Rotor system-II’ class is presented at Appx H.6.

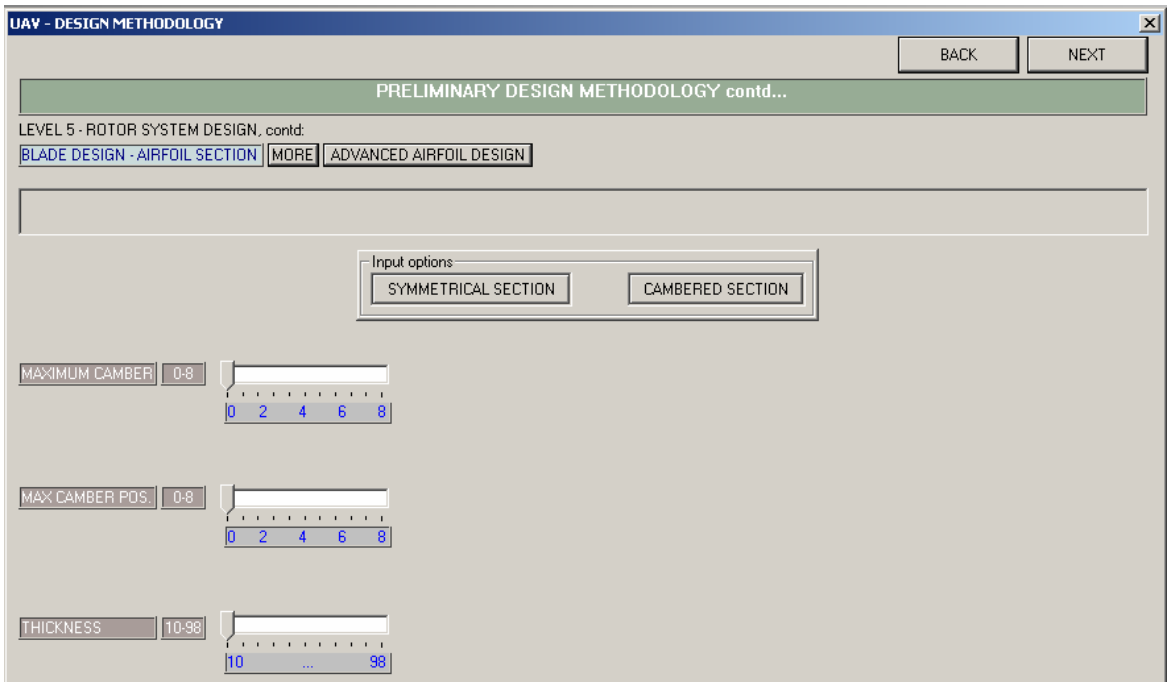


Figure 6.11: The Output Dialog of ‘Rotor system-II’ Class

6.2.7 ‘Drag’ Class

The ‘Drag’ class evaluates parasite and profile drag for further computation of power required to sustain forward flight at maximum speed. The output of ‘Drag’ class is a dialog that comprises of several buttons for information gathering / navigation as follows:

- How: It explains the detailed computation of coefficients of parasite and profile drag;
- Next: It navigates to the output dialog of 'Power' class; and
- Back: It navigates to the output dialog of 'Rotor system-II' class.

The 'Onload' sub-routine of the class evaluates the coefficients of parasite and profile drag. The dialog as an output of the 'Drag' class is presented in Fig 6.12, and the corresponding program code involved is presented at Appx H.7.

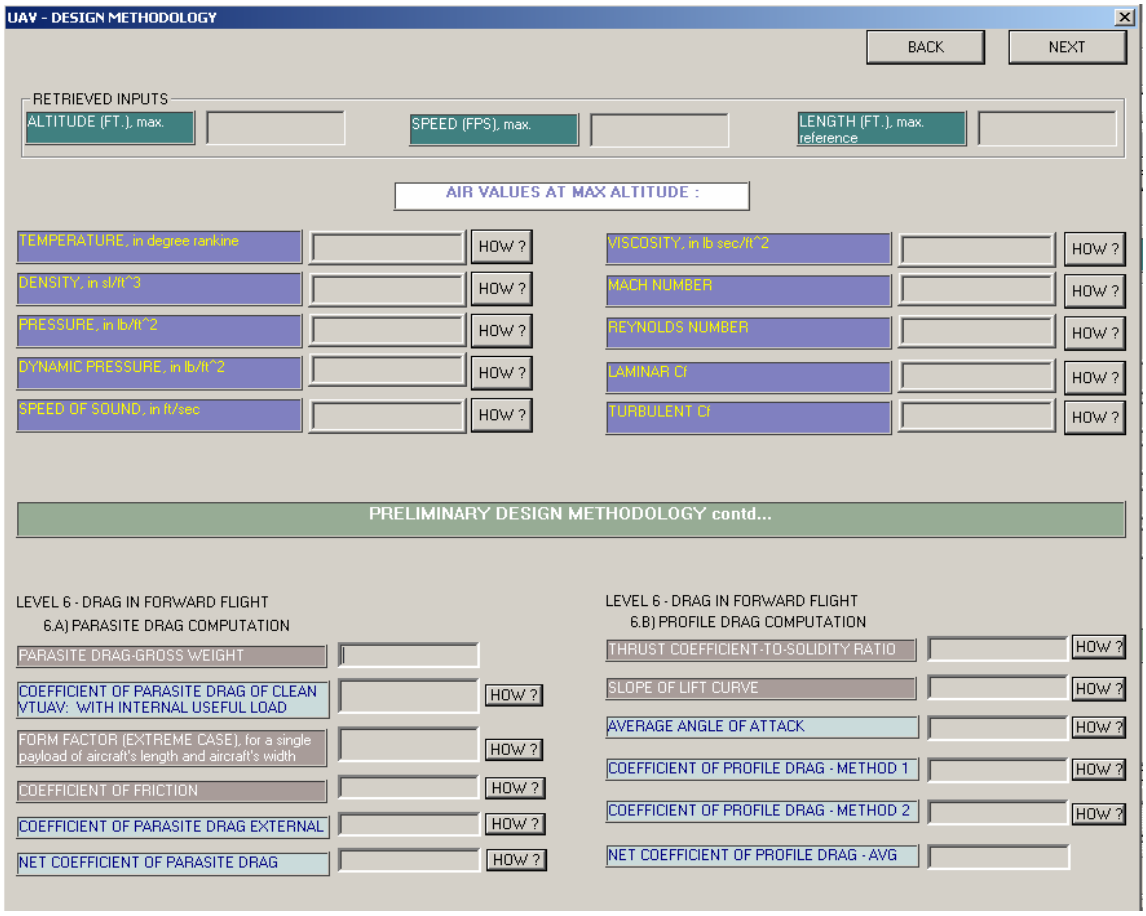


Figure 6.12: The Output Dialog of 'Drag' Class

6.2.8 'Power', 'Database-II', & 'Airframe-section weight' Classes

The 'Power', 'Database-II', and 'Airframe-section weight' classes involve the following: a) Evaluation of power required in forward flight for maximum speed; b) Reselection of engine; and c) Computation of structural weight of various airframe sections. The output of the classes is combined in one dialog to provide maximum information.

The dialog comprises of several buttons for engine-reselection / navigation as follows:

- Select engine name and click it: It selects the engine based on significant variance between evaluated and estimated power, and navigates to the output dialog of 'Balancing' class;
- How: It explains the detailed evaluation of the power required in forward flight for the maximum speed; and
- Back: It navigates to the output dialog of 'Drag' class.

The 'Onload' sub-routine of class evaluates the power required in forward flight for maximum speed, and computes the structural weight of airframe sections including rotor blades, propulsion, electrical, avionics, optional nacelles and dry systems of VTUAV.

The dialog as an output of the 'Power', 'Database-II', and 'Airframe-section weight' classes is presented in Fig 6.13, and the corresponding program code involved is presented at Appx H.8.

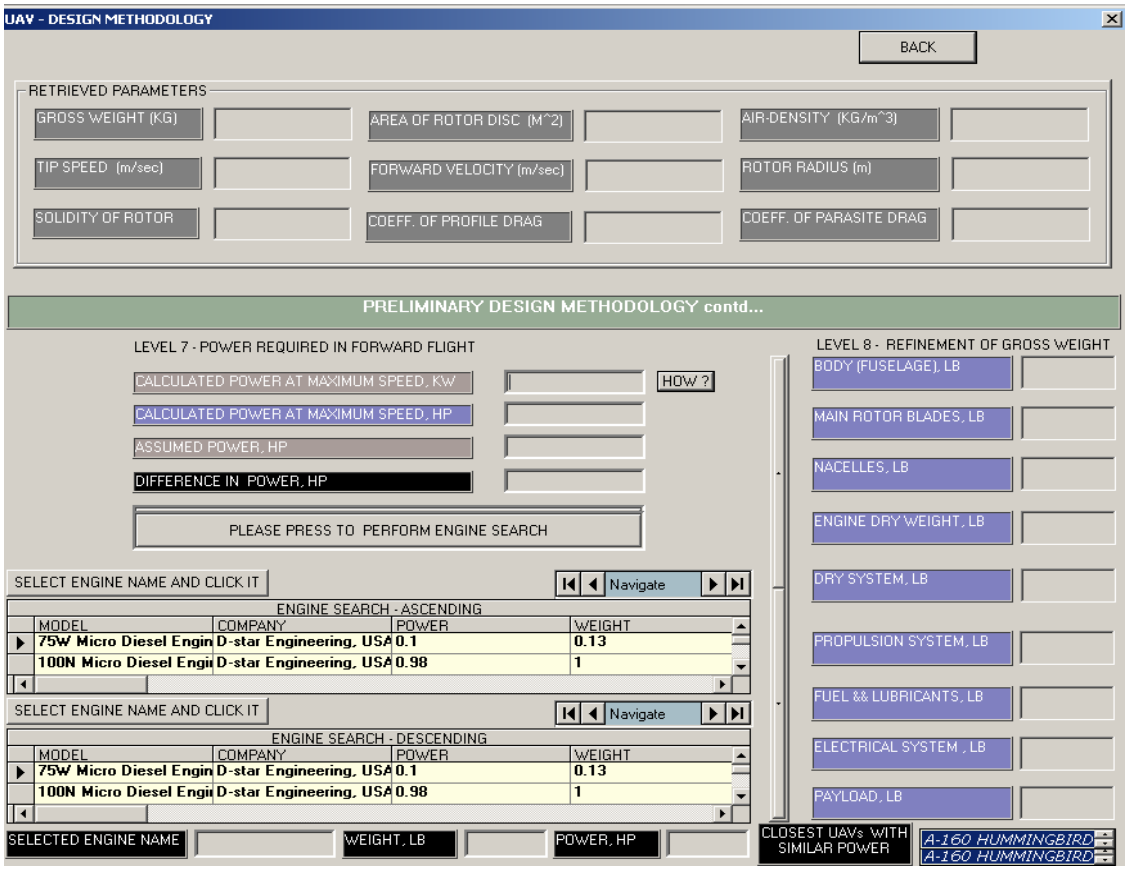


Figure 6.13: The Output Dialog of 'Power', 'Database-II', and 'Airframe-section weight' Classes

6.2.9 'Balancing' & 'Design-output' Classes

The 'Balancing' and 'Design-output' classes involve "Weight and balance" exercise to refine the inboard and external profile of the VTUAV to place the cg within the acceptable limits. The output of the classes is combined in one dialog to provide maximum information.

The dialog comprises of several buttons for engine-reselection / information gathering as follows:

- Feasibility: It determines the feasibility of positioning mission systems at various load stations by evaluating the corresponding longitudinal cg;
- Aim & Procedure: It explains the details of 'Weight and balance' exercise; and
- Back: It navigates to the output dialog of 'Power', 'Database-II', and 'Airframe-section weight' classes.

The 'Onload' sub-routine of class computes the longitudinal cg based on the load stations and the weights of mission systems. The dialog as an output of 'Balancing' and 'Design-output' classes is presented in Fig 6.14, and the corresponding program code involved is presented at Appx H.9.

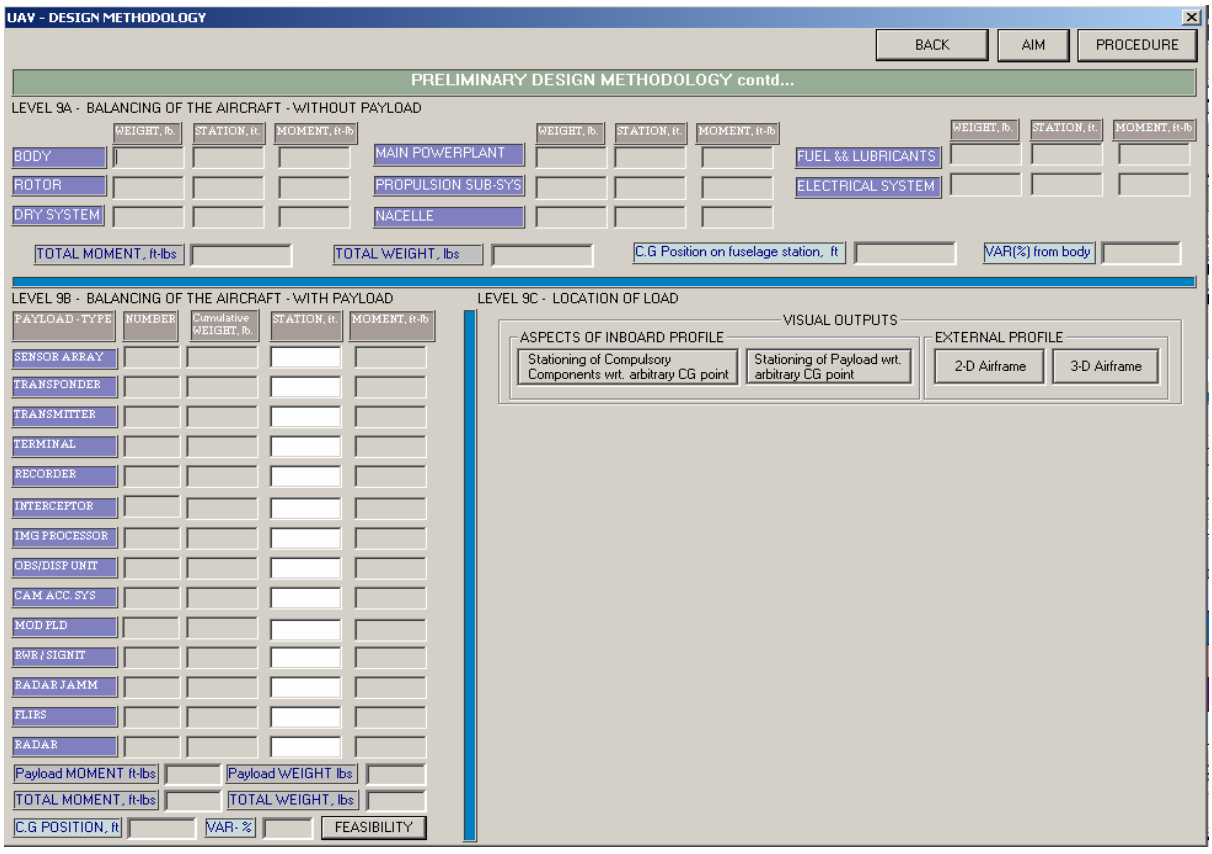


Figure 6.14: The Output Dialog of 'Balancing' and 'Design-output' Classes

6.2.10 'Non-editable database' & 'Editable database' Classes

The 'Non-editable database' and 'Editable database' classes provide read-only/update facilities of the design parameters of various VTUAVs in-development/in-service and their power-plants. The output of the classes is combined in single dialog to provide maximum information.

The dialog comprises of several buttons for 'search and sort' and print functions as follows:

- Print: It prints the design parameters of various VTUAVs and their power-plants; and
- Find: It searches VTUAV based on the design requirement (Weight, speed, endurance, ceiling, and altitude).

The 'Onload' sub-routine of class shows the design parameters of various VTUAVs in 'data-grid'. The dialog as an output of 'Non-editable database' and 'Editable database' classes is presented in Fig 6.15, and the corresponding program code involved is presented at Appx H.10.

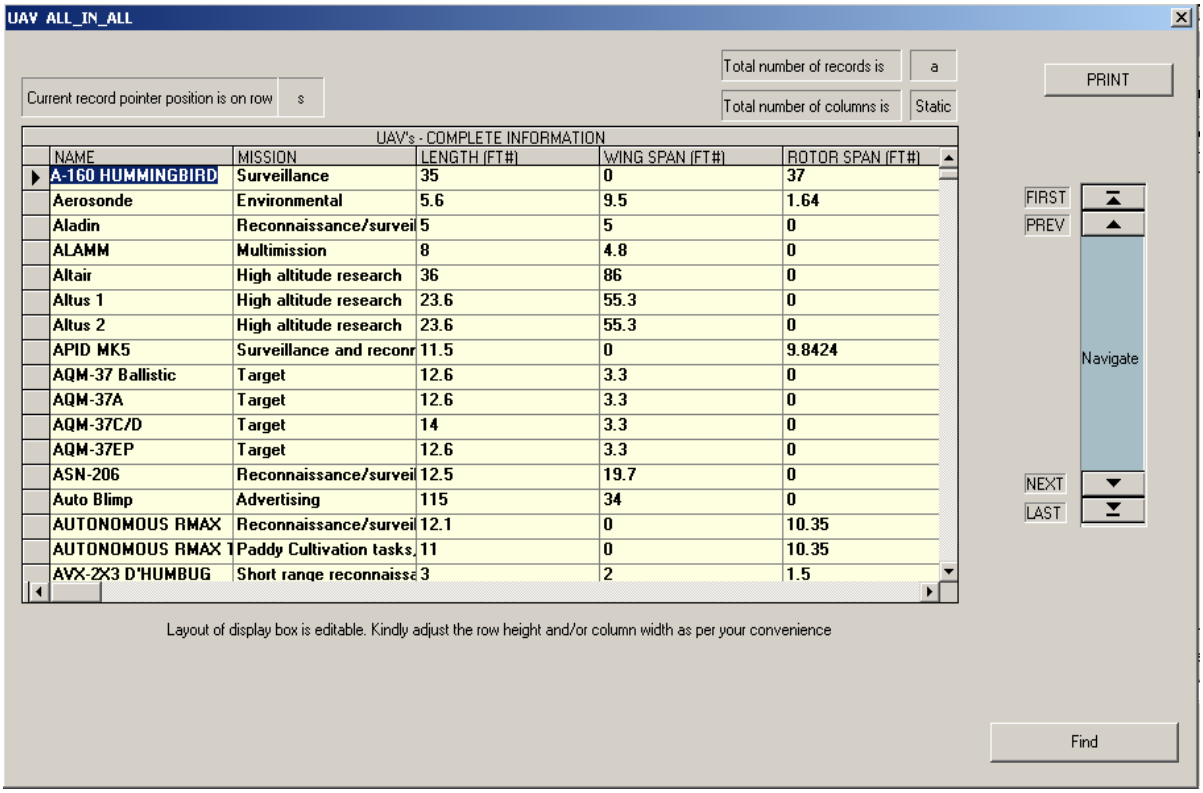


Figure 6.15: The Output Dialog of 'Non-editable database' & 'Editable database' Classes

6.2.11 ‘Calculator’, ‘Notepad’, ‘WordPad’ & ‘Send-by-mail’ Classes

The ‘Calculator’, ‘Notepad’, ‘WordPad’ and ‘Send-by-mail’ classes execute and record interim computations involved in the design of the VTUAV. These classes are inbuilt in the operating system.

6.2.12 Print Class

The print class involves printing of database and design configuration of VTUAV based on page-setup, page-copies, and page-orientation. The class is programmed as a sub-routine of the later discussed ‘View’ class. The dialog comprises of several buttons for standard print operations as follows: a) Options; b) Properties; c) Find Printer; d) OK; and e) Cancel.

The dialog as an output of print class is presented in Fig 6.16 and the corresponding program code involved is presented at Appx H.11.

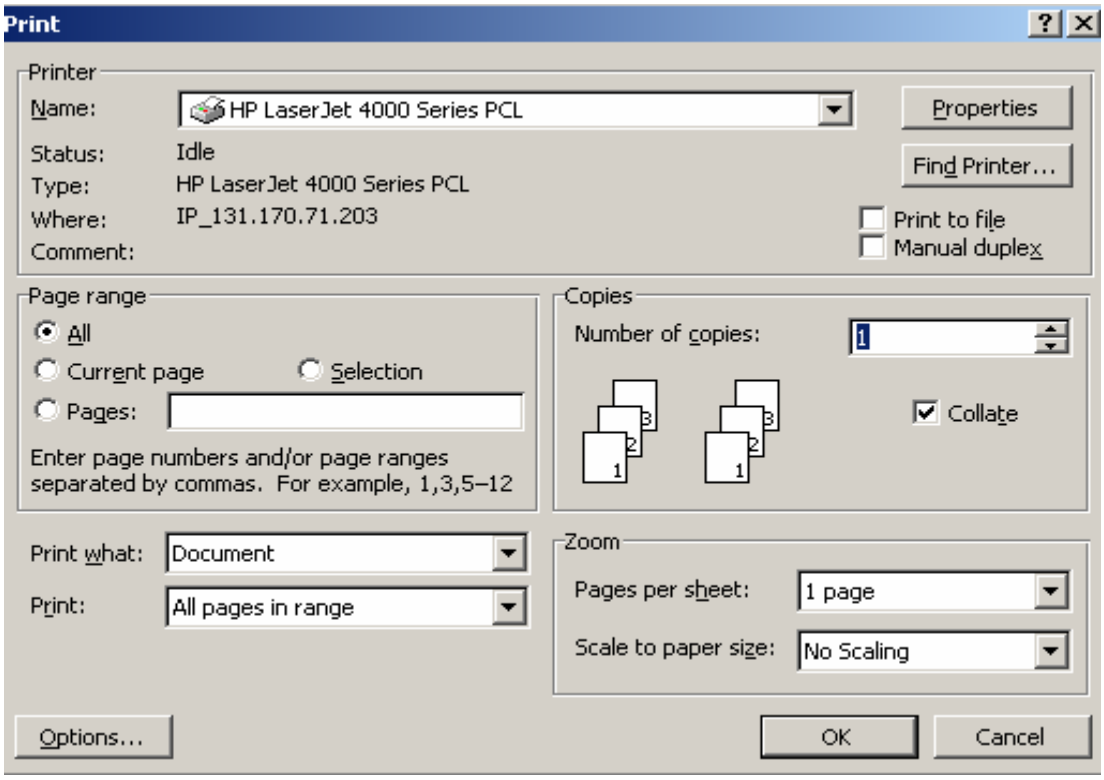


Figure 6.16: The Output Dialog of the Print Class

6.2.13 ‘Mission categories’ Class

The ‘Mission categories’ class provides the option to select the desired category of mission requirements based on the operational needs. The output of the class is a dialog that provides buttons for the selection of mission categories.

The dialog comprises of several buttons for selection/navigation and is illustrated as follows for four main mission categories (Generic, relief & monitoring and research, strategic, and tactical):

- ‘Select’ buttons: It involves the selection of mission categories; and
- Back: It navigates to the output dialog of ‘Design-input’ class.

The dialog as an output of ‘Mission categories’ class is presented in Fig 6.17, and the corresponding program code involved is presented at Appx H.12.

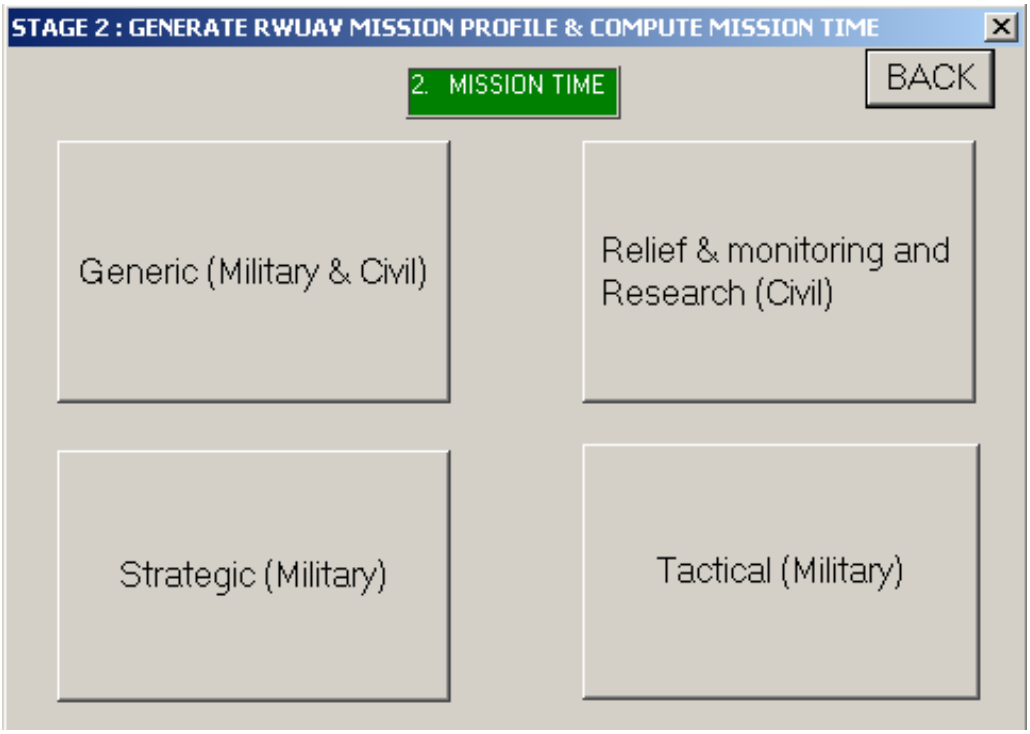


Figure 6.17: The Output Dialog of ‘Mission categories’ Class

6.2.14 ‘Mission requirements’ Class

The ‘Mission requirements’ class involves the mission requirements based on the selected mission category. The output of the class is a dialog that provides buttons for the selection of mission requirements

The dialog comprises of several buttons for selection/navigation and is illustrated as follows for ‘Generic’ mission category:

- ‘Select’ buttons: It involves the selection of mission requirements based on the mission category; and
- Back: It navigates to the output dialog of ‘Mission categories’ class.

The dialog as an output of ‘Mission requirements’ class is presented in Fig 6.18, and the corresponding program code is presented at Appx H.13.

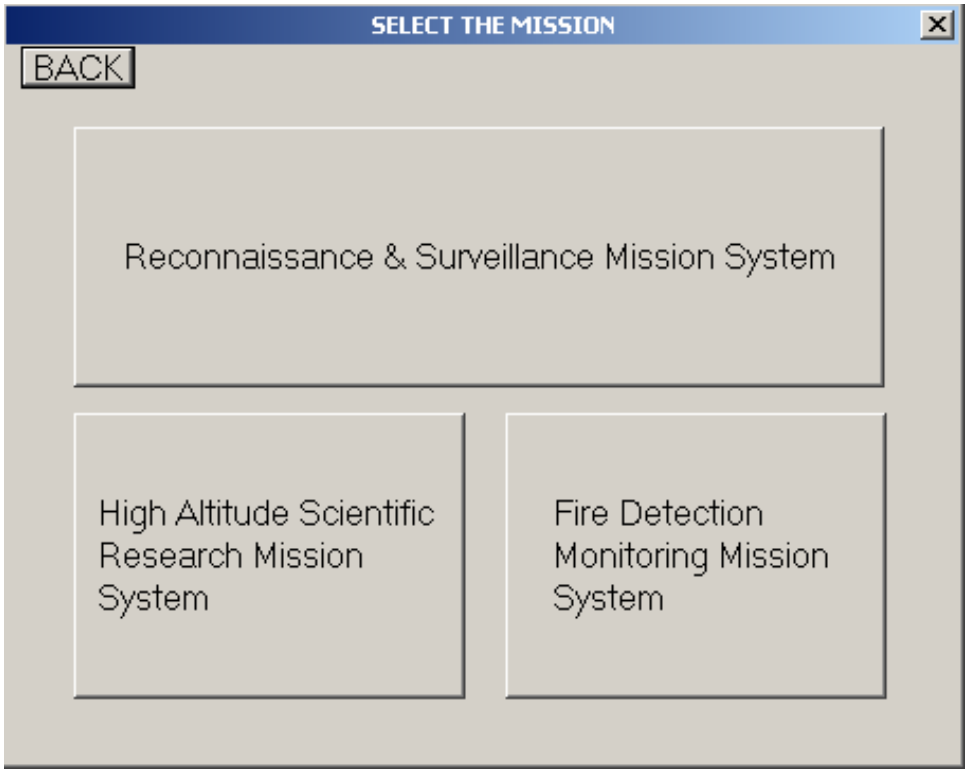


Figure 6.18: The Output Dialog of ‘Mission requirements’ Class based on Generic Mission Category

6.2.15 ‘Application’, ‘View’, & ‘Document’ Classes

The ‘Application’, ‘View’, and ‘Document’ classes are based on the internal framework of Microsoft Visual C++ and comprises of user-application interactions involving menu-bar, task-bar and input devices. These classes are not represented by a dialog as an output as it is involved in the development of software. The program code required for the design and implementation of the classes is presented at Appx H.14 to H.16.

7 RESULTS & DISCUSSIONS

The development of the conceptual design methodology for VTUAVs and its subsequent automation was based on the traditional helicopter design methodology. A systems approach was adopted to gain an insight in the required design methodology from a systems perspective. The autonomous component was integrated in the helicopter design methodology followed by the development of an expert system to automate the design. The steps adopted in the development of the design methodology and its automation, were as follows:

- Requirement analysis: To address the operational needs and operational environment for slating the mission requirements of VTUAVs.
- System design:
  - To identify the mission system technology from the identified mission requirements (that will provide the required mission capabilities to the VTUAV) by the development of system hierarchy; and
  - To investigate the vehicle system and autonomous system and its components and the traditional design stages for the development and refinement of the system design architecture.
- Design process:
  - To identify the design parameters from the identified attributes of components of the refined design architecture to serve as design inputs;
  - To develop the databases of VTUAVs in-development/in-service and their power-plants in accordance to the design parameters identified and to establish the design benchmarks; and
  - To identify the design requirements and constraints from the design parameters and update the traditional helicopter design methodology.
- Automation: To automate the updated helicopter design methodology for the development of an expert system.

7.1 Requirement Analysis

The requirement analysis involved the operational and mission analysis of the mission need statement. Based on the results of the mission analysis the mission profile was developed to identify the mission phases.

The operational analysis resulted in the identification of general/special operational needs of civil/military clientele and the operational environment of terrain, time, threat, weather, situation and littoral. The mission analysis of the operational needs and environment resulted in the identification of five mission categories (Generic, relief & monitoring, research, strategic, and tactical) under which the 23 mission requirements of VTUAV were covered as follows: a) Generic – Surveillance and detection; b) Relief and monitoring – Agriculture, relief, forest, and power; c) Research – Science and geology; d) Strategic – Supply, command, control, attack, protection, and relay; e) Tactical – Electronic detection, nuclear, biological, chemical, & communication emissions, intelligence, and force protection. The analysis of the mission requirements from an operational perspective resulted in the identification of four generic mission phases as follows: a) Take-off and landing; b) Ingress and egress; c) Engage and disengage; and d) Transit. The mission phases provided the base for the development of mission profile to compute mission time.

The mission requirements were identified to assist in the identification of the mission system technology, the functional characteristics needed to provide the required mission capabilities to the VTUAV. The mission time evaluated from the mission profile to assist in evaluating the full requirements of the VTUAV to accomplish the mission.

7.2 System Design

The system design involved the functional analysis and synthesis of VTUAV system. The functional analysis covered three subsystems (Mission, vehicle and autonomous) to identify the elements and its functional characteristics that will meet the mission requirements.

The functional analysis of the mission system technology resulted in the identification of four generic components and corresponding elements based on the functional characteristics as follows: a) Communication – Sensor, transponder, transmitter, terminal, & interceptor; b) Combat – Receiver, detector, decoy, ranger & jammer; c) Imaging – Processor, display, camera, & accessory-module; and d) Integrated – Onboard, & ground module. A hierarchy of the mission system technology was developed for a comprehensive understanding of the mission systems and its components and elements. The functional characteristics of the components were analysed to match the mission requirements and classified as either base component or additional component. The mission capabilities provided by the components to

meet the mission requirements were stipulated as the attributes of the mission system – Observation, detection, avoidance, communication, navigation, and survival.

The functional analysis of the vehicle (helicopter) involved an analysis of the design process to identify the components of the system – Airframe, power-plant, and rotor. The computation requirements in the design process of the components were designated as the attributes of components. The attributes were further refined considering all the design stages of helicopter and were identified as – Estimation, computation, iteration, optimisation, and trade-offs.

Similarly the functional analysis of the autonomous system comprising of the components (Avionics, launch & recovery, data-link, and terminal) resulted in the attributes of the command and control of flight & mission, and autopilot for mission success.

The synthesis of the three subsystems identified resulted in the development of the design architecture of VTUAV in an input-process-output system configuration for the design of a VTUAV with slated mission capabilities based on the operational needs and operational environment. The VTUAV design architecture comprised of three components with its respective attributes. Further, analysis of the operational relationships between autonomous, and mission & vehicle components from the perspective of mission commonality resulted in a refined VTUAV design architecture with only two components and re-allocation of attributes of the complete system to the mission and vehicle system.

The refined VTUAV design architecture developed will provide the base structure for the development of a design process for the VTUAV and its subsequent automation. The attributes of subsystems identified will assist in the identification of input requirements to the design process.

7.3 Design Process

The design process involved design analysis, collation and a design development to update the traditional rotary-wing design methodology from an autonomous perspective. The design analysis comprised of the identification of design parameters as inputs to the design process. The collation covered the establishment of design benchmarks from a database of VTUAVs. The design development identified the requirements and constraints and a re-consideration of the traditional design process.

The design analysis resulted in the identification of design parameters from the analysis of the attributes of vehicle and mission components of the refined design architecture of VTUAVs. The design parameters identified were designated as input requirements to the design process and included - weight, power, speed, range, ceiling, endurance, payload, rotor size, stub-wing requirement, and safety regulations.

The design collation resulted in the development of databases of various VTUAVs in-development/in-service and their power-plants for the establishment of design benchmarks of the design parameters identified. The benchmarks established; validated the design parameters prior to being used as input requirements to the design process. The benchmarks covered the micro, small, and large UAVs and provided a statistical relationship of the following: a) Payload & maximum take-off weight; and b) Speed, endurance, payload & maximum take-off weight. The statistical derivation was based on regression analysis to develop polynomial plots for the establishment of mathematical equations.

The design development resulted in the identification of the design requirements and constraints from the identified design parameters. Additional constraints were identified from the analysis of past design data. The design development subsequently updated the traditional rotary-wing design methodology for the development of conceptual design configuration of the VTUAV based on the design requirements and constraints identified and the database developed.

The updated rotary-wing design methodology developed was to assist in the development of an automated expert system for the generation of a conceptual design configuration of a VTUAV from a set of inputs comprising of slated mission requirements.

7.4 Automation

The automation involved the selection of suitable programming language and the database in a programming environment for the design of an expert system to provide output dialogs based on the updated rotary-wing design methodology. Microsoft Visual C++ was selected as the programming language; database was on Microsoft SQL Server and the dynamic environment was provided by Microsoft Visual Studio .Net. The design of the expert system comprised of the identification of various reusable classes, variables and methods from the

analysis of updated design methodology. The dialogs of the expert system covered the programming of the classes identified to achieve the desired functionality.

The design of expert system resulted in the identification of five class categories based on design, database, service, transition and support; under which 27 reusable classes were covered to provide the computational details of the design stages of updated rotary-wing methodology as follows: a) Base category – Design-input, mission-time, payload, database-I, fuel, structural weight, vertical drag, rotor system-I, rotor system-II, drag, power, database-II, airframe-section weight, balancing, & design-output; b) Database category – Non-editable, & editable database; c) Utility category – Calculator, notepad, word-pad, send-by-mail, & print; d) Navigation category – Mission categories, & mission requirements; and e) Framework category – Application, view, & document. The corresponding class variables and class methods for each of these were also identified and summarised for ready reference.

The dialogs of the expert system are the programmed outputs of the classes identified. The dialogs comprised of various controls including buttons, sliders, option-boxes, check-boxes, text-boxes, and data-grids to implement the class methods based on the class variables identified. These buttons provided for design decisions and validations based on established statistical benchmarks.

The automated expert system was developed to assist in the generation of conceptual design configuration of VTUAVs based on the user slated inputs comprising of mission requirements derived from operational needs and operational environment.

7.5 Result Summary

The results of the requirement analysis, system design, design process and automation for the conceptual design of VTUAV are summarised at Table 7.1 and the block diagram representing the various stages of analysis is presented in Fig 7.1.

Table 7.1: Result Summary – Requirement Analysis, System Design, Design Process, and Automation

Design Methodology		Design Outcomes	
Stages	**Design Analysis**	**Analysis Results**	**Design Results**
Stage 1 Requirement analysis	Mission need statement	• Customised need statement	Mission requirements
	Operational analysis	• Operational needs • Operational environment	
	Mission analysis	• Mission requirements	
	Mission phases	• Mission profile	
Stage 2 System design	Functional analysis	• Mission system and its attributes, elements, & components • Vehicle system and its attributes, elements, & components • Autonomous system and its attributes, elements, & components	Refined VTUAV design architecture
	Synthesis	• VTUAV design architecture • Refined VTUAV design architecture	
Stage 3 Design process	Design analysis	• Design parameters	VTUAV design methodology
	Design collation	• Database of VTUAV & their power-plants • Relative benchmarks to validate design parameters	
	Design development	• Design requirements & constraints • VTUAV design methodology	
Stage 4 Automation	Expert system – Design	• Base category and classes • Database category and classes • Utility category and classes • Navigation category and classes • Framework category and classes	VTUAV expert system
	Expert system – Dialog	• Dialogs as output of classes identified	

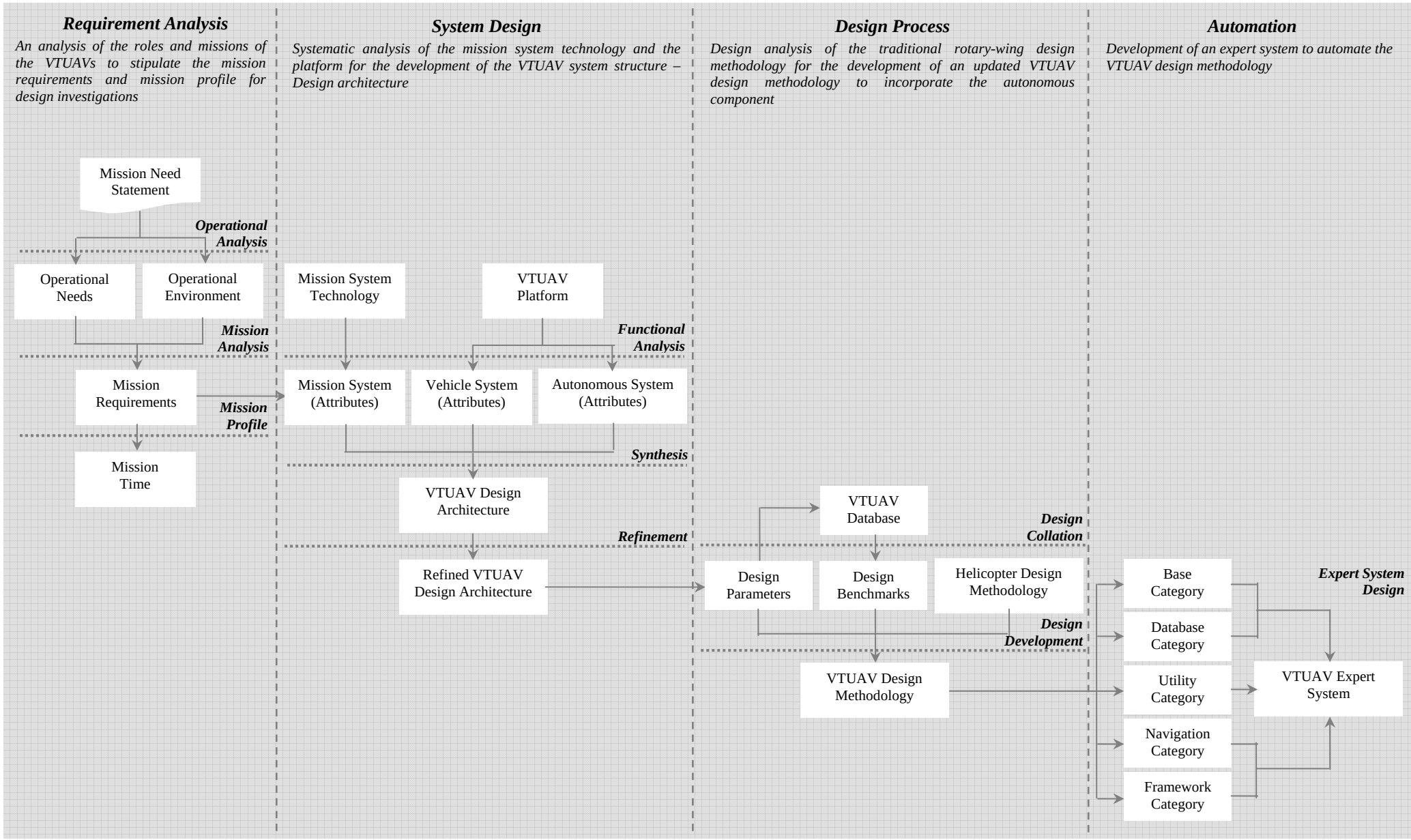


Figure 7.1: Block Diagram – Requirement Analysis, System Design, Design Process and Automation

8 RECOMMENDATIONS

The development of an expert system to automate the VTUAV design was based on the traditional helicopter design methodology and involved various design stages of requirement analysis, system design, design process and automation. The stages involved the analysis of the mission need statement to identify mission requirements for the development of VTUAV design architecture followed by VTUAV design methodology. The investigations were comprehensive and addressed various key issues of design and development including an illustration to demonstrate the model. The following issues are recommended for further research and continuing development:

- Operational doctrines: The operational doctrines of UAVs are still under development and over the time it will be firmly established. The automated operational analysis will need to be continually updated based on the emerging doctrines;
- Mission systems: The growth of the mission systems technology in recent years indicates further development to meet the emerging operational requirements. The technical parameters and its capabilities will need to be incorporated in the model to provide for the mission capabilities that can be derived from the VTUAV being designed;
- VTUAV platform: The advancement in mission system technology will provide for the suitable payload for the aerial platform. The payload design will play a significant input in the design of the VTUAV platform with the capability. Payload design process will need to be accordingly updated to incorporate the parameters of advanced mission system technology;
- Autonomous systems: The developments in the data-link, and launch & recovery systems will need to be incorporated in the database as design parameters for the design of the VTUAV. These developments will provide advanced degree of autonomy in future designs;
- Rotary wing configuration: The present autonomous model addresses only the conventional configuration - main and tail rotor. Other configurations of tandem, coaxial, NOTAR, and tilt should be included in the model to provide alternate design configurations;
- Rotor system: The tip-shape, taper, and twist parameters of the rotor design have not been adequately addressed in the design process. These design issues needs to be addressed and incorporated as a sub-routine in the program;

- CATIA modelling: The inboard and external profile developed based on design parameters evaluated will need to be transferred to CATIA for digitised simulation of the design configuration; and
- Expert system: The selected programming environment, language and the database should be upgraded to latest versions in the model for future compatibility.

The block diagram representing the key issues and recommendations for further research and development of the VTUAV expert system is presented in Fig 8.1.

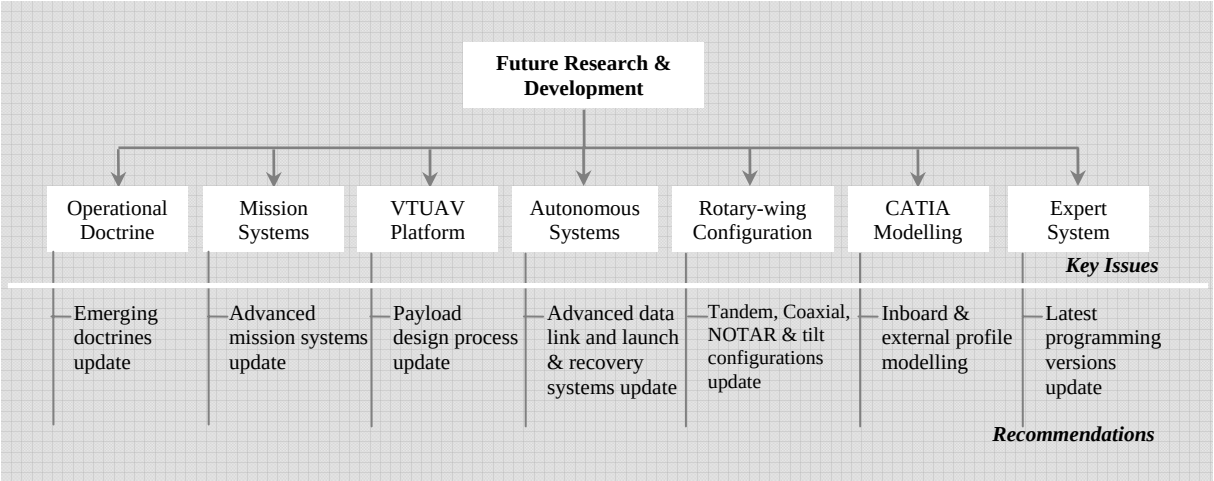


Figure 8.1: Block Diagram – Recommendations for Further Development of VTUAV Expert System

9 CONCLUDING REMARKS

A system approach was adopted for the development of an expert system from the updated rotary-wing design methodology to automate the design of a VTUAV. The process involved the development of system architecture for the identification of design parameters that serve as key inputs to the updated design methodology. The various stages of investigations to automate the design of a VTUAV were as follows:

- System approach: The system approach holistically investigated the operational needs & operational environment to identify the mission requirements of VTUAVs. Off-the shelf mission systems were analysed to identify its functional characteristics (flight and automation) that contribute to the requirements of the mission. Any future developments in mission system technology will need to be considered to supplement the functional characteristics;
- Design architecture: The system architecture was developed in an input-process-output configuration to identify the components and attributes of the system. The system comprised of mission, vehicle & autonomous components, with its functional characteristics as its attributes. The architecture was further refined to optimise the number of components and its attributes. The relationships were identified as component-component, component-attribute, and attribute-attribute to reduce the number of components. Future developments in mission and autonomous systems will need to be analysed and adapted as additional component;
- Design methodology: The traditional rotary wing design methodology involved in the conceptual design of helicopters was updated to incorporate an autonomous component for the development of conceptual design configuration of the VTUAV. The updated methodology incorporated key design inputs that were identified from the analysis of the attributes of system components. It also included design benchmarks for validation of design inputs; statistically derived from the technical parameters of in-development/in-service VTUAVs. The benchmarks will need to be updated periodically in-line with the advanced design developments; and
- Automation: The automation of the updated design methodology to generate design configurations of VTUAVs based on slated user inputs, resulted in the development of an expert system. The expert system was developed in Microsoft Visual Studio .Net 2003 & Microsoft SQL Server 2000 and comprised of various reusable classes, variables and methods that were identified from the comprehensive investigations of the updated design methodology. The expert system will need to be updated to new versions when introduced

and the classes, variables and methods will also require periodic updates based on design and technology developments.

The automated conceptual design process for VTUAVs developed in this thesis pioneers an aerospace design tool for further development. A detailed analysis of the design tool needs to be attempted for further refinement and enhancement of the design capability of the automated design process. The design tool when refined will serve as a pre-design analysis decision support system, for further design development of the VTUAV defined.

10 REFERENCES

1. Sommerville, I. 2000, *Software Engineering*, 6th edn, Addison-Wesley, Essex, U.K.
2. Sommerville, I. 1996, *Software Engineering*, Addison-Wesley, Essex, U.K.
3. Sinha, A.K. 2002, *AV365-Integrated Aircraft Design*, Systems Engineering, School of Aerospace, Mechanical and Manufacturing Engineering, RMIT, Australia.
4. Sinha, A.K., Bil, C., & Scott, M.L. 2000, 'Design of Optimum Payloads for Mid-life Upgrade of Maritime Helicopters: Stages I, II, III and IV', *Proceedings of the 3rd Australian Pacific Vertiflite Conference on Helicopter Technology*, Canberra, Australia.
5. Sinha, A.K., Bil, C., Scott, M.L., & Mohandas, P. 2000, 'Design of Optimum Payloads to Meet the Mission Requirements of Unmanned Aerial Vehicles', *Proceedings of the 22nd Congress of the International Council of Aeronautical Sciences*, Harrogate, U.K.
6. Sinha, A.K., Bil, C., & Scott, M.L. 2000, 'Design of Optimum Payloads for Mid-life Upgrade of Maritime Helicopters: Stages V and VI', *Proceedings of the International Conference on Systems Thinking in Management*, Melbourne, Australia.
7. Pidwirny, M. 2005, *Introduction to Systems Theory*, Definitions of System and Models, viewed 26 April 2005, <http://www.physicalgeography.net/fundamentals/4b.html>.
8. Aronson, D. 2005, *Overview of System Thinking*, viewed 26 April 2005, <http://www.thinking.net/Systems_Thinking/OverviewSTarticle.pdf>.
9. Gharajedaghi, J., Ackoff, R. 1984, *Mechanisms, Organisms and Social Systems*, Strategic Management Journal, viewed 26 April 2005, <http://classes.bus.oregonstate.edu/ba465/Articles/G&A's%20article.htm>.
10. Checkland, P. 1981, *Systems Thinking Systems Practice*, John Wiley & Sons, Chichester, U.K.
11. Otondo, R.F., Nichols, E.L., Feder, L., Hankins, P. D. 2005, *From Cyclical to Systems Thinking: Cycle Time Reduction in Complex Systems*, viewed 28 April 2005, <http://www.people.memphis.edu/ ~cscm/iscm1/cyclicaltosystems.pdf>.
12. Rosnay, J., De. 1997, *Analytic vs. Systemic Approaches*, viewed 28 April 2005, <http://pespmc1.vub.ac.be/ANALSYST.html>.
13. Heylighen, F. 1995, *Downward Causation*, viewed 02 May 2005, <http://pespmc1.vub.ac.be/DOWNCAUS.html>.
14. Oslovar, M. 1999, *Starting a Software Quality Organization? Take a Systems Approach*, Strategic Management Journal, viewed 29 April 2005, <http://www.ssq.org/chapter/sd/past/sdapr99.htm>.

15. Woodward, B. 2005, *Thinking More Systemically about Systems Thinking*, viewed 03 May 2005, <http://www.capitalizingonchange.org/docs/Systems%20Thinking%20Web%20Copy.pdf >.
16. Bahill, A.T. & Gissing, B. 1998, 'Re-evaluating systems engineering concepts using systems thinking', *Proceedings of the IEEE Transactions on Systems, Man and Cybernetics*, vol. 28, no. 4, pp. 516-527.
17. Flood, R.L. & Jackson, M.C. 1991, *Creative problem solving*, John Wiley & Sons, Chichester, U.K.
18. Cook, S.C. & Allison, J.S. 1998, 'A Systems Thinking Approach To Selecting Systems Methodologies For Defence High-Level Systems', *Proceedings of Systems Engineering*, Systems Engineering Society of Australia, Canberra, Australia, pp. 1-4.
19. DRM Associates 2005, *Systems Engineering Process*, ver.1, viewed 21 April 2005, <http://www.npd-solutions.com/se.html>.
20. Bahill, A.T. & Briggs, C. 2001, 'The systems engineering started in the middle process: a consensus of system engineers and project managers', *The Proceedings of Systems Engineering*, pp. 156-167.
21. Jacobson, I., Booch, G., & Rumbaugh J. 1999, *The Unified Software Development Process*, Addison-Wesley, Essex, U.K.
22. Serra, M. 2005, *Digital IC Testing: An Introduction*, University of Canada, viewed 21 April 2005, <http://www.csr.uvic.ca/~mserra/testing/draft.html>.
23. Wells, D. 2005, *Unit Test*, A kind of Automated Tests, viewed 12 April 2005, <http://c2.com/cgi/wiki?UnitTest>.
24. Jeffries, R. 2005, *Functional Test*, Acceptance Test, viewed 12 April 2005, <http://c2.com/cgi/wiki?FunctionalTest>.
25. Shishko, R. & Chamberlain, R.G. 1995, *NASA Systems Engineering Handbook*, viewed 18 Dec 2005, <http://ldcm.nasa.gov/library/Systems_Engineering_Handbook.pdf >.
26. Moore, J.W. 2005, *IEEE to Adopt ISO/IEC Systems Engineering Standard*, viewed 20 May 2005, <http://standards.ieee.org/announcements/pr_15288.html>.
27. Jackson, S., Mathers, G. & Simpson, M. J. 2003, *Towards the Development of a Domain-Specific Framework for Systems*, viewed 20 May 2005, <http://www.adse.nl/pdf/new/domainspecific_framework_for_se_of_commercial_aircraft.pdf>.

28. Institute of Electrical and Electronic Engineers 1998, *IEEE 1220*, IEEE Standard for Application and Management of the Systems Engineering Process, viewed 20 May 2005, <http://www.software.org/quagmire/descriptions/ieee1220.asp>.
29. Electronic Industries Association 1999, *EIA I/S 731: SECM Information*, viewed 20 May 2005, <http://www.secat.com/cmm/731inf.shtml>.
30. Sinha, A. K. 1998, *A System's Approach to the Conceptual Design of Helicopter Multi-role Capabilities*, PhD thesis, Royal Melbourne Institute of Technology, Melbourne, Australia.
31. Sinha, A. K. 2002, *Introduction to Preliminary Helicopter Design*, Department of Aerospace Engineering, Royal Melbourne Institute of Technology, Melbourne, Australia, pp. 12-59.
32. Sinha, A.K., Kusumo, R. Scott, M.L., Bil, C. & Mahondas, P. 2001, 'A System Approach to UAV Systems Issues and Challenges for Technology Development, Test and Validation,' *The Proceedings of 16th International UAV Systems Conference*, Bristol, U.K.
33. Sinha, A.K. 2002, *Avionics and Aerospace Systems-Mission Systems*, Lecture notes, Royal Melbourne Institute of Technology University, Melbourne, Australia.
34. Sinha, A.K., Kusumo, R. Scott, M.L. & Yuen, C.Y.Y. 2002, 'A System Approach to the design of multi-mission Airframe for Unmanned Air Vehicles systems,' *The Proceedings of 17th International UAV Systems Conference*, Bristol, U.K.
35. Unmanned Aerial Vehicles 2004, *US Commits To Further UAV Spending*, viewed 26 April 2005, <http://www.livingroom.org.au/uavblog/archives/us_commits_to_further_uav_spending.php>.
36. Unmanned Aerial Vehicles 2004, *Australia Budgets For UAVs*, viewed 26 April 2005, <http:/www.livingroom.org.au/uavblog/archives/australia_budgets_for_uavs.php>
37. Sweetman, B. 1997, *Unmanned Aerial Vehicle*, viewed 23 May 2005, <http://encarta.msn.com/text_701610394__1/Unmanned_Aerial_Vehicle.html>.
38. Department of Secretary of US Defence 2001, *Unmanned Aerial Vehicles Roadmap: 2000-2025*, Washington D.C., USA.
39. Seagle, D. 2001, 'Naval UAV Programs PMA-263', *The Proceedings of 16th International UAV Systems Conference*, Bristol, U.K.
40. Stewart, C. & Rutherford, A. 1999, 'Vigilante VTOL UAV', *The Proceedings of 14th International UAV Systems Conference*, Bristol, U.K.
41. Department of US Navy 1999, *Vertical Take-Off and Landing Tactical Unmanned Aerial Vehicle (VTUAV) Concept of Employment (CoE)*, USA.

42. Haskins, S. 2004, *The Latest Challenges, Opportunities and requirements in Unmanned Aerial Vehicles Platforms, Payloads & Missions*, Training course, London, U.K.
43. McKee, M. W. 2000, *VTOL UAVs Come of Age: US Navy Begins Development of VTUAV*, viewed 29 June 2003, <www.rotorhub.com/resource/products/uav.htm#>
44. Ferguson, G. 2003, 'Air 7000 – Beyond the Orion', *Australian Defence Magazine*, Australia.
45. Braybrook, R. 2003, 'The UAV approaches maturity', *Asia-Pacific defence reporter*, vol. 23, no. 1, Australia.
46. US Battlefield 2000, *US Battlefield UAVs*, viewed 25 July 2003, <http://www.vectorsite.net/twuav7.html>
47. SPG Media Group PLC 2005, *SEAMOS - Unmanned Aerial Vehicle*, viewed 29 Dec 2005, <http://www.naval-technology.com/project_printable.asp?ProjectID=1552>.
48. *D'Gopher Operation*, viewed 29 Dec 2005, <http://www.dstarengineering.com/newsite/downloads/GopherUAV.pdf>.
49. Anon 2003, *Shephard's Unmanned Vehicles Handbook 2003*, Unmanned Vehicles, The Shephard Press, USA.
50. Anon 2004, *Shephard's Unmanned Vehicles Handbook 2004*, Unmanned Vehicles, The Shephard Press, USA.
51. Anon 2005, *Shephard's Unmanned Vehicles Handbook 2005*, Unmanned Vehicles, The Shephard Press, USA.
52. Arkell, B. 1963, *Helicopter and Hovercraft Design*, Weidenfeld & Nicholson, USA, pp.74-83.
53. Maisel, M.D., Giulianetti, D.J., & Dugan, D.C. 2000, 'The History of the XV-15 Tilt Rotor Research Aircraft, From Concept to Flight', *Monographs in Aerospace History-The NASA History Series*, USA.
54. Brasel, R. 1999, *Item specification for the vertical takeoff and landing tactical unmanned aerial vehicle (VTUAV)*, viewed 20 July 2003, <http://www.fas.org/irp/program/collect/vtuav.htm>.
55. Moir, I. Seabridge, A. 2004, *Design and Development of Aircraft Systems*, Professional engineering publishing, London, U.K.
56. Civil Aviation Safety Regulations 2002, *Unmanned Aircrafts and Rocket Operations*, viewed 07 December 2005, <http://rrp.casa.gov.au/casrcreate/101.asp>.
57. Prouty, R.W. 1985, *Helicopter Aerodynamics*, PJS Publication, Peoria, Boston, USA.
58. Prouty, R.W. 1988, *More Helicopter Aerodynamics*, PJS Publication, Peoria, Boston, USA.

59. Prouty, R.W. 1993, *Even More Helicopter Aerodynamics*, Philip Business Information Inc, Potomac, USA.
60. Prouty, R.W. 1998, *Helicopter Performance Stability and Control*, PWS Publishers, Boston, USA.
61. Civil Aviation Safety Regulations 1998, *Airworthiness standards for rotorcraft in the normal category*, viewed 22 July 2005, <http://rrp.casa.gov.au/casr/027.asp>.
62. Civil Aviation Safety Regulations 1998, *Airworthiness standards for rotorcraft in the transport category*, viewed 22 July 2005, <http://rrp.casa.gov.au/casr/029.asp>
63. Word Reference Dictionary 2005, *Automation*, viewed 21 Dec 2005, <http://www.wordreference.com/definition/automation>.
64. Grover, P. 2005, *Introduction to Automation*, Automated Production Systems, viewed 11 Dec 2005, <http://www.utm.edu/departments/engin/lemaster/Auto%20Prod%20Sys/Lecture%2001.pdf>.
65. Word Reference 2002, *Introduction to Expert Systems Tutorial*, Module 1, viewed 21 Dec 2005, <http://www.wordreference.com/definition/automation>.
66. Belleuve Linux User Group 2004, *GUI*, viewed 21 Dec 2005, <http://www.bellevuelinux.org/gui.html>.
67. Word Reference Dictionary 2005, *Programming language*, viewed 22 Dec 2005, <http://en.wikipedia.org/wiki/Programming_language>.
68. Huston, S.D., Schimdt, D.C. 2005, *Design Challenges Middleware Solutions and ACE*, Pearson Education, viewed 22 Dec 2005, <http://www.informit.com/articles/article.asp?p=25486&rl=1>.
69. Word Reference Dictionary 2005, *Relational DBMS*, viewed 21 Dec 2005, <http://en.wikipedia.org/wiki/Database_management_system#Relational_DBMS>
70. Word Reference Dictionary 2005, *Off-the-shelf*, viewed 21 Dec 2005, <http://en.wikipedia.org/wiki/Off_the_shelf>.
71. Word Reference Dictionary 2005, *AutoCAD*, viewed 23 Dec 2005, <http://en.wikipedia.org/wiki/AutoCAD>.
72. Word Reference Dictionary 2005, *CATIA*, viewed 23 Dec 2005, <http://en.wikipedia.org/wiki/Catia>.
73. Word Reference Dictionary 2005, *Solid Works*, viewed 23 Dec 2005, <http://en.wikipedia.org/wiki/Solidworks>.
74. Word Reference Dictionary 2005, *MATLAB*, viewed 23 Dec 2005, <http://en.wikipedia.org/wiki/Matlab>.

75. Word Reference Dictionary 2005, *Programming Paradigm*, viewed 23 Dec 2005, <http://en.wikipedia.org/wiki/Programming_paradigm>.
76. PCI Geomatics 2005, *Open Database Connectivity*, Supported File Formats, viewed 25 Dec 2005, <http://www.pcigeomatics.com/cgi-bin/pcihlp/GDB%7CSupported+File+Formats%7CODBC+(Open+Database+Connectivity)>.
77. Archer, T. 2005, *Tom Archer on Visual C++*, Microsoft Visual C++ Developer Centre, viewed 22 Jan 2006, <http://msdn.microsoft.com/visualc/>.
78. Microsoft Corporation 2006, *SQL Server Bulks Up*, Microsoft Windows Server System, viewed 22 Feb 2006, <http://www.microsoft.com/sql/default.mspx>.
79. Microsoft Corporation 2005, *Get the .NET Framework 1.1*, Microsoft .NET Framework Developer Centre, viewed 10 September 2005, <http://msdn.microsoft.com/netframework/downloads/framework1_1>.
80. Microsoft Corporation 2005, *How to Use File DSNs and DSN-less Connections?*, Microsoft Help and Support, viewed 12 September 2005, <http://support.microsoft.com/kb/q165866/>.
81. Microsoft Corporation 2005, *Perform a Regression Analysis*, Microsoft Office, viewed 14 September 2005, <http://office.microsoft.com/en-us/assistance/HA011119631033.aspx#Expo>.
82. Microsoft Corporation 2005, *TREND*, Microsoft Office Online, viewed 20 Dec 2005, <http://office.microsoft.com/en-au/assistance/HP052093201033.aspx>.
83. Microsoft Corporation 2005, *FORECAST*, Microsoft Office Online, viewed 20 Dec 2005, <http://office.microsoft.com/en-au/assistance/HP052090961033.aspx>.
84. Microsoft Corporation 2005, *GROWTH*, Microsoft Office Online, viewed 20 Dec 2005, <http://office.microsoft.com/en-au/assistance/HP052091081033.aspx>.

APPENDIX A

Table A.1: Mission System Technology for Vertical Take-off UAV (Anon, 2004)

Commercial product	Mission system technology	Weight (Kg)	Functional characteristics
SONAIR	Audio sensitiser	1	Detects anti-aircraft artillery
HRN-1	Communication node	2.3	Establishes uni directional interaction between base and airborne object
HRN-2	Communication node	4	Establishes dual directional interaction between base and airborne object
TCDL	Communication channel	4.5	Assists in modular packaging via tactical data link
TVL	Video link	4	Provides real-time data for target tracking, damage assessment, situational awareness and battlefield overview
TVL – II	Video link	4.2	Provides delay less video and telemetry data transmission to ground station
StarLink	Miniature data link	3	Develops wireless systems network amongst airborne objects by incorporating group expertise
2500-Tx microwave & audio transmitter	Video and audio transmitter	5	Transmits audio/video data via remotely accessed multi-channels at 900 - 2700 MHz
ACN	Connectivity nodule	4	Provides in-theatre communication and out-of theatre reach-back connectivity via joint/coalition interoperability
EL/K-1865	Data terminal	10	Provides full motion video based on its modular architecture
VCS700	Data workstation	1	Provides NTSC/PAL real-time colour video with no image degradation
STAR	Data recorder	15	Stores strategic and tactical data based on open system architecture
SSMDR	Digital data recorder	4	Digitises data and stores it in solid-state format
COMINT	Interceptor node	45	Provides electronic order of battle with wideband interception
11ST	Processing system	20	Processes digital video received via EO/IR channel based on its step-stare system
AN/ASX-4	EO/IR sensor	97	Captures data with 3CCD day sensor vis laser pointer and illuminator.
AURORA 600	Acquisition / processing system	31	Obtains and processes data in real-time based on revolutionary algorithm
AURORA 600 sensor	Sensor	7.5	Works in conjunction with AURORA 600 processing system and operates in UV, EO and UV-VNIR spectral range
BRITE Star	Sensor system	51	Works as a laser designated , long range reconnaissance system based on its fully automatic bore-sight module
CYCLOPE 2000	Line-scan sensor	6	Used for multi spectral detection stereoscopy
DSP I	Dual sensor	26	Captures information in NTSC/PAL format through IR link
MicroSTAR II	Triple sensor	9.5	Operates as a complete surveillance and reconnaissance system
MOSP	Day/Night sensor	35	Provides all-weather and all-time operations
POP	Optical and electronic sensor	14	Operates in visible and infrared bands and provides colour CCD output
TOPLITE	Multi sensor	10	Provides high stability during 24-hour capture mode
Uncooled Thermal Imager	Imager	5	Operates in pitch-able dual field of view to generate standard CCIR video output
COMPASS	Observation system	21	Works independently as a compact multi purpose advanced stabilised day/night observation system

Commercial product	Mission system technology	Weight (Kg)	Functional characteristics
ESP-600C	Day observation system	12.3	Provides 360 degree continuous azimuthal angle on its 2 high resolution colour CCD cameras
GOSHAWK SERIES	FLIR sensor	30	Operates as a gyro stabilised UAV payload with its forward looking infra red sensor, primarily serves military
Miniature Pan-Tilt Zoom Daylight Colour TV	Display Unit	1.2	Displays using RS_170 video format, mounted on 360 degree left/right pan and 90 degree up/down tilt
Type 8042	Reconnaissance sensor	62	Offers ultra high resolution, optimised for EO/IR imaging
2553 EMC Class A	Zoom camera	1.73	Used for EOD robots & military surveillances
A-CAM 2	Camera system	4.2	Used as gyro stabilised camera system handling 4 different video sources, remotely controlled from RS232 port
Alti-Cam	Camera and turret system	0.7	Provides NTSC video output through 680000 pixel colour CCD
CA-270	Reconnaissance camera	16	Operates in dual spectral band of EO and IR and uses 3 phase power at 400 Hz
CA-295	Reconnaissance camera	18	Operates in EO and IR band and uses AC/DC dual power
Camelia	Infrared line-scan camera	5	Used as a single-axis scanning system with FOV of 60 degree
CRR-1210-L1	CMOS sensor	1	Used in harsh environment, captures at high video refresh rate
Omega	IR camera	0.1	Operates as a 160 x 120 micro-bolometer in 8-12 um thermal band
Pan-Tilt Zoom Camera with laser rangefinder	Surveillance system	5.3	Uses a 1/3 inch CCD colour sensor to capture in NTSC/PAL format
Phoenix	IR camera	3.2	Outputs as analog video in 14-bit digital data format
TMX6	Camera module	1	Operates as a rugged camera module with remote sensor head, and offers video contrast enhancement
ROCLITE	Colour/mono zoom camera	0.9	Provides pan/tilt option and operates from a carbon fibre body sealed, pressurised with nitrogen
RollVision - Camera holder	Camera holder	7	Works as a 3-axes gyro stabilised camera system, with 360 degree pan and 180 degree roll, using microwave for communication
RollVision – Complete system	Camera system	9	Provides interchangeable camera formats from state-of-art camera with a 3-axes gyro stabilised system
AES – 210 V	Electronic warfare payload	22	Provides ELINT & ESM capability, and remote operation from GCS
Airborne Communications Jamming Payload	Electronic communication module	14	Operates in 4 Jamming modes of spot, responsive, deception, & barrage, with an optional remote control
Electronic Surveillance Payload (ESP)	Surveillance node	16	Operates as an emitter location system at 0.5 to 18 GHz frequency
Ultralite	Open system	7	Interfaces with other organic systems to provide COMINT and ELINT compatibility
ALR-93 (V)	Electronic surveillance module	49	Operates as a RWR or ESM system contiguously over 0.5 - 20 GHz to provide amplified crystal video with pulse compression
CATS	Threat surveyor	15	Provides robust digital technology based on multiple receiver architecture, validated by real time geo-location algorithms
TOP-SCAN	ELINT/ESM system	20	Used as a dual axis interferometer system to localise RF emitters and low weight system with patented algorithms in a dense electromagnetic operational environment

Commercial product	Mission system technology	Weight (Kg)	Functional characteristics
Ultralight ESM	SIGINT system	24	Compact SIGINT system used with a single circular array interferometer antenna to provide 360 degree coverage
VICON 78	Dispensing system	N.A.	Used as a countermeasure dispensing system that offers a safe, reliable and cost effective solution to enhances survivability
ALE-50 Towed Decoy System	Radar decoy	N.A.	Used as an expendable active radar decoy, includes launch controller, launcher and towed decoy, that interfaces with an aircraft over standard database
Tactical Radar Jammer	Jammer	46	Supports jamming and self protection with real time programmability in a forced air cooling environment
ASELFLIR – 200	Imaging system	85	An airborne thermal imaging system, used for multiple purposes, including pilotage, navigation, surveillance, & search-rescue missions
FOX-450Z/ZE	Imaging camera	10	Operates as a thermal imaging camera with zoom lens and MWIR sensor type to offer 3rd generation FPA facilities
ISS AeroScout	Multi-sensor	33	Provides enhanced image processing through a micro scanner using mid-wave IR and laser ranging
Omega Camera	Infrared micro-sensor	0.11	Uses an uncooled micro-bolometer sensor to capture RS-170A video output at 30 Hz
Night Conqueror	Thermal imaging system	0.8	Achieves mission critical performances with a forward looking thermal imaging system that uses 1 Mega-pixel and 640 x 512 pixels staring array to generate 8-bit analog and 14-bit digital video
SCC500 micro IR camera core	Imaging system	3	Serves military as a high performance, compact, lightweight and configurable infrared imaging system
White Star 2751	Surveillance system	13.5	Used a high resolution uncooled Forward Looking Infra-Red camera to offer airborne surveillance
2604 Mini DV airborne VCR	On-Board UAV system recorder	0.8	Operates as a front loading VCR to provide high resolution stills, packaged with a built in heater, full function remote control panel and a multi deck controller
ADT	Air-data Terminal	0.9	Operates as a 4th generation, fully integrated data-link system to provide multi-mission and SAR imaging capability
ATX-2740 (v)	Data-link system	14	Operates as a light-weight exportable data link system, meets NATO STANAG 7085 standards, to provide real time, full duplex sensor data and voice communication
BLU-108 Sensor Fused Sub-munition	IR and Laser sensor	28	Uses 4 smart skeet warheads and dual mode passive infrared and active laser sensors to operate in redundant self destruct logic modes
GuideStar (GS-111)	Flight control system	0.6	Provides inertial navigation through eight 14 bit analogue channels and stores data in 48 MB solid state, non-volatile hard-drive
Golden Bay	Real time processing system	2.2	Offers time-critical targeting in day/night operations by interfacing with scanning pod / observation device that minimising sensor-to-shooter cycle time
MP2028	Autopilot unit	0.03	Smallest miniature UAV autopilot that is used in micro-pilot UAVs
UMAS	Multi application on-board system	1.3	Uses a 32-bit processor, 9 input / 5 output serial ports, and 8 digital outputs to provide data processing
Montage	Modular target control on-ground unit	N.A.	Based on a distributable multi-processor architecture, it can expand to accommodate flights of upto 8(16) different targets, each operating with an independent data link
MRS-2000	Digital packing receiving system	N.A.	Fully integrated control and communication system, based on MRS and MCS , that provides non-LOS view of the enemy

Commercial product	Mission system technology	Weight (Kg)	Functional characteristics
TCS	Tactical Control System	N.A.	Shelterised as a land based system, it provides 5 levels of TCS functionality
WePilot1000	Flight control system	N.A.	Used for small remote controlled helicopters, it contains a flight control processor, built in computer system, & GPS receiver to offer a 6-degree of freedom inertial measurement
EL/M-2055	Reconnaissance payload	N.A.	Used for high resolution real-time, day/night, full-duplex imagery
LAIRS	Advanced imaging radar system	45	Operates in 3 imaging modes of target detection for navigation
MMWR	Wave radar	21	Acquires and tracks stationary and airborne tactical targets
Mk 4	Miniature radar altimeter	0.4	Operates upto 700 metres with a 12.5 cm performance accuracy
Modular AN/APS-144 multimode radar	Multimode radar	34	Operational modes include surface moving target detection, imaging synthetic aperture and airborne intercept
STacSAR	Small tactical synthetic aperture radar	30	Provides fixed target imagery in both spotlight and search modes through real-time data exploitation allowed by onboard processor
AN/APY-8	High resolution radar	40	Provides precision mapping and high resolution through a Ku band radar with unique intelligence analysis and targeting capabilities
Tactical UAV radar	Tactical radar	30	Operates in 2 operational modes of strip & spot, and MTI
TESAR	Surveillance radar	35	Operates in SAR mode that is continuous and fully focussed, providing high-resolution within 25 metres of target

APPENDIX B

Mission System Technology for Vertical Take-off Unmanned Aerial Vehicles (Anon, 2004)

1. Acoustic Sensors

Sensor Array

- **SONAIR**
  - Acoustic sensor array for detecting and locating anti-aircraft artillery;
  - 1 Kg processor;
  - 6 microphone fitting;
  - Power: < 10 watts; and
  - Suitable for UAVs airspeeds up to 400 Km/h

2. Communications

Transponders

- **HRN-1**
  - Airborne directional antenna pedestal;
  - Velocity: 30 deg/sec;
  - Positioning range: 360 deg x N;
  - Positioning accuracy: 0.7 deg;
  - Operation frequency: 4.4 to 5.1 GHz;
  - Weight: 2.3 Kg; and
  - Input voltage: 15 V / 0.4 Amp, 5 V / 0.5 Amp, 28 V / 1 Amp

- **HRN-2**
  - Airborne dual directional antenna pedestal;
  - Velocity: Lower antenna - 20 deg/sec, Upper antenna - 30 deg/sec;
  - Positioning range: 360 deg x N;
  - Positioning accuracy: 0.7 deg;
  - Operation frequency: 4.4 to 5.1 GHz;
  - Weight: 4 Kg; and
  - Input voltage: 15V/0.8Amp, 5V/0.5Amp, 28V/1.7Amp

- **TCDL**
  - Tactical common data link;
  - DSP driven CDL programmable data rates, adjustable during flight;
  - Modular packaging;
  - Flexible interfaces: RS-422, Ethernet, Fibre Optic, RS232;
  - Weight: 4.5 Kg;
  - Power: < 125 W;
  - Full duplex transmitter/receiver; and
  - Frequency band: Ku-, X- / user select

- **TVL**
  - Tactical video link;
  - Lightweight video receiver system;
  - Downloads video and telemetry data;
  - Provides real-time data for target tracking, damage assessment, situational awareness and overview of battlefield; and
  - Weight: 4 Kg

- **TVL-II**
  - Tactical video link;
  - Revolutionary data reception speed;
  - Eliminates delay during video and telemetry data transmission to HQ; and
  - Weight: 4.2 Kg

- **StarLink**
  - Data link for mini-UAVs;
  - Incorporating groups expertise;
  - Used for developing wireless systems;
  - Comprises of air data terminal units, ground data terminals and air data relays; and
  - Weight: 3 Kg

Transmitters

- **2500-Tx Microwave & Audio Transmitter**
  - Microwave video and audio transmitter;
  - Multi-channel with option for remote channel selection;
  - Frequency bands from 900-2700

MHz;
- Power: 1W, dual power version;
- Weight: 5 Kg; and
- Carbon housing, waterproof and fireproof

- **ACN**
  - Airborne communication node;
  - In-theatre communication and out-of theatre reach-back connectivity;
  - Supports SIGINT (SIGnal INTelligence); and
  - Supports joint/coalition interoperability

Terminal
- **EL/K-1865**
  - Compact air terminal;
  - Transceiver: 210 x 300 x 160 mm, Front end: 382 x 260 x 160 mm (W x D x H);
  - Modular architecture;
  - Frequencies: 1500 MHz, Baseline: 600 MHz;
  - Power:100-250 W; and
  - Weight:2.5 to10 Kg

- **VCS700**
  - Real time colour video compression system;
  - Compression of full-motion video with no visible image degradation;
  - Programmable compression ratio of 350:1;
  - User programmable video format via RS-232 port;
  - NTSC and PAL colour compatibility; and
  - Weight: 1 Kg

Recorder
- **S/TAR**
  - Strategic tactical airborne recorder;
  - High data-rate, large capacity system designed for use on strategic/tactical UAVs;
  - Fast data access;
  - Multi-sensor, simultaneous record/play;
  - Supplements open-system architecture;
  - Circuit modules: 6U-160 mm; and
  - Weight:15 Kg

- **SSMDR**
  - Solid state mission data recorder;
  - Digital data recorder;
  - 50 GB of storage;
  - 480 MB/s transfer rate;
  - Power = 200 W;
  - Dimensions: 8.7 in x 6.0 in x 3.75 in; and
  - Weight = 4 Kg

Interceptor
- **COMINT Payload**
  - Designed for MALE/HALE UAV;
  - Provides electronic order of battle;
  - Wideband interception;
  - Automatic communications activity, alert and targeting capabilities on VHF & UHF{30-300 MHz}; and
  - Weight: 45 Kg

3. Electro-Optical System

Imaging Processor / Sensor / Sensor System
- **11ST**
  - M11 digital video electro-optical/infrared(EO/IR) countermine step-stare system;
  - Active gyro-stabilisation (4-axis);
  - Turret dimension (in): 11(d) x 14.5(h);
  - Power: 85/170 W; and
  - Weight:17.7 Kg (Normal), 20 Kg (UAV)

- **AN/ASX-4**
  - Optical/IR sensor;
  - Colour 3CCD day sensor with 800 lines resolution;
  - NTSC/PAL format;
  - Laser range finder;
  - Pointer/Illuminator option;
  - Dimensions (cm): 53(d) x 67(h);

- Power: 500 W normal / 1000 W max; and
- Weight: 97 Kg

- **AURORA 600**
  - Acquisition/Processing system
  - Attachment to the Aurora 600 sensor;
  - Adaptive real time, sub-pixel ATR;
  - Dimensions: 8.5 in x 17.5 in x 17.25 in (W x L x H);
  - Power: 540 W; and
  - Weight: 31 Kg

- **AURORA 600 sensor**
  - HSI sensor for the ultraviolet, visible and UV-VNIR spectral range;
  - Sensor model 600;
  - Sp Channel 480;
  - Cross track pixels 640;
  - Frame rate 0-200 fps;
  - Spectral Range 600 nm;
  - Power: 50 W; and
  - Weight: 7.5 Kg

- **BRITE Star**
  - Multiple sensor;
  - Laser designated and long range (Thermal/visible) reconnaissance system;
  - Onboard integrated laser cooling;
  - Fully-automatic integrated bore-sight module;
  - Turret Dimensions (cm): 23(d) x 31(h); and
  - Weight: 51 Kg

- **CYCLOPE 2000**
  - Infrared line-scan sensor;
  - Multi-spectral detection stereoscopy;
  - Resolution: 0.5 to 1 mrad;
  - Dimension (cm):17 x 17 x 20; and
  - Weight: 6 Kg

- **DSP-1**
  - Dual sensor stabilised payload;
  - 4 gimbals with +10 to -105 deg elevation;
  - IR sensor: 3-5 micron;
  - NTSC/PAL format;
  - 786 x 494 pixel CCD TV;
  - 20x zoom lens;
  - Power: 28 VDC, 110 W;
  - Dimensions(cm): 32(d) x 50(h); and
  - Weight: 26 Kg

- **MicroSTAR 11**
  - 3-sensor surveillance and reconnaissance system (Sensor 1 : 3 to 5 um, Sensor 2 : long range, Sensor 3: 1.54 um);
  - Power: 18-32 VDC;
  - Turret Dimensions (cm): 30(d) x 35(h); and
  - Weight: 9.5 Kg

- **MOSP**
  - Day/Night sensor payload;
  - Dimensions (in): 14(d) x 21(h);
  - Power: 350 W; and
  - Weight: 35 Kg

- **POP**
  - Optical and electronic payload operating in visible and infrared bands;
  - Day/night configurations sensors with colour CCD output;
  - Dimensions (in): 10(d) x 15(h);
  - Power: 95 W; and
  - Weight: 14 Kg

- **TOPLITE**
  - Highly stabilised;
  - Multi sensor;
  - Optical and electronic payload;
  - Day/Night observation;
  - Easy installation;
  - Easy maintenance; and
  - Weight: 10 Kg

- **Uncooled Thermal Imager**
  - Pitch-able dual field of view;
  - Resolution: 384 x 288 pixels;
  - Dual FOV;
  - Standard CCIR video output;
  - Temperature: -10 to +50 deg;

- Dimensions : < 192 mm(d);
- Power: 15 W; and
- Weight: 5 Kg

Display Units / Observation Systems

- **COMPASS**
  - Compact multi-purpose advanced stabilised day/night observation system;
  - 3rd generation CCD TV camera;
  - 3-5 um FLIR camera;
  - Laser pointer, laser rangefinder; and
  - Weight: 21 Kg

- **ESP-600C**
  - Stabilised daylight observation system;
  - 360 degree continuous in azimuth;
  - +10 to -110 degree in elevation;
  - 2 high resolution colour CCD cameras;
  - Dimensions(cm): 30(d) x 44(h); and
  - Weight:12.3 Kg

- **GOSHAWK SERIES**
  - Military specification gyro-stabilised UAV payload;
  - FLIR sensors;
  - Zoom TV cameras;
  - Power: 600/400 W (Peak/RMS);
  - Turret dimensions(cm): 35 x 41; and
  - Weight: 30 Kg

- **Miniature Pan-Tilt Zoom Daylight Colour TV**
  - Left/right pan movement (360 degree);
  - Up/down tilt movement (0-90 degree);
  - Sony EVI electronically stabilised TV;
  - Dimensions:10in x 6in x 8in (L x W x H);
  - Power: 12V;
  - Video Format: RS_170; and
  - Weight: 1.2 Kg

- **Type 8042**
  - Ultra High resolution (450mm focal length, 8 micron CCD, 12288 pixel linear array);
  - Reconnaissance sensor;
  - Optimised for visual band/IR imaging;
  - 12.4 deg FOV;
  - Dimensions (cm): 81 x 35 x 31;
  - Power: 115 V at 400 Hz; and
  - Weight: 62 Kg

Camera / Modular Accessories / Complete System

- **2553 EMC Class A**
  - Robust colour/mono zoom camera;
  - Suitable for EOD robots & military surveillance system;
  - Thermal insulation;
  - 2.7 to 48 degree zoom range;
  - 460 line resolution; and
  - Weight: 1.73 Kg

- **A-CAM 2**
  - Gyro stabilised camera system;
  - Remote controlled via RS232;
  - Handles upto 4 different video sources;
  - Dimension (cm): 9.4(h) x 28.4(w);
  - Power: 20-36 V;
  - Consumption: 0.3 - 1.0 A @ 24 V; and
  - Weight: 4.2 Kg

- **Alti-Cam**
  - Inertial stabilised camera and turret system;
  - 680000 pixel colour CCD;
  - NTSC video output;
  - Zoom: 25x optical, 4x digital;
  - Exposure: 1/4 sec to 1/10000 sec;
  - Dimensions: 7 inch spherical radius;
  - Power: 7 W; and
  - Weight: 0.7 Kg

- **CA-270 Digital Reconnaissance Camera**
  - Dual spectral band (Visible/IR)

aerial reconnaissance camera;
- Dimensions (in): 51 x 20.5 x 20.5;
- Power: 3 phase, 400 Hz, 1018 W, 28 V dc, 20 W; and
- Weight: 16 Kg

- **CA-295 Digital Reconnaissance Camera**
  - Dual spectral band (Visible/IR) aerial reconnaissance camera;
  - Dimensions (in): 51 x 20.5 x 20.5;
  - Power: 3 phase, 400 Hz, 1018 W, 28 V dc, 20 W; and
  - Weight: 18 Kg

- **Camelia**
  - Infrared line scan camera;
  - Single axis scanning system;
  - 3.5 microns infrared detector;
  - FOV: 60 degree;
  - Standard CCIR channel;
  - Power: 18-72 VDC at 60 Watts;
  - Dimensions (cm): 23 x 17 x 16; and
  - Weight: 5 Kg

- **CRR-1210-L1**
  - Harsh environment installation;
  - 1280 x 1024 CMOS sensor;
  - Video refresh rate: 30 frames/sec; and
  - Weight: 1 Kg

- **Omega**
  - Small-low power infrared camera;
  - 160 x 120 micro-bolometer operating in the 8-12 um thermal band;
  - Sensitivity: 80mK;
  - Uses f1.6 optical fibre;
  - Power: 800mW;
  - Dimensions (cu inches): 3; and
  - Weight: 0.1 Kg

- **Pan-Tilt Zoom Camera with laser rangefinder**
  - Surveillance system with laser rangefinder;
  - 1/3 inch CCD colour sensor;
  - NTSC/PAL format;
  - Resolution: > 380 lines;
  - Dimensions: 9 in x 16 in x 11 in (L x W x H);
  - Power: 12 V; and
  - Weight: 5.3 Kg

- **Phoenix**
  - High performance, modular infrared camera;
  - Available in 320 x 256 and 640 x 512 SWIR, MWIR and LWIR versions;
  - Dimensions (in): 5 x 4.5 x 6.5;
  - Power: 18 to 36 VDC;
  - Output: Analog video (RS-170/CCIR) / 14-bit digital data; and
  - Weight: 3.2 Kg

- **TMX6**
  - Rugged camera module with remote sensor head;
  - Designed for integration with airborne payloads;
  - Automatic shutter;
  - Video contrast enhancement;
  - Operational temp: 25 deg C to 55 deg C;
  - Power: < 7 W; and
  - Weight: 1 Kg

- **ROCLITE**
  - Light weight, colour/mono zoom camera;
  - Pan/tilt option;
  - Carbon fibre body sealed and pressurised with nitrogen;
  - 18x optical zoom with 4x digital zoom;
  - 2.7 deg to 48 deg zoom range; and
  - Weight: 0.9 Kg

- **RollVision - Camera holder**
  - 3-axes gyro stabilised camera system;
  - 360 degree pan, 180 degree roll;
  - Dimensions (in) (L x W x H): 14.5 x 19 x 19.5;
  - Power: Microwave - 2.4GHz;
  - Li-Battery: 12/24V; and

- Weight: 7 Kg

- **RollVision - Complete system**
  - State-of-art camera with a 3-axes gyro stabilised camera system for interchangeable camera formats;
  - 360-degree pan, 180 degree roll; and
  - Weight: 9 Kg

4. EW Systems

Modular Payload

- **AES - 210 V**
  - Light weight, modular and easy installation;
  - ELINT and ESM capability;
  - Remote operation from GCS;
  - Fine data frequency;
  - Windows NT based MML;
  - Temp: -55 to +70 C;
  - Power: 28 VDC, 250W; and
  - Weight: 22 Kg

- **Airborne Communications Jamming Payload**
  - ECM payload is derived from TRC-274;
  - Compact airborne packaging;
  - Jamming modes: Spot, responsive, deception, & barrage;
  - Dimensions (cm): 37 x 22 x 21;
  - Optional remote control; and
  - Weight: 14 Kg

- **Electronic Surveillance Payload (ESP)**
  - Emitter location system;
  - Dimensions (cm): 34 x 13 x 19;
  - Frequency: 0.5 to18 GHz; and
  - Weight:16 Kg

- **Ultralite**
  - JASA compliant;
  - Open system;
  - COMINT / ELINT compatibility;
  - Interface with other organic systems; and
  - Weight: 7 Kg

RWR / ESM / SIGNIT System

- **ALR-93 (V)**
  - RWR and ESM system;
  - Contiguous over 0.5 - 20 GHz;
  - Amplified crystal video;
  - IFM, SHR receiver types;
  - 360 deg azimuthal coverage;
  - Pulse compression; and
  - Weight: 49 Kg

- **Compact Airborne Threat Surveyor (CATS)**
  - Robust digital technology;
  - Multiple receiver architecture;
  - Validated real time geo-location algorithm;
  - Library: >5000 modes;
  - Size (Processing unit): ARINC 600 (6 MCU);
  - Power: 260 W; and
  - Weight: 15 Kg

- **TOP-SCAN**
  - Airborne ELINT/ESM System;
  - Dual axis interferometer system;
  - Localises RF emitters;
  - Low weight system;
  - Patented algorithms;
  - Dense electromagnetic operational environment; and
  - Weight: 20 Kg

- **Ultralight ESM**
  - Compact SIGINT system;
  - Single circular array interferometer antenna;
  - 360 deg coverage;
  - Frequency: 2-18 GHz;
  - Bandwidth: 4 GHz;
  - Directional accuracy: < 2 deg;
  - Sensitivity: 65 to 75 dB; and
  - Weight: 24 Kg

- **VICON 78**
  - Countermeasure dispensing system;
  - Safe, reliable and cost effective solution;
  - Enhances survivability;
  - 6 payload types per dispenser; and

- Upto 28 payload types can be carried

Radar Decoys / Jammer

- **ALE-50 Towed Decoy System**
  - Expendable active radar decoy;
  - Includes launch controller, launcher and towed decoy;
  - Interfaces with aircraft over standard database;
  - Decoy packed in sealed canister; and
  - 10 year shelf life of decoy

- **Tactical Radar Jammer**
  - Supports jamming and self protection;
  - Frequency: 0.6 to18 GHz with 0.1 to 40 GHz as an option;
  - Real time programmability;
  - Forced air cooling;
  - Power: 10W (0.5 to 18 GHz) / 100W(6 to 18 GHz); and
  - Weight: 46 Kg

5. FLIRS / Thermal Imaging Systems

- **ASELFLIR – 200**
  - Airborne thermal imaging system;
  - Multipurpose, thermal imaging sensor for pilotage/navigation;
  - Surveillance, search-rescue missions;
  - Parallel detector channels: 240 x 4 FPA;
  - Spectral Band: 8-12 um;
  - Electronic zoom: 2:1 / 4:1;
  - Turret Dimensions(in): 12.75(d) x 14.68(h); and
  - Weight: 85 Kg

- **FOX-450Z/ZE**
  - Thermal imaging camera with zoom lens;
  - Sensor type: MWIR;
  - 3rd generation FPA;
  - Spectral range: 3-5 microns;
  - Number of pixels: 320 x 256;
  - Lens Type: Continuous;
  - 22.5x optical zoom;
  - Dimensions(cm):19(w) x 21(h) x 35(l);
  - Power: 28 VDC/30 W; and
  - Weight:10 Kg

- **ISS AeroScout**
  - Airborne multi-sensor;
  - Provides mid-wave IR;
  - Laser ranging;
  - Performance specification: 3 FOV optical telescope;
  - 4x micro-scanner;
  - Enhanced image processing;
  - RS-232/422 control;
  - RS-170 video output;
  - Dimensions(cm): 37 x 38 x 53;
  - Power: 28 VDC / 200 W; and
  - Weight: 33 Kg

- **Omega Camera - Infrared Micro-sensor**
  - Uncooled micro-bolometer sensor;
  - RS-170A video output at 30 Hz;
  - CCIR option at 25 Hz;
  - Temperature: -40 deg C to +55 deg C;
  - Dimensions (cm^3): 77;
  - Power: 1.5 W;
  - Standard input voltage range of 3.5 to 9 VDC;
  - Used as IR payload for Dragon Eye; and
  - Weight: 0.11 Kg

- **Night Conqueror**
  - Forward looking thermal imaging system;
  - 1 Mega-pixel and 640 x 512 pixels staring array;
  - 8-bit analog and 14-bit digital video;
  - Mission critical performance;
  - Dimensions(in): 3 x 3 x 4;
  - Power: 15 W; and
  - Weight: 0.8 Kg

- **SCC500 micro IR camera core**
  - High performance infrared imaging;

- Compact, lightweight and configurable package;
- Low cost IR camera;
- Primary clientele is military;
- Real time 60Hz frame rate; and
- Weight: 3 Kg

o **White Star 2751**
- Airborne surveillance system;
- High resolution uncooled FLIR;
- IR camera;
- SONY colour camera;
- FCP 470;
- Video zoom and 6deg/24deg dual FOV IR camera;
- Gyro-stabilised composite gimbal;
- Dimensions(cm): 27 x 39;
- Power: 18-35 VDC;
- Fuse: 10A; and
- Weight:13.5 Kg

6. Integrated Systems

On Board UAV System

o **2604 Mini DV airborne VCR**
- Front loading VCR;
- High resolution, perfect stills;
- 146 mm wide;
- Built in heater;
- Full function remote control panel;
- Multi deck controller; and
- Weight: 0.8 Kg

o **Air data Terminal**
- 4 generation, fully integrated data-link system;
- Multi-mission capability;
- SAR imaging;
- Digital data rate: 3 - 45 Mbps;
- Range: 250 km (LOS);
- Frequency band: UHF, L, S, C, X, and Ku;
- Frequency selection: Synthesised;
- Modes of operation: Analogue/Digital;
- Immunity: Jam resistant;
- Power: 28 V; and
- Weight: 0.9 Kg

o **ATX-2740 (v)**
- Light weight exportable data link system;
- Robust COTS;
- Meets NATO STANAG 7085 standards;
- Real time, full duplex sensor data and voice communication;
- Frequency: X-band / Ku-band;
- TWTA transmitter;
- Integral multiplexing/de-multiplexing; and
- Weight:14 Kg

o **BLU-108 Sensor Fused Sub-munition**
- 4 smart skeet warheads;
- Dual mode passive infrared and active laser sensors;
- Built-in redundant self destruct logic modes;
- Dimensions (in): 31(l) x 5(d); and
- Weight: 28 Kg

o **GuideStar (GS-111)**
- Integrated digital flight control system;
- Inertial navigation;
- Power: 5 W;
- Voltage: 18-50 VDC;
- Serial I/O: 2 RS-232 (115 k baud), 2 RS-232 (250 k baud), 1 RS-232 (920 k baud);
- Analogue input: 8 channels – 14 bit A/D;
- 48 MB solid state, non-volatile data storage; and
- Weight: 0.6 Kg

o **Golden Bay**
- Provides real time processing of reconnaissance imagery;
- Accurate target generation rate;
- Minimises sensor-to-shooter cycle time;
- Offers time critical targeting;
- Interfaces with any scanning pod / observation device;
- Day/night operation; and
- Weight: 2.2 Kg

- **MP2028**
  - Newest product in a family of micropilot UAVs;
  - Smallest miniature UAV autopilot;
  - Dimensions (cm): 10(l) x 4(w); and
  - Weight: 0.03 Kg

- **UMAS - Unmanned multi application system**
  - Off-the-shelf, tum-key solution;
  - 32-bit processor;
  - 9 input / 5 output serial ports;
  - 8 digital outputs;
  - Dimensions (cm): 18 x 11 x 8;
  - Temperature: -40 to 71 C;
  - Non-operating shock;
  - Max. acceleration: 50G; and
  - Weight: 1.3 Kg

Ground system

- **Montage**
  - Modular networked target control environment;
  - Based on distributable multi-processor architecture;
  - Expandable to accommodate flights of upto 8(16) different targets;
  - Independent data link; and
  - Operations in UHF, L-, and C-band spectrum

- **MRS-2000**
  - Digital packing receiving system;
  - Fully integrated control and communication system;
  - Based on MRS and MCS;
  - Provides non-LOS view of the enemy; and
  - Pentium computing power based on a complete imaging awareness

- **Tactical Control System (TCS)**
  - Shelterised as land based system;
  - 5 levels of TCS functionality; and
  - Level 1 and 2 enable receipt of imagery, 3 & 4 provide command and control of the air vehicle, imagery and payload receipt, 5 provides the ability to takeoff and payload control

- **WePilot1000**
  - Flight control system for small remote controlled helicopters;
  - Flight control processor with a built in computer system;
  - GPS receiver; and
  - 6 degree of freedom inertial measurement unit

7. RADAR

- **EL/M-2055**
  - SAR / MTI UAV reconnaissance payload;
  - High resolution real time, day-night imagery;
  - Interface: RS-422A;
  - Full duplex;
  - Downlink output: 2.048 Mbps (Optional: 10.71 Mbps);
  - Power: 28 V dc; and
  - Weight: 56 Kg

- **LAIRS (Lockheed Martin Advanced Imaging Radar System)**
  - Synthetic aperture radar composed of both airborne and ground based subsystems;
  - 3 imaging modes;
  - Target detection mode;
  - Navigation: < 185 Km; and
  - Weight: 45 Kg

- **Milli-Metric Wave Radar (MMWR)**
  - Acquires and tracks fixed high-value targets;
  - Stationary and moving ground and airborne tactical targets;
  - Frequency modulated, FMCW radar utilising dual polarisation and high range resolution;
  - Dimensions (cm): 31 x 18; and
  - Weight: 21 Kg

- **Miniature Radar Altimeter Mk 4**
  - Small and robust;
  - RS232 output;
  - Performance: 12.5 cm accuracy;

- Operates upto 700 m;
- Power: 6 Watts (Optional:12/28); and
- Weight: 0.4 Kg

- **Modular An/APS-144 multimode radar**
  - Operational modes include surface moving target detection, imaging synthetic aperture and airborne intercept;
  - Frequency: J-band (12-18 GHz);
  - Azimuth beamwidth: 2 deg operation mode;
  - Long range search: 5-20 Km;
  - 75 m resolution;
  - 360 deg of sector scan;
  - Frequency: 6.25 KHz;
  - Short range search: 3-10 Km; and
  - Weight: 34 Kg

- **Small Tactical Synthetic Aperture Radar (STacSAR)**
  - Fixed target imagery in both spotlight and search modes;
  - Real-time exploitation allowed by onboard processor;
  - Wavelength: J-band (17 GHz);
  - Resolution: 0.5-50 m;
  - Slant range: 10 Km; and
  - Weight:30 Kg

- **AN/APY-8 Lynx Very High Resolution Radar**
  - SAR / GMTI / Precision mapping;
  - High resolution;
  - Ku band radar with unique intelligence analysis and targeting capabilities;
  - Operational mode: Baseline;
  - Higher performance system;
  - CLAW provides enhanced exploitation;
  - Power: Tailored;
  - Dimensions: Tailored; and
  - Weight: 40 Kg

- **Tactical Unmanned Aerial Vehicle Radar**
  - Radar provides 2 operational modes: Strip & spot, and MTI mode;
  - Target reports overlaid on digital mode;
  - RF Frequency: Ku-band;
  - Power: 474 W;
  - Maintainability: 2-level bit to LRM level; and
  - Weight: 30 Kg

- **TESAR (AN/SPQ-1)**
  - Surveillance radar;
  - Modes: Synthetic aperture radar and moving target indicator;
  - SAR mode is continuous, fully focus, high resolution, coordinates of each map centre are provided to within 25 metres CEP;
  - Assumes p-coded GPS; and
  - Weight: 35 Kg

APPENDIX C

Re-classification of Mission System Technology for Vertical Takeoff UAV

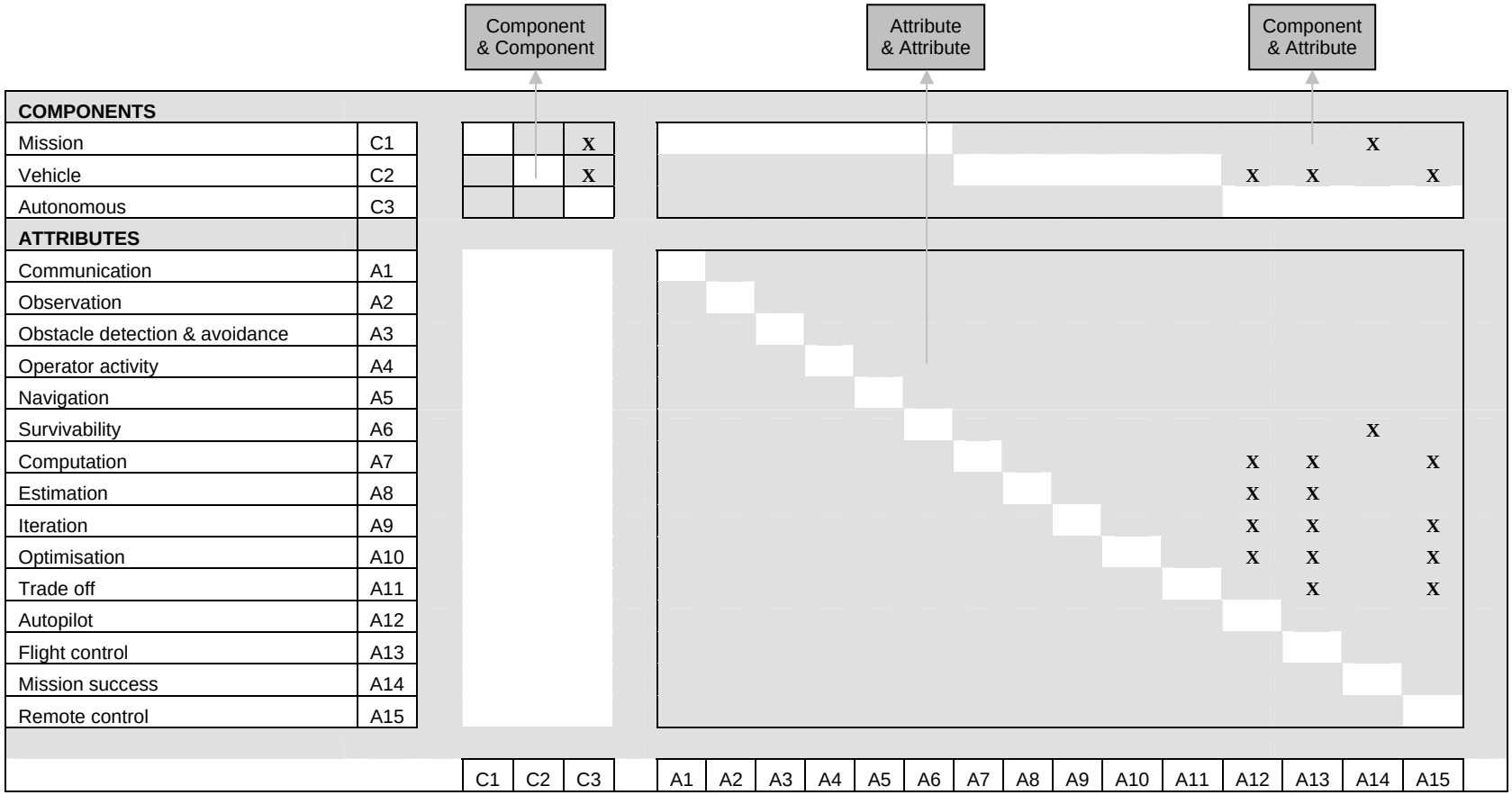
Commercial product	Mission system technology	Grouping (Level-II classification)	Grouping (Level-I classification)
SONAIR	Audio sensitiser	Sensor array	Acoustic sensor
HRN-1	Communication node	Transponder	Communication unit
HRN-2	Communication node	Transponder	Communication unit
TCDL	Communication channel	Transponder	Communication unit
TVL	Video link	Transponder	Communication unit
TVL – II	Video link	Transponder	Communication unit
StarLink	Miniature data link	Transponder	Communication unit
2500-Tx microwave & audio transmitter	Video and audio transmitter	Transmitter	Communication unit
ACN	Connectivity nodule	Transmitter	Communication unit
EL/K-1865	Data terminal	Terminal	Communication unit
VCS700	Data workstation	Terminal	Communication unit
STAR	Data recorder	Terminal	Communication unit
SSMDR	Digital data recorder	Terminal	Communication unit
COMINT	Interceptor node	Interceptor	Communication unit
11ST	Processing system	Imaging processor / Sensor system	Electro-optical unit
AN/ASX-4	EO/IR sensor	Imaging processor / Sensor system	Electro-optical unit
AURORA 600	Acquisition / processing system	Imaging processor / Sensor system	Electro-optical unit
AURORA 600 sensor	Sensor	Imaging processor / Sensor system	Electro-optical unit
BRITE Star	Sensor system	Imaging processor / Sensor system	Electro-optical unit
CYCLOPE 2000	Line-scan sensor	Imaging processor / Sensor system	Electro-optical unit
DSP I	Dual sensor	Imaging processor / Sensor system	Electro-optical unit
MicroSTAR II	Triple sensor	Imaging processor / Sensor system	Electro-optical unit
MOSP	Day/Night sensor	Imaging processor / Sensor system	Electro-optical unit
POP	Optical and electronic sensor	Imaging processor / Sensor system	Electro-optical unit
TOPLITE	Multi sensor	Imaging processor / Sensor system	Electro-optical unit
Uncooled Thermal Imager	Imager	Imaging processor / Sensor system	Electro-optical unit
COMPASS	Observation system	Display & Observation unit	Electro-optical unit
ESP-600C	Day observation system	Display & Observation unit	Electro-optical unit
GOSHAWK SERIES	FLIR sensor	Display & Observation unit	Electro-optical unit
Miniature Pan-Tilt Zoom Daylight Colour TV	Display unit	Display & Observation unit	Electro-optical unit

Commercial product	Mission system technology	Grouping (Level-II classification)	Grouping (Level-I classification)
Type 8042	Reconnaissance sensor	Display & Observation unit	Electro-optical unit
2553 EMC Class A	Zoom camera	Camera & modular accessories	Electro-optical unit
A-CAM 2	Camera system	Camera & modular accessories	Electro-optical unit
Alti-Cam	Camera and turret system	Camera & modular accessories	Electro-optical unit
CA-270	Reconnaissance camera	Camera & modular accessories	Electro-optical unit
CA-295	Reconnaissance camera	Camera & modular accessories	Electro-optical unit
Camelia	Infrared line-scan camera	Camera & modular accessories	Electro-optical unit
CRR-1210-L1	CMOS sensor	Camera & modular accessories	Electro-optical unit
Omega	IR camera	Camera & modular accessories	Electro-optical unit
Pan-Tilt Zoom Camera with laser rangefinder	Surveillance system	Camera & modular accessories	Electro-optical unit
Phoenix	IR camera	Camera & modular accessories	Electro-optical unit
TMX6	Camera module	Camera & modular accessories	Electro-optical unit
ROCLITE	Colour/mono zoom camera	Camera & modular accessories	Electro-optical unit
RollVision - Camera holder	Camera holder	Camera & modular accessories	Electro-optical unit
RollVision - Complete system	Camera system	Camera & modular accessories	Electro-optical unit
AES – 210 V	Electronic warfare payload	Modular payload	Electronic warfare unit
Airborne Communications Jamming Payload	Electronic communication module	Modular payload	Electronic warfare unit
Electronic Surveillance Payload (ESP)	Surveillance node	Modular payload	Electronic warfare unit
Ultralite	Open system	Modular payload	Electronic warfare unit
ALR-93 (V)	Electronic surveillance module	Radar warning receiver / Electronic surveillance module / Signal intelligence component	Electronic warfare unit
CATS	Threat surveyor	Radar warning receiver / Electronic surveillance module / Signal intelligence component	Electronic warfare unit
TOP-SCAN	ELINT/ESM system	Radar warning receiver / Electronic surveillance module / Signal intelligence component	Electronic warfare unit
Ultralight ESM	SIGINT system	Radar warning receiver / Electronic surveillance module / Signal intelligence component	Electronic warfare unit
VICON 78	Dispensing system	Radar warning receiver / Electronic surveillance module / Signal intelligence component	Electronic warfare unit

Commercial product	Mission system technology	Grouping (Level-II classification)	Grouping (Level-I classification)
ALE-50 Towed Decoy System	Radar decoy	Radar decoys / Jammers	Electronic warfare unit
Tactical Radar Jammer	Jammer	Radar decoys / Jammers	Electronic warfare unit
ASELFLIR – 200	Imaging system	Imaging component	Forward looking & thermal imaging unit
FOX-450Z/ZE	Imaging camera	Imaging camera	Forward looking & thermal imaging unit
ISS AeroScout	Multi-sensor	Imaging module	Forward looking & thermal imaging unit
Omega Camera	Infrared micro-sensor	Imaging module	Forward looking & thermal imaging unit
Night Conqueror	Thermal imaging system	Imaging component	Forward looking & thermal imaging unit
SCC500 micro IR camera core	Imaging system	Imaging component	Forward looking & thermal imaging unit
White Star 2751	Surveillance system	On-board UAV component	Integrated unit
2604 Mini DV airborne VCR	On-Board UAV system recorder	On-board UAV component	Integrated unit
ADT	Air-data Terminal	On-board UAV component	Integrated unit
ATX-2740 (v)	Data-link system	On-board UAV component	Integrated unit
BLU-108 Sensor Fused Sub-munition	IR and Laser sensor	On-board UAV component	Integrated unit
GuideStar (GS-111)	Flight control system	On-board UAV component	Integrated unit
Golden Bay	Real time processing system	On-board UAV component	Integrated unit
MP2028	Autopilot unit	On-board UAV component	Integrated unit
UMAS	Multi application on-board system	On-board UAV component	Integrated unit
Montage	Modular target control on-ground unit	Base component	Integrated unit
MRS-2000	Digital packing receiving system	Base component	Integrated unit
TCS	Tactical Control System	Base component	Integrated unit
WePilot1000	Flight control system	Base component	Integrated unit
EL/M-2055	Reconnaissance payload	Radio detection and ranging component	RADAR
LAIRS	Advanced imaging radar system	Radio detection and ranging module	RADAR
MMWR	Wave radar	Radio detection and ranging component	RADAR
Mk 4	Miniature radar altimeter	Radio detection and ranging module	RADAR
Modular AN/APS-144 multimode radar	Multimode radar	Radio detection and ranging component	RADAR
STacSAR	Small tactical synthetic aperture radar	Radio detection and ranging component	RADAR
AN/APY-8	High resolution radar	Radio detection and ranging component	RADAR
Tactical UAV radar	Tactical radar	Radio detection and ranging component	RADAR
TESAR	Surveillance radar	Radio detection and ranging component	RADAR

APPENDIX D

Relationship Matrix identifying Relationships between Components and Attributes of a VTUAV Design System



APPENDIX E

Design specifications of in-development and in-service VTUAVs (Anon, 2004)

VTUAV Platform	Payload	Powerplant (Number & model)	MTOW (lb)	Speed (mph)	Endurance (hr)	Range (miles)	Altitude (ft)	Fuselage length (ft)	Fuselage diameter (ft)	Rotor span (ft)	Wing span (ft)
A-160 HUMMINGBIRD	EO/IR & SAR	1 x 300hp piston engine	4000	161	40	2876	30000	35	12	37	0
Aerosonde	Meteorological	1 x Aerosonde H-Type, 1.2 kw	30	86.4	30	2000	21000	5.6	0.75	1.64	9.5
APID MK5	EW systems & EO/IR System	1 x 125/250cc JP8, 28hp gasoline	220	62	6	290	8000	11.5	2.1	9.8424	0
AUTONOMOUS RMAX	Attitude sensors, & communication modem	1 x 21hp/246cc gasoline	187	55	1.5	80	7000	12.1	2.1	10.35	0
AUTO RMAX TYPE 2	GPS	1 x 21hp/246cc gasoline	187	55	1.5	80	7000	11	2.1	10.35	0
AVX-2X3 D'HUMBUG	Multiple payload	1 x 4-stroke DS-ETC Engine	50	100	2	180	5000	3	0.8	1.5	2
AVX-7X10 D'Gopher	Multiple payload	1 x D-STAR super light turbofan	800	300	4	1100	10000	10	1.5	4	7
Camcopter	Multiple payload	1 x 172cc 22hp, gasoline	165	80.5	6	410	10000	8.25	1.6	10.28	0
CL-327	TV camera & tracker IR	1 x Williams International WTS-117-5, 100shp	770	138	6.25	128	18000	3	6	13.1	0
CL-427	TV camera & tracker IR	1 x Williams International WTS-117-5, 100hp	750	138	7	124	18000	3	7.7	13.1	0
CVG HIGH TECH	SAGEM Cyclope 2000	1 x 26hp piston engine	50	75	5	110	14000	5.5	2.5	6	0
CYPHER II	EO/IR	1 x 50hp class power	220	144	3	220	12000	6	2	8	10
DP-4	Laterally retractable landing skids allow	1 x Quadra-Arow QA200 200cc	140	99.2	1.5	110	4000	10	7.5	16	0
Dragon Warrior	EO/B&W lowlight	1 x 38hp heavy fuel	200	120	5	290	7000	6	1.8	4.2	8.6
Eagle Eye	Radar, EO/IR	1 x Allison 250-C20	3000	253	6	390	20000	16.5	5	10	14.2
EYE_MAV	Multiple payload	1 x 1.3hp D-Star Mini-Diesel engine	10	25	1	15	400	0.2	0.5	0.3	0
Fire Scout (RQ-8A)	EO/IR	1 x RR 250-C20W	2550	144	8	126	20000	22.9	9	27	0
Heliwing	Multiple payload	1 x Williams WTS-117, 130hp	837	171	1	150	12500	8.1	9	14	16.6
HV-911 Eagle Eye	Radar, EO/IR	1 x PWC PW200	2600	253	6	127	20000	17.3	2.8	24.2	15.3
Ka-137	Multiple payload	1 x Hirth 2706 P05, 65 hp	615	110	4	270	11500	17.4	3.9	17.5	0
NITRO HAWK	Thermal device, 35mm camera	1x State-of-art 2-stroke 4shp	22	40	1	25	1500	2.1	1.2	2.5	0
ORKA 1200	Multiple	1x State-of-art engine	1496	120	8	750	18000	14.5	11	15.84	0
Robocopter 300	Agricultural spraying, & offshore oil support	1 x Lycoming HO-360 124kw auto gasoline	1747	110	1	1.15	2000	23	2.15	25	0

VTUAV Platform	Payload	Powerplant (Number & model)	MTOW (lb)	Speed (mph)	Endurance (hr)	Range (miles)	Altitude (ft)	Fuselage length (ft)	Fuselage diameter (ft)	Rotor span (ft)	Wing span (ft)
RPH 2A	2 axis gimbaled camera	1 x 2 stoke 83.5hp	710	62	1	60	6600	17.5	5	11	0
SARD	EO Camera	1 x Bergin Twin gasoline 2-stroke	35	40	0.2	5	2100	4.5	2.5	6	0
Scorpio 30	Flexible payload	1 x State-of-art engine	83	33	2	2	5000	6.67	2.5	7	0
Scorpio 6	Flexible payload	1 x State-of-art engine	30	21	2	3	5000	5.6	2.5	5.4	0
SPRITE	thermal imaging TV	2 x 2 Cycle units, 5kw	79	45	2	65	10000	2	3.125	5.25	0
Vigilante 496	EO/IR, SAR, search & rescue, hyper spectral	1 x Hirth F30 110hp, avgas	1100	108	3	77	9000	20	7	23	0
Vigilante 502	EO/IR, SAR, search & rescue, hyper spectral	1 x Rotax 914 turbo 115hp, avgas	1100	134	7	80.5	12000	20	8	23	0
VTOL UAV	Autonomous navigation	1 x 25hp engine	110	50	4	100	12000	0.5	0.9	0.7	3

APPENDIX F

Database of Power-plants applicable for use on VTUAVs (Anon, 2004)

Model	Manufacturer	Power (hp)	Weight (lb)	Diameter (ft)
100 hp Rotary UEL engine	Elbit Systems Ltd.	100	110	1.45
100N Micro Diesel engine	D-star Engineering, USA	0.98	1	0.26
1KW Micro Diesel engine	D-star Engineering, USA	1.30	1.32	0.23
498ia	Zanzottera Engines, Italy	39	40.92	2.21
52 hp Rotary UEL engine	Elbit Systems Ltd.	52	55	1.09
75W Micro Diesel engine	D-star Engineering, USA	0.10	0.13	0.13
88-H	Bernard Hooper Engineering Ltd.	7	5.17	0.81
A150B	Aerrow Inc., Canada	17	8.25	0.65
A200B	Aerrow Inc., Canada	21	11	0.65
A200RSS	Aerrow Inc., Canada	27	10.40	0.65
A2600	Alturdyne, USA	156	1100	3.83
A650	Alturdyne, USA	60	650	1.33
AR682	UAV Engine Ltd., UK	75	112.20	1.44
AR682R	UAV Engine Ltd., UK	120	124.30	1.61
AR731	UAV Engine Ltd., UK	38	21.70	0.86
AR741	UAV Engine Ltd., UK	38	23.50	0.86
AR801	UAV Engine Ltd., UK	60	43	0.82
AR801R	UAV Engine Ltd., UK	51	56	1.12
B1000ia	Zanzottera Engines, Italy	72	64.90	2.21
B2000i	Zanzottera Engines, Italy	130	98	2.21
B2000ir	Zanzottera Engines, ITALY	170	112.20	2.21
BS 151	Bet Shemesh Engines Ltd.	297	66	0.90
BS 175	Bet Shemesh Engines Ltd.	330	55.50	0.90
CLASS-A	Lightning Aircraft Engines, USA	25	14	1.08
CLASS-B	Lightning Aircraft Engines, USA	30	14.75	1.17
Diesel DH160V41	Delta Hawk Inc., USA	160	327	1.72
Diesel DH200V41	Delta Hawk Inc., USA	200	327	1.72
ENYA 60-4C	Enya Inc.	1	2.20	0.10
H 30 ES	Göbler-Hirthmotoren KG	102	92.70	1.67
H 32 E	Göbler-Hirthmotoren KG	65	68.40	1.95
H 37 E	Göbler-Hirthmotoren KG	90.60	99.30	1.65
L 2400 DFi/EFi	Limbach Flugmotoren, GmbH & Co. KG, Germany	100	167.20	2.62
L 2400 DT/ET	Limbach Flugmotoren, GmbH & Co. KG, Germany	130	189.20	2.64
L 2400 DTX	Limbach Flugmotoren, GmbH & Co. KG, Germany	160	189.20	2.64
L1700 EA	Limbach Flugmotoren, GmbH & Co. KG, Germany	60	151.80	2.58
L1700E0/EC	Limbach Flugmotoren, GmbH & Co. KG, Germany	68	162.80	2.56
L2000 EA	Limbach Flugmotoren, GmbH & Co. KG, Germany	80	160.60	2.58
L2000 EB	Limbach Flugmotoren, GmbH & Co. KG, Germany	80	168.30	2.63
L2000E0/EC	Limbach Flugmotoren, GmbH & Co. KG, Germany	80	160.60	2.56
L2400 EB	Limbach Flugmotoren, GmbH & Co. KG, Germany	87	180.40	2.63
L275 E	Limbach Flugmotoren, GmbH & Germany	20	15.840	2.67
L550 E	Limbach Flugmotoren, GmbH & Co. KG, Germany	50	35.20	2.36
Lycoming IO-360 A series	Honeywell, USA	200	326	1.95
Mazda 13-B	Alturdyne, USA	78	800	2.17

Model	Manufacturer	Power (hp)	Weight (lb)	Diameter (ft)
Mazda 20-B	Alturdyne, USA	117	950	3
Model 250 – A	Rolls-Royce, USA	400	274	1.83
Model 250 – B	Rolls-Royce, USA	425	274	1.83
Model 250 – C	Rolls-Royce, USA	450	274	1.83
Model 250 – D	Rolls-Royce, USA	475	274	1.83
Model 250 – E	Rolls-Royce, USA	500	274	1.83
Model 250 – F	Rolls-Royce, USA	525	274	1.83
Model 250 – G	Rolls-Royce, USA	550	274	1.83
Model 250 – H	Rolls-Royce, USA	575	274	1.83
Model 250 – I	Rolls-Royce, USA	600	274	1.83
Model 250 – J	Rolls-Royce, USA	625	274	1.83
Model 250 – L	Rolls-Royce, USA	675	274	1.83
Model 250 – M	Rolls-Royce, USA	700	274	1.83
Model 250 – N	Rolls-Royce, USA	725	274	1.83
Model 250 – O	Rolls-Royce, USA	750	274	1.83
Model 250-C3OR/3	Rolls-Royce, USA	650	274	1.83
Q1000B	Aerrow Inc., Canada	9.40	6	0.51
Q100B	Aerrow Inc., Canada	12	6.60	0.75
Q100M	Aerrow Inc., Canada	12	8.30	0.76
Q100RSS	Aerrow Inc., Canada	16	6.60	0.75
Q250BXR	Aerrow Inc., Canada	2.80	2.65	0.57
Q400B	Aerrow Inc., Canada	4	3.73	0.50
Q400M	Aerrow Inc., Canada	4	4.53	0.50
Q52B	Aerrow Inc., Canada	4.50	4.10	0.54
Q52M	Aerrow Inc., Canada	4.50	5.10	0.54
Q65M	Aerrow Inc., Canada	6.80	6.50	0.72
Q75B	Aerrow Inc., Canada	10	5.20	0.72
Q75M	Aerrow Inc., Canada	10	6.50	0.72
Q75RSS	Aerrow Inc., Canada	11	5.10	0.72
Rotapower engine-A	Freedom Motors	150	170.50	1.43
Rotapower engine-B	Freedom Motors	75	1250	1.89
Rotary UEL engine / Hermes 180	Elbit Systems Ltd.	38	21	0.85
SOREK 4	Bet Shemesh Engines Ltd.	1442	159	1.09
SPV580	Bernard Hooper Engineering Ltd.	47.50	40.70	1.20

APPENDIX G

Database of UAVs - Maximum Take-Off Weight (MTOW), Empty Weight & Payload
(Anon, 2004)

- *Micro UAVs*

UAV Platform	MTOW (Kg)	Empty weight (Kg)	Payload (Kg)
Aladin	3.00	2.40	0.60
POINTER	4.55	3.64	0.91
PUMA	5.44	4.44	1.00
BUSTER UAV	5.45	4.08	1.37
BACKPACK UAV	5.45	4.09	1.37
GRASS HOPPER	8.20	6.00	2.20

- *Small UAVs*

UAV Platform	MTOW (Kg)	Empty weight (Kg)	Payload (Kg)
AZIMUT 2	9.00	7.00	2.00
JAVELIN	9.10	7.65	1.45
Aerosonde	15.00	10.00	5.00
SCANEAGLE A	15.40	11.10	4.30
SEASCAN	15.40	11.20	4.20
PCUAV	18.00	13.00	5.00
ALO	20.00	14.00	6.00
CVG HIGH TECH	22.73	11.36	11.37
AVX-2X3	22.80	18.25	4.55
MERCURY	30.00	19.00	11.00
AEC1000	31.00	24.00	7.00
D-1	35.00	25.00	10.00
SPRITE	36.00	30.00	6.00
Aerolight	40.00	32.00	8.00
MUKHBAR	40.00	35.00	5.00
PARWAZ	40.00	35.00	5.00
OBSERVER	45.00	39.00	6.00
SPARROW	45.00	33.00	12.00
NEPTUNE	52.30	43.20	9.10
ANSER BRUMBY MK4	60.00	42.00	18.00
GOLDEN EYE	69.00	48.50	20.50
Aerosky	70.00	52.00	18.00
CHACAL	75.00	55.00	20.00
RAVEN	84.00	66.80	17.20
RMAX 2	90.00	60.00	30.00
RPG MIDGET MK 3	90.00	75.00	15.00
APID MK5	100.00	63.00	37.00

- *Large UAVs*

UAV Platform	MTOW (Kg)	Empty weight (Kg)	Payload (Kg)
DRAGON WARRIOR	113.00	97.00	16.00
SENTRY	113.7	84.15	29.55
KESTREL 2	120.00	90.00	30.00
JASOOS 2	125.00	95.00	30.00
NASNAS MK1	125.00	100.00	25.00
PCHELA-IT	138.00	68.00	70.00
DRAGONFLY 2000	140.00	124.00	16.00
SOJKA	145.00	120.00	25.00
SPECTRE 2	145.60	109.23	36.37
SHADOW 200	149.10	121.80	27.30
BLUE HORIZON 2	150.00	113.00	37.00
CRECERELLE	150.00	115.00	35.00
VULTURE	150.00	115.00	35.00
HELLFOX	160.00	100.00	60.00
KZO	161.00	126.00	35.00
DAKOTA	180.00	140.00	40.00
NB-X2	180.00	130.00	50.00
ISIS	194.00	160.00	34.00
HERMES 180	195.00	163.00	32.00
Aerostar	200.00	150.00	50.00
VECTOR MK 1	200.00	175.00	25.00
PIONEER	203.00	158.00	45.00
SHADOW 400	203.00	173.00	30.00
PHOENIX	210.00	158.00	52.00
ASN 206	222.00	172.00	50.00
CL-289	240.00	210.00	30.00
TAI UAV-X1	245.00	200.00	45.00
SHADOW 600	266.00	220.50	45.50
RANGER	275.00	230.00	45.00
SEAKER 2	275.00	225.00	50.00
KA-137	280.00	200.00	80.00
NIGHT INTRUDER 300	290.00	245.00	45.00
SIVA	290.00	241.00	49.00
RPH 2A	330.00	230.00	100.00
PROWLER 2	340.00	294.70	45.30
FALCO	340.00	210.00	130.00
PATHFINDER PLUS	340.00	295.00	45.00
CL-327	350.00	250.00	100.00
NIBBIO	350.00	220.00	130.00
NISHANT	350.00	290.00	60.00
SKYEYE	354.60	293.20	61.40
VINDICATOR	363.63	272.73	90.90
AVX-7X10 D'GPOHER	364.00	227.63	136.37
SEARCHER MK 2	426.00	326.00	100.00
I-GNAT	703.00	612.00	91.00
HUNTER (RQ-5A)	727.00	613.00	114.00
HELIOS	750.00	610.00	140.00
X-50A	810.00	719.00	91.00
E-HUNTER	954.00	840.00	114.00
ALTUS	975.00	825.00	150.00
PREDATOR A	1070.00	866.00	204.00
PERSEUS B	1100.00	940.00	160.00
HERON	1150.00	900.00	250.00
FIRE SCOUT	1160.00	1069.00	91.00
EAGLE 1	1200.00	950.00	250.00

APPENDIX H

H.1 'Design-input' Class

The program code of the class validates the acquired design requirements and constraints as follows:

```
// File name: InputDlg.cpp
// Inclusion of header files
#include "stdafx.h"
#include "VTUAV_Expert_System.h"
#include "InputDlg.h"
#include "WelcomeDlg.h"
#include "MRDlg.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

// Declaration & initialisation of class variables
CInputDlg::CInputDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CInputDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CInputDlg)
	m_input1 = 0.0;
	m_input2 = 0.0;
	m_input3 = 0.0;
	m_input4 = 0.0;
	m_input5 = 0.0;
	m_input6 = 0.0;
	m_input7 = 0.0;
	m_input8 = 0.0;
	m_input9 = 0.0;
	m_input9a = 0.0;
	m_cf = 0;
	m_time = 1;
	m_yn = 1;
	m_input11 = 0.0;
	//}}AFX_DATA_INIT
}

// Limiting variables values
Void CInputDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CInputDlg)
	DDX_Text(pDX, IDC_EDIT1, m_input1);
	DDV_MinMaxDouble(pDX, m_input1, 4.9, 9586.);
	DDX_Text(pDX, IDC_EDIT2, m_input2);
	DDV_MinMaxDouble(pDX, m_input2, 1.7, 6414.);
	DDX_Text(pDX, IDC_EDIT3, m_input3);
	DDV_MinMaxDouble(pDX, m_input3, 6.6, 16000.);
	DDX_Text(pDX, IDC_EDIT4, m_input4);
	DDV_MinMaxDouble(pDX, m_input4, 21., 253.);
	DDX_Text(pDX, IDC_EDIT5, m_input5);
	DDV_MinMaxDouble(pDX, m_input5, 6.e-002, 40.);
	DDX_Text(pDX, IDC_EDIT6, m_input6);
	DDV_MinMaxDouble(pDX, m_input6, 1.15, 6762.);
	DDX_Text(pDX, IDC_EDIT7, m_input7);
	DDV_MinMaxDouble(pDX, m_input7, 400., 330000.);
```

```
	DDX_Text(pDX, IDC_EDIT8, m_input8);
	DDV_MinMaxDouble(pDX, m_input8, 0.5, 115.);
	DDX_Text(pDX, IDC_EDIT9, m_input9);
	DDV_MinMaxDouble(pDX, m_input9, 0.5, 247.);
	DDX_Text(pDX, IDC_EDIT9a, m_input9a);
	DDV_MinMaxDouble(pDX, m_input9a, 0.2, 34.);
	DDX_Radio(pDX, IDC_RADIO1, m_cf);
	DDX_Radio(pDX, IDC_RADIO5, m_time);
	DDX_Radio(pDX, IDC_RADIO3, m_yn);
	DDX_Text(pDX, IDC_EDIT11, m_input11);
	DDX_Control(pDX, IDC_LABEL34, m_label);
	//}}AFX_DATA_MAP
}

// Defining user actions vis-a-vis system messages
BEGIN_MESSAGE_MAP (CInputDlg, CDialog)
	//{{AFX_MSG_MAP(CInputDlg)
	ON_BN_CLICKED (IDIH, OnIh)
	ON_BN_CLICKED (IDIN, OnIn)
	ON_BN_CLICKED (IDIOFC, OnIofc)
	ON_BN_CLICKED (IDIMI, OnImi)
	ON_EN_CHANGE (IDC_EDIT2, OnChangeEdit2)
	ON_EN_CHANGE (IDC_EDIT4, OnChangeEdit4)
	ON_EN_CHANGE (IDC_EDIT1, OnChangeEdit1)
	ON_BN_CLICKED (IDC_RADIO3, OnRadio3)
	ON_BN_CLICKED (IDC_RADIO4, OnRadio4)
	ON_BN_CLICKED (IDMPE, OnMpe)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP ()

// Class method – Dialog initialisation
BOOL CInputDlg::OnInitDialog()
{

	CDialog::OnInitDialog();
	Extern double ewt; //Empty weight
	Extern double pwt; //Payload weight
	Extern double lwt; //MTOW or launch weight
	Extern double sp; //Forward speed
	Extern double en; //Endurance
	Extern double ra; //Range
	Extern double al; //Altitude
	Extern double le; //Fuselage length
	Extern double sn; //Fuselage span or width
	Extern double di; //Rotor diameter
	Extern double sr; //Safety regulations
	Extern double aw; //Stub wing or additional wing
	Extern double fr; //Fuel reserve weight
	Extern double wsn; //Wing span

	// Retrieve values during back navigation
	If ((ewt > 0))
	{
		m_input1=ewt;
		m_input2=pwt;
		m_input3=lwt;
		m_input4=sp;
		m_input5=en;
		m_input6=ra;
		m_input7=al;
		m_input8=le;
		m_input9=sn;
		m_input9a=di;
```

```
                m_input11=wsn;
                m_cf=int(sr);
                m_yn=int(aw);
                m_time=int(fr);
                UpdateData(FALSE);
        }
        If (m_yn==0)
        {
                CWnd* a=(CWnd*)GetDlgItem(IDC_EDIT11);
                a->ShowWindow(SW_SHOW);
        }
        return TRUE;
}

// Class method – Dialog display
Void CInputDlg::OnIh ()
{
        CDialog::EndDialog(IDOK);
        CWelcomeDlg a;
        a.DoModal();
}

// Class method – Input validation
Void CInputDlg::OnIn()
{
        // Recalling global (public) variable declarations
        Extern double ewt; //Empty weight
        Extern double pwt; //Payload weight
        Extern double lwt; //MTOW or launch weight
        Extern double ewt1; //Interim empty weight
        Extern double pwt1; //Interim payload weight
        Extern double lwt1;  //Interim MTOW or launch weight
        Extern double sp; //Forward speed
        Extern double en; //Endurance
        Extern double ra; //Range
        Extern double al; //Altitude
        Extern double le; //Fuselage length
        Extern double sn; //Fuselage span or width
        Extern double di; //Rotor diameter
        Extern double sr; //Safety regulations
        Extern double aw; //Stub wing or additional wing
        Extern double fr; //Fuel reserve weight
        Extern double wsn; //Wing span
        Extern double uavprojectedarea; // Projected area of the fuselage
        UpdateData(TRUE);


        bool proceed=FALSE, proceed1=FALSE;
        double var;
        CString buffer;

        // Validation of design parameters
        if((m_input1 >= 4.9) && (m_input1 <= 9586) && (m_input2 >= 1.7) && (m_input2 <= 6414) &&
        (m_input3 >= 6.6) && (m_input3 <= 16000) && (m_input4 >= 21) && (m_input4 <= 253) &&
        (m_input5 >= 0.06) && (m_input5 <= 40) && (m_input6 >= 1.15) && (m_input6 <= 6762) &&
        (m_input7 >= 400) && (m_input7 <= 330000) && (m_input8 >= 0.5) && (m_input8 <= 115) &&
        (m_input9 >= 0.5) && (m_input9 <= 247) && (m_input9a >= 0.2) && (m_input9a <= 34) )
        {
                if((m_input3 / 2.2) <= 8)
                {
                        var=1.064 * ((0.0177 * (m_input3 / 2.2) * (m_input3 / 2.2)) + (0.1132 * (m_input3 /
                        2.2)) + 0.0854);
                        var=var*2.2;
```

```
                    if (var >= m_input2)
                    {
                            proceed =TRUE;
                    }
                    else
                    {
                            buffer.Format("%lf",var);
                            if(MessageBox("The maximum payload weight for MTOW is "
                            +buffer+ " lb. Please reduce the payload or alternatively increase
                            the empty weight of the aircraft. Do you want to view the involved
                            formulas? ","Unexpected Payload Values" , MB_YESNO |
                            MB_ICONEXCLAMATION  | MB_DEFBUTTON1 ) == IDYES)
                            {
                                    MessageBox("Micro UAV: Payload-
                                    max=1.064(0.0177(MTOW)^2 + 0.1132(MTOW) +
                                    0.0854); Small UAV: Payload-
                                    max=1.235(0.0025(MTOW)^2 + 0.0184(MTOW) +
                                    3.6847); Large UAV: Payload-max=1.393(1E-
                                    05(MTOW)^2 + 0.1264(MTOW) + 22.25)","Launch Wt.,
                                    Empty Wt. and Payload Wt. co-relation" , MB_OK |
                                    MB_ICONQUESTION  |MB_DEFBUTTON1 );
                            }
                    }
}
else if(((m_input3 / 2.2) > 8) && ((m_input3 / 2.2) <= 100))
{
        var=1.235 * ((0.0025 * (m_input3 / 2.2) * (m_input3 / 2.2)) + (0.0184 * (m_input3 /
        2.2)) + 3.6847);
        var=var*2.2;
        if (var >= m_input2)
        {
                proceed=TRUE;
        }
        else
        {
                buffer.Format("%lf",var);
                if(MessageBox("The maximum payload weight for MTOW is " +buffer+ "
                lb. Please reduce the payload or alternatively increase the empty weight of
                the aircraft. Do you want to view the involved formulas? ","Unexpected
                Payload Values", MB_YESNO | MB_ICONEXCLAMATION |
                MB_DEFBUTTON1) == IDYES)
                {
                        MessageBox("Micro UAV: Payload-max=1.064(0.0177(MTOW)^2
                        + 0.1132(MTOW) + 0.0854); Small UAV: Payload-
                        max=1.235(0.0025(MTOW)^2 + 0.0184(MTOW) + 3.6847); Large
                        UAV: Payload-max=1.393(1E-05(MTOW)^2 + 0.1264(MTOW) +
                        22.25)","Launch Wt., Empty Wt. and Payload Wt. co-relation" ,
                        MB_OK | MB_ICONQUESTION  |MB_DEFBUTTON1 );
                }
          }
}
else
{
        var=1.393 * ((1E-05 * (m_input3 / 2.2) * (m_input3 / 2.2)) + (0.1264 * (m_input3 /
        2.2)) + 22.25);
        var=var*2.2;
        if (var >= m_input2)
        {
                proceed=TRUE;
        }
        else
        {
                buffer.Format("%lf",var);
```

```
                        if(MessageBox("The maximum payload weight for MTOW is " +buffer+ "
                        lb. Please reduce the payload or alternatively increase the empty weight of
                        the aircraft. Do you want to view the involved formulas? ","Unexpected
                        Payload Values" , MB_YESNO | MB_ICONEXCLAMATION  |
                        MB_DEFBUTTON1 ) == IDYES)
                        {
                                MessageBox("Micro UAV: Payload-max=1.064(0.0177(MTOW)^2
                                + 0.1132(MTOW) + 0.0854); Small UAV: Payload-
                                max=1.235(0.0025(MTOW)^2 + 0.0184(MTOW) + 3.6847); Large
                                UAV: Payload-max=1.393(1E-05(MTOW)^2 + 0.1264(MTOW) +
                                22.25)","Launch Wt., Empty Wt. and Payload Wt. co-relation" ,
                                MB_OK | MB_ICONQUESTION  |MB_DEFBUTTON1 );
                        }
                }
        }
        if(((m_input4*m_input5/m_input6) >= 1) && ((m_input4*m_input5/m_input6) <= 4))
        {
                proceed1=TRUE;
        }
        else
        {
                MessageBox("Speed, endurance and range values are not as expected. The lower limit
                for (Speed*Endurance/Range) value is 1 and the upper limit for
                (Speed*Endurance/Range) value is 4. Please adjust the values
                accordingly","Unexpected Speed, Range and Endurance Co-relation" , MB_OK |
                MB_ICONEXCLAMATION  | MB_DEFBUTTON1 );
        }
}

// Save the validated design parameters
if(proceed && proceed1 && (m_input1 >= 4.9) && (m_input1 <= 9586) && (m_input2 >= 1.7) &&
(m_input2 <= 6414) && (m_input3 >= 6.6) && (m_input3 <= 16000) && (m_input4 >= 21) &&
(m_input4 <= 253) && (m_input5 >= 0.06) && (m_input5 <= 40) && (m_input6 >= 1.15) &&
(m_input6 <= 6762) && (m_input7 >= 400) && (m_input7 <= 330000) && (m_input8 >= 0.5) &&
(m_input8 <= 115) && (m_input9 >= 0.5) && (m_input9 <= 247) && (m_input9a >= 0.2) &&
(m_input9a <= 34) )
{
        ewt=m_input1;
        pwt=m_input2;
        wt=m_input3;
        sp=m_input4;
        en=m_input5;
        ra=m_input6;
        al=m_input7;
        le=m_input8;
        sn=m_input9;
        di=m_input9a;
        sr=m_cf;
        aw=m_yn;
        fr=m_time;
        wsn=m_input11;

        // Conversion to SI system – Pound to kg
        ewt1=ewt/2.2;
        pwt1=pwt/2.2;
        lwt1=lwt/2.2;
        CString sewt1,spwt1,slwt1;
        sewt1.Format("%lf", ewt1);
        spwt1.Format("%lf", pwt1);
        slwt1.Format("%lf", lwt1);

        // CASA Safety Regulation
        if(m_cf==0)
```

```
{
	if(lwt1<=(0.1))
	{
		MessageBox("1. Since the MTOW of the UAV is less than 100 grams, the
		aircraft should be classified as a MICRO UAV. 2. Similar to remotely
		controlled model aircrafts, micro UAVs are exempt from most of regulatory
		requirements (registration and marking, use of radio and certification)
		applying to larger UAVs but they are subject to rules governing hazardous
		operation, flight in controlled airspace and operations exceeding 400ft AGL.
		3. A need for a telephony device is strongly stated. Please ensure that you
		later select it from payload classification section. 4. Finally, all the local state
		laws are applicable. Please obtain a copy of them from the respective offices
		before the design is put into practical implementation","CASA
		REGULATION CLASSIFICATION - MICRO UAV", MB_OK |
		MB_ICONINFORMATION  |MB_DEFBUTTON1);
	}
	else if((lwt1>0.1) && (lwt1<=100))
	{
		if(lwt1>0.1 && lwt1<=8)
		{
			MessageBox("CASA classifies micro UAV aircraft to be of
			maximum 100 gms which is inconsistent with the world standard of
			upto 8kg MTOW","SMALL UAV CLASSIFICATION", MB_OK |
			MB_ICONINFORMATION  |MB_DEFBUTTON1);
		}
		MessageBox("1. Since the MTOW of the UAV is more than 100 grams and
		less than 100 kg, the aircraft should be classified as a SMALL UAV. 2. They
		may only fly outside an approved area if they are kept clear of populous
		areas and controlled airspace. If flying above 400ft AGL, the operation must
		have CASA approval. 3. A need for a telephony device is strongly stated.
		Please ensure that you later select it from payload classification section. 4.
		Finally, all the local state laws are applicable. Please obtain a copy of them
		from the respective offices before the design is put into practical
		implementation","CASA REGULATION CLASSIFICATION - SMALL
		UAV", MB_OK | MB_ICONINFORMATION  |MB_DEFBUTTON1);
	}
	else
	{
		MessageBox("1. Since the MTOW of the UAV is more than 100 kg, the
		aircraft should be classified as a LARGE UAV. Usually the weight
		classification for large UAVs starts at 150 Kg but in case of rotorcraft UAVs
		it starts at 100 kg. 2. Large UAVs are effectively treated as conventional
		aircraft for registration purposes and are operated under a special certificate
		of airworthiness (restricted category) or an experimental certificate.  3.
		Additional regulatory requirements include a UAV operator certificate
		(similar to an AOC), controller certificate (similar to a pilot licence) and
		rules for maintenance, use of radiotelephone and ATC procedures. There are
		also operational and equipment specifications. 4. Finally, all the local state
		laws are applicable. Please obtain a copy of them from the respective offices
		before the design is put into practical implementation.","CASA
		REGULATION CLASSIFICATION - LARGE UAV", MB_OK |
		MB_ICONINFORMATION  |MB_DEFBUTTON1);
	}
}

// Other Safety Standards
if(m_cf==1)
{
	if(lwt1<=8)
	{
		MessageBox("1. Since the MTOW of the UAV is less than 8 kg, the aircraft
		should be classified as a MICRO UAV. 2.Like remotely controlled model
		aircraft, micro UAVs aircraft are exempt from most of the regulatory
```

```
                requirements (registration and marking, use of radio and certification)
                applying to larger UAVs. But they are subject to rules governing hazardous
                operation, flight in controlled airspace and operations exceeding 400ft AGL.
                3. A need for a telephony device is strongly stated. Please ensure that you
                later select it from payload classification section. 4. Finally, all the local state
                laws are applicable. Please obtain a copy of them from the respective offices
                before the design is put into practical implementation","WORLD SAFETY
                STANDARDS - MICRO UAV", MB_OK | MB_ICONINFORMATION
                |MB_DEFBUTTON1);
        }
        else if(lwt1>8 && lwt1<=100)
        {
                MessageBox("1. Since the MTOW of the UAV is more than 8 kg and less
                than 100 kg, the aircraft should be classified as a SMALL UAV. 2. They
                may only fly outside an approved area if they are kept clear of populous
                areas and controlled airspace. If flying above 400ft AGL, the operation must
                have local authorities approval. 3. A need for a telephony device is strongly
                stated. Please ensure that you later select it from payload classification
                section. 4. Finally, all the local state laws are applicable. Please obtain a
                copy of them from the respective offices before the design is put into
                practical implementation","WORLD SAFETY STANDARDS - SMALL
                UAV", MB_OK | MB_ICONINFORMATION  |MB_DEFBUTTON1);
        }
        else
        {
                MessageBox("1. Since the MTOW of the UAV is more than 100 kg, the
                aircraft should be classified as a LARGE UAV. Usually the weight
                classification for large UAVs starts at 150 Kg but in case of rotorcraft UAVs
                it starts at 100 kg. 2. Large UAVs are effectively treated as conventional
                aircraft for registration purposes and are operated under a special certificate
                of airworthiness (restricted category) or an experimental certificate.  3.
                Additional regulatory requirements include a UAV operator certificate
                (similar to an AOC), controller certificate (similar to a pilot licence) and
                rules for maintenance, use of radiotelephone and ATC procedures. There are
                also operational and equipment specifications. 4. Finally, all the local state
                laws are applicable. Please obtain a copy of them from the respective offices
                before the design is put into practical implementation.","WORLD SAFETY
                STANDARDS - LARGE UAV", MB_OK | MB_ICONINFORMATION
                |MB_DEFBUTTON1);
        }
}
MessageBox("The SI value for the empty weight is "+sewt1+" kg. The SI value for the
payload weight is "+spwt1+" kg. The SI value for the launch weight is "+slwt1+" kg","SI
UNITS CONVERSION" , MB_OK | MB_ICONEXCLAMATION  | MB_DEFBUTTON1);

// Calculation of the projected area
uavprojectedarea=0.0;
if (aw == 0)
{
        // Stub wings required
        uavprojectedarea = 0.5*wsn*le;
}
else
{
        // Stub wing not required
        uavprojectedarea = (2*0.5*3.1416*(le/6)*(le/6)) + (le*sn);
}
if(m_yn ==0)
{
        if(wsn>0)
        {
                CDialog::EndDialog(IDOK);
                CMRDlg a;
```

```
                        a.DoModal();
                }
                else
                {
                        MessageBox("Please enter Wing Span value","Incorrect Values");
                }
        }
        else
        {
        CDialog::EndDialog(IDOK);
        CMRDlg a;
        a.DoModal();
        }
    }
}
```

// Class method – Constraints display

```
void CInputDlg::OnIofc()
{
    MessageBox("A> Disc loading at minimum end of scale, valued at 4.0 lb/ft*ft  B> Drag coefficient =
    0.3  c> Number of blades = 4  d> Coefficient of thrust to solidity = 0.17  e> Blade aspect ratio = 12.0 f>
    Initial estimate of blade chord = 0.0127m.  g> Tip speed ratio = 0.5","Fixed Design Constraints" ,
    MB_OK | MB_ICONINFORMATION  |MB_DEFBUTTON1);
}
```

// Class method –Safety regulation display

```
void CInputDlg::OnImi()
{
    MessageBox(" 1. UAVs to be classified as either micro(less than 100 grams), small (between 100g and
    100 kg) or large(more than 100kg) with separate rules applicable to each category. 2. Micro UAVs are
    largely exempt from regulation while a small UAV may be flown by an unqualified person in certain
    conditions without any form of certification. A large UAV is a de-facto manned aircraft and must be
    certificated and registered and its controllers qualified. 3. There are also requirements on the use of a
    flight radiotelephone; 4. Height restrictions for unmanned moored balloons, kites, model aircraft and
    rockets to be raised from 300ft Above Ground Level (AGL) to 400ft AGL in line with overseas
    practice. 5. Requirements under relevant state and territory laws will continue to apply." ,"CASR Part
    101 - Unmanned aircraft and rocket operations", MB_OK | MB_ICONFIRMATION
    |MB_DEFBUTTON1);

    MessageBox(" 1. UAVs to be classified as either micro(less than 8 kg), small (between 8kg and 100kg)
    or large(more than 100kg) with separate rules applicable to each category. 2. Micro UAVs are largely
    exempt from regulation while a small UAV may be flown by an unqualified person in certain
    conditions without any form of certification. A large UAV is a de-facto manned aircraft and must
    generally be certificated and registered and its controllers qualified. 3. There are also requirements on
    the use of Position Lights, Anti-Collision Lights, Transponder, Radios, Navigation Systems, UAV
    System and Altitude Displays, Flight and Voice Recorder, and Built-in Test mechanisms. 4. Height
    restrictions for model aircraft are set at 400ft AGL 5. Requirements under relevant state and territory
    laws will continue to apply." ,"World Safety Standards - Unmanned aircrafts", MB_OK |
    MB_ICONINFORMATION  |MB_DEFBUTTON1);

}
```

// Class method – String to decimal conversion

```
void CInputDlg::OnChangeEdit2()
{
    CEdit* a=(CEdit*)GetDlgItem(IDC_EDIT1);
    CEdit* b=(CEdit*)GetDlgItem(IDC_EDIT2);
    CEdit* c=(CEdit*)GetDlgItem(IDC_EDIT3);
    CString x, y;
    a->GetWindowText(x);
    b->GetWindowText(y);
    char  *stopstring;
    double z, z1;
```

```
	z = strtod( x, &stopstring);
	z1 = strtod( y, &stopstring);
	z=z+z1;
	y.Format("%lf",z);
	c->SetWindowText(y);
}

// Class method – String to decimal conversion
void CInputDlg::OnChangeEdit4()
{
	CEdit* a=(CEdit*)GetDlgItem(IDC_EDIT1);
	CEdit* b=(CEdit*)GetDlgItem(IDC_EDIT2);
	CString x, y;
	a->GetWindowText(x);
	b->GetWindowText(y);
	char  *stopstring;
	double z, z1;
	 z = strtod( x, &stopstring);
	z1 = strtod( y, &stopstring);

	if(z<z1)
	{
		MessageBox("Payload weight cannot be more than the empty weight","Please Enter Values
		Carefully");
		a->SetFocus();
	}
}

// Class method – String to decimal conversion
void CInputDlg::OnChangeEdit1()
{
	CEdit* a=(CEdit*)GetDlgItem(IDC_EDIT1);
	CEdit* b=(CEdit*)GetDlgItem(IDC_EDIT2);
	CEdit* c=(CEdit*)GetDlgItem(IDC_EDIT3);
	CString x, y;
	a->GetWindowText(x);
	b->GetWindowText(y);
	char  *stopstring;
	double z, z1;
	z = strtod( x, &stopstring);
	z1 = strtod( y, &stopstring);
	z=z+z1;
	y.Format("%lf",z);
	c->SetWindowText(y);
}

// Class method – Show other dialog
void CInputDlg::OnRadio3()
{
	CWnd* a=(CWnd*)GetDlgItem(IDC_EDIT11);
	a->ShowWindow(SW_SHOW);
}

// Class method – Show other dialog
void CInputDlg::OnRadio4()
{
	CWnd* a=(CWnd*)GetDlgItem(IDC_EDIT11);
	a->ShowWindow(SW_HIDE);
}
```

H.2 ‘Mission time’ Class

The ‘Mission-time’ class estimates and validates the time required to complete the slated mission. The program code of the ‘Mission-time’ class based on the reconnaissance and surveillance mission is as follows:

```
// File name: MR1A1Dlg.cpp
// Header Files
#include "stdafx.h"
#include "VTUAV_Expert_System.h"
#include "MR1A1Dlg.h"
#include "MR1ADlg.h"
#include "UavdesignfcsDlg.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

// Declaration & initialisation of class variables
CMR1A1Dlg::CMR1A1Dlg(CWnd* pParent /*=NULL*/)
	: CDialog(CMR1A1Dlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CMR1A1Dlg)
	m_1 = 0;
	m_10 = 0;
	m_11 = 0;
	m_12 = 0;
	m_13 = 0;
	m_14 = 0;
	m_15 = 0;
	m_2 = 0;
	m_3 = 0;
	m_4 = 0;
	m_5 = 0;
	m_6 = 0;
	m_7 = 0;
	m_8 = 0;
	m_9 = 0;
	m_16a = 0.0;
	//}}AFX_DATA_INIT
}

// Limiting variables values
void CMR1A1Dlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CMR1A1Dlg)
	DDX_Control(pDX, IDC_EDIT16a, m_16c);
	DDX_Text(pDX, IDC_EDIT1, m_1);
	DDV_MinMaxInt(pDX, m_1, 0, 60);
	DDX_Text(pDX, IDC_EDIT10, m_10);
	DDV_MinMaxInt(pDX, m_10, 0, 60);
	DDX_Text(pDX, IDC_EDIT11, m_11);
	DDV_MinMaxInt(pDX, m_11, 0, 60);
	DDX_Text(pDX, IDC_EDIT12, m_12);
	DDV_MinMaxInt(pDX, m_12, 0, 60);
	DDX_Text(pDX, IDC_EDIT13, m_13);
	DDV_MinMaxInt(pDX, m_13, 0, 60);
	DDX_Text(pDX, IDC_EDIT14, m_14);
	DDV_MinMaxInt(pDX, m_14, 0, 60);
```

```
	DDX_Text(pDX, IDC_EDIT15, m_15);
	DDV_MinMaxInt(pDX, m_15, 0, 60);
	DDX_Text(pDX, IDC_EDIT2, m_2);
	DDV_MinMaxInt(pDX, m_2, 0, 60);
	DDX_Text(pDX, IDC_EDIT3, m_3);
	DDV_MinMaxInt(pDX, m_3, 0, 60);
	DDX_Text(pDX, IDC_EDIT4, m_4);
	DDV_MinMaxInt(pDX, m_4, 0, 60);
	DDX_Text(pDX, IDC_EDIT5, m_5);
	DDV_MinMaxInt(pDX, m_5, 0, 60);
	DDX_Text(pDX, IDC_EDIT6, m_6);
	DDV_MinMaxInt(pDX, m_6, 0, 60);
	DDX_Text(pDX, IDC_EDIT7, m_7);
	DDV_MinMaxInt(pDX, m_7, 0, 60);
	DDX_Text(pDX, IDC_EDIT8, m_8);
	DDV_MinMaxInt(pDX, m_8, 0, 60);
	DDX_Text(pDX, IDC_EDIT9, m_9);
	DDV_MinMaxInt(pDX, m_9, 0, 60);
	DDX_Text(pDX, IDC_EDIT16a, m_16a);
	//}}AFX_DATA_MAP
}

// Defining user-actions vis-a-vis system messages
BEGIN_MESSAGE_MAP(CMR1A1Dlg, CDialog)
	//{{AFX_MSG_MAP(CMR1A1Dlg)
	ON_BN_CLICKED(IDMR1A1P, OnMr1a1p)
	ON_BN_CLICKED(IDMR1A1N, OnMr1a1n)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

// Class method – Navigate to previous dialog
void CMR1A1Dlg::OnMr1a1p()
{
	EndDialog(IDCANCEL);
	CMR1ADlg a;
	a.DoModal();
}

// Class method – Compute validated mission time and navigate to next dialog
void CMR1A1Dlg::OnMr1a1n()
{
	UpdateData(TRUE);
	mt=0;
	if(m_1 > 0)
	mt += (m_1*3600);
	if(m_2 > 0)
	mt += (m_2*60);
	if(m_3 > 0)
	mt += (m_3);
	if(m_4 > 0)
	mt += (m_4*3600);
	if(m_5 > 0)
	mt += (m_5*60);
	if(m_6 > 0)
	mt += (m_6);
	if(m_7 > 0)
	mt += (m_7*3600);
	if(m_8 > 0)
	mt += (m_8*60);
	if(m_9 > 0)
	mt += (m_9);
	if(m_10 > 0)
	mt += (m_10*3600);
```

```
	if(m_11 > 0)
	mt += (m_11*60);
	if(m_12 > 0)
	mt += (m_12);
	if(m_13 > 0)
	mt += (m_13*3600);
	if(m_14 > 0)
	mt += (m_14*60);
	if(m_15 > 0)
	mt += (m_15);
	if(mt>0 && m_1<60 && m_2<60 && m_3<60 && m_4<60 && m_5<60 && m_6<60 && m_7<60
	&& m_8<60 && m_9<60 && m_10<60 && m_11<60 && m_12<60 && m_13<60 && m_14<60
	&& m_15<60)
	{
		if(MessageBox("Please verify the entered values after auto-validation. Ensure all input boxes
		have value in [0,60]. Do you want proceed further?", "Verify the Entered Values" ,
		MB_YESNO | MB_ICONQUESTION |MB_DEFBUTTON1 ) == IDYES)
		{
			m_16c.EnableWindow(TRUE);
			m_16a = mt;
			UpdateData(FALSE);
			m_16c.EnableWindow(FALSE);
			CString y;
			y.Format("%lf",mt);
			AfxMessageBox(" The computed mission time is "+y +"
			seconds");EndDialog(IDCANCEL);
			CUavdesignfcsDlg a;
			a.DoModal();
		}
	}
	else
	MessageBox("Please enter time in correct format","ZERO VALUE - EXCEED VALUE ERROR");
}
```

H.3 'Payload' class

The program code of 'Payload' class based on the acoustic-sensor and communication components of mission systems involves addition of selected mission systems weights to compute the payload weight as follows:

```
// File name: DPDlg.cpp
// Declaration of header files
#include "stdafx.h"
#include "VTUAV_Expert_System.h"
#include "DPDlg.h"
#include "Uavdesignfcs1Dlg.h"
#include "DPDlg1.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

// Declaration & initialisation of class variables
CDPDlg::CDPDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CDPDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CDPDlg)
```

```
	//}}AFX_DATA_INIT
}

// Limiting variables values
void CDPDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CDPDlg)
	//}}AFX_DATA_MAP
}

// Defining user-actions vis-a-vis system messages
BEGIN_MESSAGE_MAP(CDPDlg, CDialog)
	//{{AFX_MSG_MAP(CDPDlg)
	ON_BN_CLICKED(IDP1P, OnP1p)
	ON_BN_CLICKED(IDP1N, OnP1n)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

// Class method – Dialog initialisation
BOOL CDPDlg::OnInitDialog()
{
	CDialog::OnInitDialog();
	extern double pwt1;//si unit payload weight
	CString a;
	a.Format("%lf", pwt1);
	MessageBox("The following equipment and instrument capabilities should be installed on the UAV
	and/or be available to the supervising controller in order to comply with the requirements for safe flight
	under IFR procedures: Position Lights, Anti-Collision Lights, Transponder, Radios, Navigation
	Systems, UAV System and Altitude Displays, Flight and Voice Recorder, Built-in Test mechanisms.
	You may now select the payload as per your requirements. Your initially estimated payload weight was
	"+a+" kg", "SAFETY REQUIREMENTS - PAYLOAD DESIGN", MB_OK |
	MB_ICONINFORMATION  |MB_DEFBUTTON1);

	return TRUE;  // return TRUE unless you set the focus to a control
		// EXCEPTION: OCX Property Pages should return FALSE
}

// Class method – Navigate to previous dialog
void CDPDlg::OnP1p()
{
	// TODO: Add your control notification handler code here
	EndDialog(IDCANCEL);
	CUavdesignfcs1Dlg a(this);
	a.DoModal();

}

// Class method – Aggregate selected mission systems weight and navigate to next dialog
void CDPDlg::OnP1n()
{
	extern int n;// no. of checkboxes
	n=0;
	extern double pa;//payload
	pa=0;

	//INITIALISE Array to Null
	extern char* payloadname_final[74];
	extern double payloadwt_final[74];

	for(int i=0;i<=73;i++)
	{
		payloadname_final[i]="";
```

```
		payloadwt_final[i]=0.0;
	}

	//obviously assign null to [0] value
	payloadname_final[0]="";payloadwt_final[0]=0.0;


	if(IsDlgButtonChecked(IDC_CHECKBOX1)){
	pa+=1.3;n+=1;payloadname_final[1]="SENSOR ARRAY";payloadwt_final[1]=1.3;}
	if(IsDlgButtonChecked(IDC_CHECKBOX2)){
	pa+=2.3;n+=1;payloadname_final[2]="TRANSPONDER";payloadwt_final[2]=2.3;}
	if(IsDlgButtonChecked(IDC_CHECKBOX3)){
	pa+=4.0;n+=1;payloadname_final[3]="TRANSPONDER";payloadwt_final[3]=4.0;}
	if(IsDlgButtonChecked(IDC_CHECKBOX4)){
	pa+=4.5;n+=1;payloadname_final[4]="TRANSPONDER";payloadwt_final[4]=4.5;}
	if(IsDlgButtonChecked(IDC_CHECKBOX5)){
	pa+=4.0;n+=1;payloadname_final[5]="TRANSPONDER";payloadwt_final[5]=4.0;}
	if(IsDlgButtonChecked(IDC_CHECKBOX6)){
	pa+=4.2;n+=1;payloadname_final[6]="TRANSPONDER";payloadwt_final[6]=4.2;}
	if(IsDlgButtonChecked(IDC_CHECKBOX7)){
	pa+=3.0;n+=1;payloadname_final[7]="TRANSPONDER";payloadwt_final[7]=3.0;}
	if(IsDlgButtonChecked(IDC_CHECKBOX8)){
	pa+=5.0;n+=1;payloadname_final[8]="TRANSMITTER";payloadwt_final[8]=5.0;}
	if(IsDlgButtonChecked(IDC_CHECKBOX9)){
	pa+=4.0;n+=1;payloadname_final[9]="TRANSMITTER";payloadwt_final[9]=4.0;}
	if(IsDlgButtonChecked(IDC_CHECKBOX10)){
	pa+=2.5;n+=1;payloadname_final[10]="TERMINAL";payloadwt_final[10]=2.5;}
	if(IsDlgButtonChecked(IDC_CHECKBOX11)){
	pa+=1.0;n+=1;payloadname_final[11]="TERMINAL";payloadwt_final[11]=1.0;}
	if(IsDlgButtonChecked(IDC_CHECKBOX12)){
	pa+=15.0;n+=1;payloadname_final[12]="RECORDER";payloadwt_final[12]=15.0;}
	if(IsDlgButtonChecked(IDC_CHECKBOX13)){
	pa+=4.0;n+=1;payloadname_final[13]="RECORDER";payloadwt_final[13]=4.0;}
	if(IsDlgButtonChecked(IDC_CHECKBOX14)){
	pa+=45.0;n+=1;payloadname_final[14]="INTERCEPTOR";payloadwt_final[14]=45.0;}

	CString a;
	a.Format("%lf",pa);
	CString num;
	num.Format("%d", n);
	if(MessageBox("Please ensure that you have double clicked the checkboxes. As of now you have
	selected "+num+" different payload checkboxes and their total weight is "+a+" kg. Do you want to
	proceed further?","Verify and proceed Ahead" , MB_YESNO | MB_ICONQUESTION
	|MB_DEFBUTTON1 ) == IDYES)
	{
		EndDialog(IDCANCEL);
		CDPDlg1 a(this);
		a.DoModal();
	}
}
```

H.4 'Database-I' class

The programme code of 'Database-I' class involves identification of VTUAVs based on the design requirements for estimation of power as follows:

```
// File name:  DatabaseSearchDLG.cpp : Implementation file
// Inclusion of header files
#include "stdafx.h"
```

```
#include "VTUAV_Expert_System.h"
#include "DatabaseSearchDLG.h"
#include "GSDlg.h"
#include "_recordset.h"
#include "7_1Dlg.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

// Declaration & initialisation of class variables
CDatabaseSearchDLG::CDatabaseSearchDLG(CWnd* pParent /*=NULL*/)
	: CDialog(CDatabaseSearchDLG::IDD, pParent)
{
	//{{AFX_DATA_INIT(CDatabaseSearchDLG)
	//}}AFX_DATA_INIT
}

// Limiting variables values
void CDatabaseSearchDLG::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CDatabaseSearchDLG)
	DDX_Control(pDX, IDC_BUTTONPFG1, m_pfg1);
	DDX_Control(pDX, IDC_BUTTONPFG, m_pfg);
	DDX_Control(pDX, IDC_EDIT9c, m_9c);
	DDX_Control(pDX, IDC_EDIT9b, m_9b);
	DDX_Control(pDX, IDC_EDIT9a, m_9a);
	DDX_Control(pDX, IDC_EDIT9, m_9);
	DDX_Control(pDX, IDC_EDIT8, m_8);
	DDX_Control(pDX, IDC_EDIT7, m_7);
	DDX_Control(pDX, IDC_EDIT6, m_6);
	DDX_Control(pDX, IDC_EDIT5, m_5);
	DDX_Control(pDX, IDC_EDIT4, m_4);
	DDX_Control(pDX, IDC_EDIT3, m_3);
	DDX_Control(pDX, IDC_ADOSEARCH, m_adosearch);
	DDX_Control(pDX, IDC_DATAGRIDAS, m_dgas);
	DDX_Control(pDX, IDC_DATAGRIDDS, m_dgds);
	DDX_Control(pDX, IDC_ADOSEARCH2, m_adosearch2);
	//}}AFX_DATA_MAP
}

// Defining user-actions vis-à-vis system messages
BEGIN_MESSAGE_MAP(CDatabaseSearchDLG, CDialog)
	//{{AFX_MSG_MAP(CDatabaseSearchDLG)
	ON_BN_CLICKED(ID6B, On6b)
	ON_BN_CLICKED(ID6S, On6s)
	ON_BN_CLICKED(IDC_BUTTONPFG, OnButtonpfg)
	ON_BN_CLICKED(IDC_BUTTONPFG1, OnButtonpfg1)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()
// Class method – Initialisation of dialog
BOOL CDatabaseSearchDLG::OnInitDialog()
{
	CDialog::OnInitDialog();
	extern double ewt;//Empty weight
	extern double pwt;//Payload weight
	extern double lwt;//MTOW or launch weight
	extern double sp;//Forward speed
	extern double en;//Endurance
	extern double ra;//Range
	extern double al;//Altitude
```

```
	extern double le;//Fuselage length
	extern double sn;//Fuselage span or width
	extern double di;//Rotor diameter
	extern double sr;//Safety regulations
	extern double aw;//Stub wing or additional wing
	extern double fr;//Fuel reserve weight
	extern double mt;//Mission time
	extern double is;//Integrated system weight

	CString ewts, pwts, lwts, sps, ens, ras, als, les, sns, dis, srs, aws, frs, mts, iss;
	ewts.Format("%lf",ewt);
	pwts.Format("%lf",pwt);
	lwts.Format("%lf",lwt);
	sps.Format("%lf",sp);
	ens.Format("%lf",en);
	ras.Format("%lf",ra);
	als.Format("%lf",al);
	les.Format("%lf",le);
	sns.Format("%lf",sn);
	dis.Format("%lf",di);
	srs.Format("%lf",sr);
	aws.Format("%lf",aw);
	frs.Format("%lf",fr);
	mts.Format("%lf",mt);
	iss.Format("%lf",is);

	m_3.SetWindowText(lwts);
	m_4.SetWindowText(sps);
	m_5.SetWindowText(ens);
	m_6.SetWindowText(ras);
	m_7.SetWindowText(als);
	m_8.SetWindowText(les);
	m_9.SetWindowText(sns);
	m_9a.SetWindowText(dis);
	m_9b.SetWindowText(mts);
	m_9c.SetWindowText(iss);

	return TRUE;  // return TRUE unless you set the focus to a control
	// EXCEPTION: OCX Property Pages should return FALSE
}
```

// Class method – Navigate to previous dialog

```
void CDatabaseSearchDLG::On6b()
{
	EndDialog(IDCANCEL);
	CGSDlg a;
	a.DoModal();
}
```

// Class method – Search method

```
void CDatabaseSearchDLG::On6s()
{
//Determine the parameters
CString lwts,sps,ens,ras,als;
extern double lwt;
extern double sp;
extern double en;
extern double ra;
extern double al;
double wt_parameter;
double sp_parameter;
double en_parameter;
```

```
double ra_parameter;
double al_parameter;

//Weight limit
if (lwt <= 100)  wt_parameter = 12.5;
else if((lwt>100) && (lwt<=800) ) wt_parameter = 32;
else if((lwt>800) && (lwt<=1800) ) wt_parameter = 75;
else if((lwt>1800) && (lwt<=4000) ) wt_parameter = 280;
else  wt_parameter = 750;

//Speed limit
if (sp <= 62)  sp_parameter = 4;
else if((sp>62) && (sp<=75) ) sp_parameter = 7;
else if((sp>75) && (sp<=161) ) sp_parameter = 3;
else if((sp>161) && (sp<=300) ) sp_parameter = 10;
else if((sp>300) && (sp<=627) ) sp_parameter = 22;
else  sp_parameter = 220;

//Endurance limit
if (en <= 20)  en_parameter = 1;
else  en_parameter = 3.5;

//Range limit
if (ra <= 310)  ra_parameter = 10;
else if((ra>310) && (ra<=410) ) ra_parameter = 12.5;
else if((ra>410) && (ra<=823) ) ra_parameter = 35;
else if((ra>823) && (ra<=1356) ) ra_parameter = 70;
else if((ra>1356) && (ra<=2050) ) ra_parameter = 130;
else  ra_parameter = 510;

//Altitude limit
if (al <= 5000)  al_parameter = 500;
else if((al>5000) && (al<=21000) ) al_parameter = 750;
else if((al>21000) && (al<=26500) ) al_parameter = 1000;
else if((al>26500) && (al<=72182) ) al_parameter = 25000;
else  al_parameter = 75000;

CString query, query1;

//Pre-first ascending sort query
query = "SELECT TOP 5 NAME, WEIGHT, SPEED, ENDURANCE, RANGE, ALTITUDE, ENGINE,
MISSION, LENGTH, WINGSPAN, BODYDIA FROM Querytable WHERE (WEIGHT BETWEEN ";

lwts.Format("%lf",lwt);
query+=lwts;
query+=" AND ";
lwts.Format("%lf",(lwt + wt_parameter));
query+=lwts;
query+=") AND (SPEED BETWEEN ";
sps.Format("%lf",sp);
query+=sps;
query+=" AND ";
sps.Format("%lf",(sp + sp_parameter));
query+=sps;
query+=") AND (ENDURANCE BETWEEN ";
ens.Format("%lf",en);
query+=ens;
query+=" AND ";
ens.Format("%lf",(en + en_parameter));
query+=ens;
query+=") AND (RANGE BETWEEN ";
ras.Format("%lf",ra);
query+=ras;
```

```
query+=" AND ";
ras.Format("%lf",(ra + ra_parameter));
query+=ras;
query+=") AND (ALTITUDE BETWEEN ";
als.Format("%lf",al);
query+=als;
query+=" AND ";
als.Format("%lf",(al + al_parameter));
query+=als;
query+=") ORDER BY WEIGHT, SPEED, ENDURANCE, RANGE, ALTITUDE;";

//Pre-first descending sort query
query1 = "SELECT TOP 5 NAME, WEIGHT, SPEED, ENDURANCE, RANGE, ALTITUDE, ENGINE,
MISSION, LENGTH, WINGSPAN, BODYDIA FROM Querytable WHERE (WEIGHT BETWEEN ";

lwts.Format("%lf",(lwt - wt_parameter));
query1+=lwts;
query1+=" AND ";
lwts.Format("%lf",lwt);
query1+=lwts;
query1+=") AND (SPEED BETWEEN ";
sps.Format("%lf",(sp - sp_parameter));
query1+=sps;
query1+=" AND ";
sps.Format("%lf",sp);
query1+=sps;
query1+=") AND (ENDURANCE BETWEEN ";
ens.Format("%lf",(en - en_parameter));
query1+=ens;
query1+=" AND ";
ens.Format("%lf",en);
query1+=ens;
query1+=") AND (RANGE BETWEEN ";
ras.Format("%lf",(ra - ra_parameter));
query1+=ras;
query1+=" AND ";
ras.Format("%lf",ra);
query1+=ras;
query1+=") AND (ALTITUDE BETWEEN ";
als.Format("%lf",(al - al_parameter));
query1+=als;
query1+=" AND ";
als.Format("%lf",al);
query1+=als;
query1+=") ORDER BY WEIGHT DESC, SPEED DESC, ENDURANCE DESC, RANGE DESC, ALTITUDE
DESC";

//Set the record-source and update
m_adosearch.SetRecordSource(query);
m_adosearch.Refresh();
m_adosearch2.SetRecordSource(query1);
m_adosearch2.Refresh();
if(m_adosearch.GetRecordset().GetRecordCount() == 0 && m_adosearch2.GetRecordset(). GetRecordCount()
== 0)
{
        //First ascending sort query involving /weight, speed, endurance, range
        query="";
        query1="";
        query = "SELECT TOP 5 NAME, WEIGHT, SPEED, ENDURANCE, RANGE, ALTITUDE,
        ENGINE, MISSION, LENGTH, WINGSPAN, BODYDIA FROM Querytable WHERE (WEIGHT
        BETWEEN ";

        lwts.Format("%lf",lwt);
```

```
query+=lwts;
query+=" AND ";
lwts.Format("%lf",(lwt + wt_parameter));
query+=lwts;
query+=") AND (SPEED BETWEEN ";
sps.Format("%lf",sp);
query+=sps;
query+=" AND ";
sps.Format("%lf",(sp + sp_parameter));
query+=sps;
query+=") AND (ENDURANCE BETWEEN ";
ens.Format("%lf",en);
query+=ens;
query+=" AND ";
ens.Format("%lf",(en + en_parameter));
query+=ens;
query+=") AND (RANGE BETWEEN ";
ras.Format("%lf",ra);
query+=ras;
query+=" AND ";
ras.Format("%lf",(ra + ra_parameter));
query+=ras;
query+=") ORDER BY WEIGHT, SPEED, ENDURANCE, RANGE, ALTITUDE ";

//First descending sort query
query1 = "SELECT TOP 5 NAME, WEIGHT, SPEED, ENDURANCE, RANGE, ALTITUDE,
ENGINE, MISSION, LENGTH, WINGSPAN, BODYDIA FROM Querytable WHERE (WEIGHT
BETWEEN ";

lwts.Format("%lf",(lwt - wt_parameter));
query1+=lwts;
query1+=" AND ";
lwts.Format("%lf",lwt);
query1+=lwts;
query1+=") AND (SPEED BETWEEN ";
sps.Format("%lf",(sp - sp_parameter));
query1+=sps;
query1+=" AND ";
sps.Format("%lf",sp);
query1+=sps;
query1+=") AND (ENDURANCE BETWEEN ";
ens.Format("%lf",(en - en_parameter));
query1+=ens;
query1+=" AND ";
ens.Format("%lf",en);
query1+=ens;
query1+=") AND (RANGE BETWEEN ";
ras.Format("%lf",(ra - ra_parameter));
query1+=ras;
query1+=" AND ";
ras.Format("%lf",ra);
query1+=ras;
query1+=") ORDER BY WEIGHT DESC, SPEED DESC, ENDURANCE DESC, RANGE DESC,
ALTITUDE DESC";

//Set the record-source and update
m_adosearch.SetRecordSource(query);
m_adosearch.Refresh();
m_adosearch2.SetRecordSource(query1);
m_adosearch2.Refresh();

//Second query involving weight, speed and endurance
```

```
if(m_adosearch.GetRecordset().GetRecordCount() == 0 &&
m_adosearch2.GetRecordset().GetRecordCount() == 0)
{
        query="";
        query1="";

        //Second ascending sort query
        query = "SELECT TOP 5 NAME, WEIGHT, SPEED, ENDURANCE, RANGE, ALTITUDE,
        ENGINE, MISSION, LENGTH, WINGSPAN, BODYDIA FROM Querytable WHERE
        (WEIGHT BETWEEN ";

        lwts.Format("%lf",lwt);
        query+=lwts;
        query+=" AND ";
        lwts.Format("%lf",(lwt + wt_parameter));
        query+=lwts;
        query+=") AND (SPEED BETWEEN ";
        sps.Format("%lf",sp);
        query+=sps;
        query+=" AND ";
        sps.Format("%lf",(sp + sp_parameter));
        query+=sps;
        query+=") AND (ENDURANCE BETWEEN ";
        ens.Format("%lf",en);
        query+=ens;
        query+=" AND ";
        ens.Format("%lf",(en + en_parameter));
        query+=ens;
        query+=") ORDER BY WEIGHT, SPEED, ENDURANCE, RANGE, ALTITUDE ";

        //Second descending sort query
        query1 = "SELECT TOP 5 NAME, WEIGHT, SPEED, ENDURANCE, RANGE,
        ALTITUDE, ENGINE, MISSION, LENGTH, WINGSPAN, BODYDIA FROM Querytable
        WHERE (WEIGHT BETWEEN ";

        lwts.Format("%lf",(lwt - wt_parameter));
        query1+=lwts;
        query1+=" AND ";
        lwts.Format("%lf",lwt);
        query1+=lwts;
        query1+=") AND (SPEED BETWEEN ";
        sps.Format("%lf",(sp - sp_parameter));
        query1+=sps;
        query1+=" AND ";
        sps.Format("%lf",sp);
        query1+=sps;
        query1+=") AND (ENDURANCE BETWEEN ";
        ens.Format("%lf",(en - en_parameter));
        query1+=ens;
        query1+=" AND ";
        ens.Format("%lf",en);
        query1+=ens;
        query1+=") ORDER BY WEIGHT DESC, SPEED DESC, ENDURANCE DESC, RANGE
        DESC, ALTITUDE DESC";

        m_adosearch.SetRecordSource(query);
        m_adosearch.Refresh();
        m_adosearch2.SetRecordSource(query1);
        m_adosearch2.Refresh();

        //Weight and speed
        if(m_adosearch.GetRecordset().GetRecordCount() == 0 &&
        m_adosearch2.GetRecordset().GetRecordCount() == 0)
```

```
{
        query="";
        query1="";

        //Third ascending sort query
        query = "SELECT TOP 5 NAME, WEIGHT, SPEED, ENDURANCE, RANGE,
        ALTITUDE, ENGINE, MISSION, LENGTH, WINGSPAN, BODYDIA FROM
        Querytable WHERE (WEIGHT BETWEEN ";

        lwts.Format("%lf",lwt);
        query+=lwts;
        query+=" AND ";
        lwts.Format("%lf",(lwt + wt_parameter));
        query+=lwts;
        query+=") AND (SPEED BETWEEN ";
        sps.Format("%lf",sp);
        query+=sps;
        query+=" AND ";
        sps.Format("%lf",(sp + sp_parameter));
        query+=sps;
        query+=") ORDER BY WEIGHT, SPEED, ENDURANCE, RANGE, ALTITUDE ";


        //Third descending sort query
        query1 = "SELECT TOP 5 NAME, WEIGHT, SPEED, ENDURANCE, RANGE,
        ALTITUDE, ENGINE, MISSION, LENGTH, WINGSPAN, BODYDIA FROM
        Querytable WHERE (WEIGHT BETWEEN ";

        lwts.Format("%lf",(lwt - wt_parameter));
        query1+=lwts;
        query1+=" AND ";
        lwts.Format("%lf",lwt);
        query1+=lwts;
        query1+=") AND (SPEED BETWEEN ";
        sps.Format("%lf",(sp - sp_parameter));
        query1+=sps;
        query1+=" AND ";
        sps.Format("%lf",sp);
        query1+=sps;
        query1+=") ORDER BY WEIGHT DESC, SPEED DESC, ENDURANCE DESC,
        RANGE DESC, ALTITUDE DESC";

        m_adosearch.SetRecordSource(query);
        m_adosearch.Refresh();
        m_adosearch2.SetRecordSource(query1);
        m_adosearch2.Refresh();

        //Only weight search
        if(m_adosearch.GetRecordset().GetRecordCount() == 0 &&
        m_adosearch2.GetRecordset().GetRecordCount() == 0)
        {
                query="";
                query1="";

                //Fourth ascending sort query
                query = "SELECT TOP 5 NAME, WEIGHT, SPEED, ENDURANCE,
                RANGE, ALTITUDE, ENGINE, MISSION, LENGTH, WINGSPAN,
                BODYDIA FROM Querytable WHERE (WEIGHT BETWEEN ";

                lwts.Format("%lf",lwt);
                query+=lwts;
                query+=" AND ";
                lwts.Format("%lf",(lwt + wt_parameter));
```

```
                                query+=lwts;
                                query+=") ORDER BY WEIGHT, SPEED, ENDURANCE, RANGE,
                                ALTITUDE ";

                                //Fourth descending sort query
                                query1 = "SELECT TOP 5 NAME, WEIGHT, SPEED, ENDURANCE,
                                RANGE, ALTITUDE, ENGINE, MISSION, LENGTH, WINGSPAN,
                                BODYDIA FROM Querytable WHERE (WEIGHT BETWEEN ";

                                lwts.Format("%lf",(lwt - wt_parameter));
                                query1+=lwts;
                                query1+=" AND ";
                                lwts.Format("%lf",lwt);
                                query1+=lwts;
                                query1+=") ORDER BY WEIGHT DESC, SPEED DESC, ENDURANCE
                                DESC, RANGE DESC, ALTITUDE DESC";

                                m_adosearch.SetRecordSource(query);
                                m_adosearch.Refresh();
                                m_adosearch2.SetRecordSource(query1);
                                m_adosearch2.Refresh();

                        }

                }

        }

}

UpdateData(FALSE);

if(m_adosearch.GetRecordset().GetRecordCount() > 0 && m_adosearch2.GetRecordset().GetRecordCount() >
0)
{
        m_pfg.EnableWindow(TRUE);
        m_pfg.ShowWindow(SW_SHOW);
        m_pfg1.EnableWindow(TRUE);
        m_pfg1.ShowWindow(SW_SHOW);
        MessageBox("Results have been found on both data lists. The program pick is "+m_dgas.GetText() +"
        UAV. However, you may override the default selection by clicking on the name of the aircraft(column
        1) that you want to select from the two data grids and then click the corresponding 'Pick From Grid'
        button", "State your preference");
}

//list 1 result chosen automatically
else if (m_adosearch.GetRecordset().GetRecordCount() > 0 )
{
        m_pfg.EnableWindow(TRUE);
        m_pfg.ShowWindow(SW_SHOW);
        MessageBox("Results have been found in first datalist. The program pick is "+m_dgas.GetText() +"
        UAV. Please click on the name of the aircraft(column 1)that you want to select from the first datagrid
        and then click the corresponding 'Pick From Grid' button", "State your preference");
}

//list 2 result chosen automatically
else if (m_adosearch2.GetRecordset().GetRecordCount() > 0 )
{
        m_pfg1.EnableWindow(TRUE);
        m_pfg1.ShowWindow(SW_SHOW);
        MessageBox("Results have been found in second datalist. The program pick is "+m_dgds.GetText() +"
        UAV. Please click on the name of the aircraft(column 1)that you want to select from the second
        datagrid and then click the corresponding 'Pick From Grid' button", "State your preference");
```

```
}

//no records matched
else
{
        MessageBox("No results found for the input parameters. Change the values all over again.",
        "Unprecedented Aircraft Parameters");
}

}

// Class method – Selection method (Algorithm 1)
void CDatabaseSearchDLG::OnButtonpfg()
{
        extern CString uavname_dbsearch;
        extern CString uavmission_dbsearch;
        extern CString uavengine_dbsearch;
        extern double uavweight_dbsearch;
        extern double uavpower_dbsearch;
        extern double uavspeed_dbsearch;
        extern double uavendurance_dbsearch;
        extern double uavrange_dbsearch;
        extern double uavaltitude_dbsearch;
        extern double uavlength_dbsearch;
        extern double uavwingspan_dbsearch;
        extern double uavrotorspan_dbsearch;
        extern double uavdiameter_dbsearch;

        //Pointer to ending  number in a string
        char  *stopstring;
        uavname_dbsearch = m_dgas.GetText();
        CString query_check;
        query_check = "SELECT * FROM Querytable WHERE NAME ='";
        query_check+=m_dgas.GetText();
        query_check+="'";
        m_adosearch.SetRecordSource(query_check);
          m_adosearch.Refresh();

        if(m_adosearch.GetRecordset().GetRecordCount() > 0)
        {
                query_check = "SELECT MISSION FROM Querytable WHERE NAME ='";
                query_check+=m_dgas.GetText();
                query_check+="'";
                m_adosearch2.SetRecordSource(query_check);
                m_adosearch2.Refresh();
                uavmission_dbsearch = m_dgds.GetText();

                /Results not displayed immediately but used later
                query_check = "SELECT ENGINE FROM Querytable WHERE NAME ='";
                query_check+=m_dgas.GetText();
                query_check+="'";
                m_adosearch2.SetRecordSource(query_check);
                m_adosearch2.Refresh();
                uavengine_dbsearch = m_dgds.GetText();

                query_check = "SELECT WEIGHT FROM Querytable WHERE NAME ='";
                query_check+=m_dgas.GetText();
                query_check+="'";
                m_adosearch2.SetRecordSource(query_check);
                m_adosearch2.Refresh();
                uavweight_dbsearch = strtod(m_dgds.GetText(),&stopstring);

                query_check = "SELECT POWER FROM Querytable WHERE NAME ='";
```

```
		query_check+=m_dgas.GetText();
		query_check+="'";
		m_adosearch2.SetRecordSource(query_check);
		m_adosearch2.Refresh();
		uavpower_dbsearch = strtod(m_dgds.GetText(),&stopstring);

		query_check = "SELECT SPEED FROM Querytable WHERE NAME ='";
		query_check+=m_dgas.GetText();
		query_check+="'";
		m_adosearch2.SetRecordSource(query_check);
		m_adosearch2.Refresh();
		uavspeed_dbsearch = strtod(m_dgds.GetText(),&stopstring);

		query_check = "SELECT ENDURANCE FROM Querytable WHERE NAME ='";
		query_check+=m_dgas.GetText();
		query_check+="'";
		m_adosearch2.SetRecordSource(query_check);
		m_adosearch2.Refresh();
		uavendurance_dbsearch = strtod(m_dgds.GetText(),&stopstring);

		query_check = "SELECT RANGE FROM Querytable WHERE NAME ='";
		query_check+=m_dgas.GetText();
		query_check+="'";
		m_adosearch2.SetRecordSource(query_check);
		m_adosearch2.Refresh();
		uavrange_dbsearch = strtod(m_dgds.GetText(),&stopstring);

		query_check = "SELECT ALTITUDE FROM Querytable WHERE NAME ='";
		query_check+=m_dgas.GetText();
		query_check+="'";
		m_adosearch2.SetRecordSource(query_check);
		m_adosearch2.Refresh();
		uavaltitude_dbsearch = strtod(m_dgds.GetText(),&stopstring);

		CString suavweight_dbsearch;
		CString suavpower_dbsearch;
		CString suavspeed_dbsearch;
		CString suavendurance_dbsearch;
		CString suavrange_dbsearch;
		CString suavaltitude_dbsearch;

		suavweight_dbsearch.Format("%lf",uavweight_dbsearch);
		suavpower_dbsearch.Format("%lf",uavpower_dbsearch);
		suavspeed_dbsearch.Format("%lf",uavspeed_dbsearch);
		suavendurance_dbsearch.Format("%lf",uavendurance_dbsearch);
		suavrange_dbsearch.Format("%lf",uavrange_dbsearch);
		suavaltitude_dbsearch.Format("%lf",uavaltitude_dbsearch);

		MessageBox("The selection is as follows. Name = '"+uavname_dbsearch +"',
		Mission='"+uavmission_dbsearch +"', Weight(lbs)='"+suavweight_dbsearch
		+"',Power(hp)='"+suavpower_dbsearch +"', Speed(miles/hr)='"+suavspeed_dbsearch +"',
		Range(miles)='"+suavrange_dbsearch +"', Endurance(hour)='"+suavendurance_dbsearch
		+"',Altitude(ft.)='"+suavaltitude_dbsearch +"'.","Retrieved Success");

		EndDialog(IDCANCEL);
		C7_1Dlg a;
		a.DoModal();
}

else
{
		MessageBox("Please select the name field only. You may have to click the search button once
		more.","Warning");
```

```
	}

}

// Class method – Selection method (Algorithm 2)
void CDatabaseSearchDLG::OnButtonpfg1()
{
	extern CString uavname_dbsearch;
	extern CString uavmission_dbsearch;
	extern CString uavengine_dbsearch;
	extern double uavweight_dbsearch;
	extern double uavpower_dbsearch;
	extern double uavspeed_dbsearch;
	extern double uavendurance_dbsearch;
	extern double uavrange_dbsearch;
	extern double uavaltitude_dbsearch;
	extern double uavlength_dbsearch;
	extern double uavwingspan_dbsearch;
	extern double uavrotorspan_dbsearch;
	extern double uavdiameter_dbsearch;

	//Pointer to ending number in a string
	char  *stopstring;
	CString suavweight_dbsearch;
	CString suavpower_dbsearch;
	CString suavspeed_dbsearch;
	CString suavendurance_dbsearch;
	CString suavrange_dbsearch;
	CString suavaltitude_dbsearch;

	uavname_dbsearch = m_dgds.GetText();
	CString query_check;
	query_check = "SELECT * FROM Querytable WHERE NAME ='";
	query_check+=m_dgds.GetText();
	query_check+="'";
	m_adosearch2.SetRecordSource(query_check);
	m_adosearch2.Refresh();

	if(m_adosearch2.GetRecordset().GetRecordCount() > 0)
	{
		query_check = "SELECT MISSION FROM Querytable WHERE NAME ='";
		query_check+=m_dgds.GetText();
		query_check+="'";
		m_adosearch.SetRecordSource(query_check);
		m_adosearch.Refresh();
		uavmission_dbsearch = m_dgas.GetText();

		//Results not displayed immediately but used later
		query_check = "SELECT ENGINE FROM Querytable WHERE NAME ='";
		query_check+=m_dgds.GetText();
		query_check+="'";
		m_adosearch.SetRecordSource(query_check);
		m_adosearch.Refresh();
		uavengine_dbsearch = m_dgas.GetText();

		query_check = "SELECT WEIGHT FROM Querytable WHERE NAME ='";
		query_check+=m_dgds.GetText();
		query_check+="'";
		m_adosearch.SetRecordSource(query_check);
		m_adosearch.Refresh();
		uavweight_dbsearch = strtod(m_dgas.GetText(),&stopstring);

		query_check = "SELECT POWER FROM Querytable WHERE NAME ='";
```

```
		query_check+=m_dgds.GetText();
		query_check+="'";
		m_adosearch.SetRecordSource(query_check);
		m_adosearch.Refresh();
		uavpower_dbsearch = strtod(m_dgas.GetText(),&stopstring);

		query_check = "SELECT SPEED FROM Querytable WHERE NAME ='";
		query_check+=m_dgds.GetText();
		query_check+="'";
		m_adosearch.SetRecordSource(query_check);
		m_adosearch.Refresh();
		uavspeed_dbsearch = strtod(m_dgas.GetText(),&stopstring);

		query_check = "SELECT ENDURANCE FROM Querytable WHERE NAME ='";
		query_check+=m_dgds.GetText();
		query_check+="'";
		m_adosearch.SetRecordSource(query_check);
		m_adosearch.Refresh();
		uavendurance_dbsearch = strtod(m_dgas.GetText(),&stopstring);

		query_check = "SELECT RANGE FROM Querytable WHERE NAME ='";
		query_check+=m_dgds.GetText();
		query_check+="'";
		m_adosearch.SetRecordSource(query_check);
		m_adosearch.Refresh();
		uavrange_dbsearch = strtod(m_dgas.GetText(),&stopstring);

		query_check = "SELECT ALTITUDE FROM Querytable WHERE NAME ='";
		query_check+=m_dgds.GetText();
		query_check+="'";
		m_adosearch.SetRecordSource(query_check);
		m_adosearch.Refresh();
		uavaltitude_dbsearch = strtod(m_dgas.GetText(),&stopstring);

		suavweight_dbsearch.Format("%lf",uavweight_dbsearch);
		suavpower_dbsearch.Format("%lf",uavpower_dbsearch);
		suavspeed_dbsearch.Format("%lf",uavspeed_dbsearch);
		suavendurance_dbsearch.Format("%lf",uavendurance_dbsearch);
		suavrange_dbsearch.Format("%lf",uavrange_dbsearch);
		suavaltitude_dbsearch.Format("%lf",uavaltitude_dbsearch);

		MessageBox("The selection is as follows. Name = '"+uavname_dbsearch +"',
		Mission='"+uavmission_dbsearch +"', Weight(lbs)='"+suavweight_dbsearch
		+"',Power(hp)='"+suavpower_dbsearch +"', Speed(miles/hr)='"+suavspeed_dbsearch +"',
		Range(miles)='"+suavrange_dbsearch +"', Endurance(hour)='"+suavendurance_dbsearch
		+"',Altitude(ft.)='"+suavaltitude_dbsearch +"'.","Retrieved Success");

		EndDialog(IDCANCEL);
		C7_1Dlg a;
		a.DoModal();
	}

	else
	{
		MessageBox("Please select the name field only. You may have to click the search button once
		more.","Warning");
	}

}
```

H.5 ‘Fuel’, ‘Structural weight’, ‘Vertical drag’, & ‘Rotor system-I’ Classes

The fuel, structural weight, vertical drag and rotor system classes involve the evaluation of fuel, structural weight, vertical drag, and number of rotor blades based on the coaxial configuration. These classes are programmed in a single dialog to provide maximum information as follows:

```
// File name: 7_1Dlg.cpp
// Declaration of header files
#include "stdafx.h"
#include "VTUAV_Expert_System.h"
#include "7_1Dlg.h"
#include "MRDlg.h"
#include "7_2Dlg.h"
#include "InputDlg.h"
#include <cmath>   // equivalent of math.h
#include "DatabaseSearchDLG.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

// Declaration & initialisation of class variables
C7_1Dlg::C7_1Dlg(CWnd* pParent /*=NULL*/)
	: CDialog(C7_1Dlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(C7_1Dlg)
	m_1 = _T("");
	m_2 = _T("");
	m_3 = _T("");
	m_4 = _T("");
	m_5 = _T("");
	m_6 = _T("");
	m_10a = _T("");
	m_7 = _T("");
	m_8 = _T("");
	m_9 = _T("");
	m_6a = _T("");
	m_6a2 = _T("");
	m_6a3 = _T("");
	m_6a4 = _T("");
	m_11 = _T("");
	m_12 = _T("");
	m_13 = _T("");
	m_14 = _T("");
	m_15 = _T("");
	m_17 = _T("");
	m_18 = _T("");
	m_19 = _T("");
	m_20 = _T("");
	m_19a = _T("");
	m_21 = _T("");
	m_22 = _T("");
	m_23 = _T("");
	m_24 = _T("");
	m_25 = _T("");
```

```
	m_26 = _T("");
	m_27 = _T("");
	m_28 = _T("");
	m_6a5 = _T("");
	m_6a6 = _T("");
	m_6a7 = _T("");
	m_6a8 = _T("");
	//}}AFX_DATA_INIT
}

// Limiting variables values
void C7_1Dlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(C7_1Dlg)
	DDX_Text(pDX, IDC_EDIT1, m_1);
	DDX_Text(pDX, IDC_EDIT2, m_2);
	DDX_Text(pDX, IDC_EDIT3, m_3);
	DDX_Text(pDX, IDC_EDIT4, m_4);
	DDX_Text(pDX, IDC_EDIT5, m_5);
	DDX_Text(pDX, IDC_EDIT6, m_6);
	DDX_Text(pDX, IDC_EDIT10a, m_10a);
	DDX_Text(pDX, IDC_EDIT7, m_7);
	DDX_Text(pDX, IDC_EDIT8, m_8);
	DDX_Text(pDX, IDC_EDIT9, m_9);
	DDX_Control(pDX, IDC_LABELFE, m_labelfr);
	DDX_Text(pDX, IDC_EDIT6a, m_6a);
	DDX_Text(pDX, IDC_EDIT6a2, m_6a2);
	DDX_Text(pDX, IDC_EDIT6a3, m_6a3);
	DDX_Text(pDX, IDC_EDIT6a4, m_6a4);
	DDX_Text(pDX, IDC_EDIT11, m_11);
	DDX_Text(pDX, IDC_EDIT12, m_12);
	DDX_Text(pDX, IDC_EDIT13, m_13);
	DDX_Text(pDX, IDC_EDIT14, m_14);
	DDX_Text(pDX, IDC_EDIT15, m_15);
	DDX_Text(pDX, IDC_EDIT17, m_17);
	DDX_Text(pDX, IDC_EDIT18, m_18);
	DDX_Text(pDX, IDC_EDIT19, m_19);
	DDX_Text(pDX, IDC_EDIT20, m_20);
	DDX_Text(pDX, IDC_EDIT19a, m_19a);
	DDX_Text(pDX, IDC_EDIT21, m_21);
	DDX_Text(pDX, IDC_EDIT22, m_22);
	DDX_Text(pDX, IDC_EDIT23, m_23);
	DDX_Text(pDX, IDC_EDIT24, m_24);
	DDX_Text(pDX, IDC_EDIT25, m_25);
	DDX_Text(pDX, IDC_EDIT26, m_26);
	DDX_Text(pDX, IDC_EDIT27, m_27);
	DDX_Text(pDX, IDC_EDIT28, m_28);
	DDX_Text(pDX, IDC_EDIT6a5, m_6a5);
	DDX_Text(pDX, IDC_EDIT6a6, m_6a6);
	DDX_Text(pDX, IDC_EDIT6a7, m_6a7);
	DDX_Text(pDX, IDC_EDIT6a8, m_6a8);
	//}}AFX_DATA_MAP
}

// Defining user-actions vis-a-vis system messages
BEGIN_MESSAGE_MAP(C7_1Dlg, CDialog)
	//{{AFX_MSG_MAP(C7_1Dlg)
	ON_BN_CLICKED(ID7B, On7b)
	ON_BN_CLICKED(ID7H1, On7h1)
	ON_BN_CLICKED(ID7H2, On7h2)
	ON_BN_CLICKED(ID7H3, On7h3)
	ON_BN_CLICKED(ID7H4, On7h4)
```

```
	ON_BN_CLICKED(ID7H5, On7h5)
	ON_BN_CLICKED(ID7H6, On7h6)
	ON_BN_CLICKED(ID7H7, On7h7)
	ON_BN_CLICKED(ID7H8, On7h8)
	ON_BN_CLICKED(ID7H9, On7h9)
	ON_BN_CLICKED(ID7H10, On7h10)
	ON_BN_CLICKED(ID7H11, On7h11)
	ON_BN_CLICKED(ID7H12, On7h12)
	ON_BN_CLICKED(ID7H13, On7h13)
	ON_BN_CLICKED(ID7H14, On7h14)
	ON_BN_CLICKED(ID7H15, On7h15)
	ON_BN_CLICKED(ID7H16, On7h16)
	ON_BN_CLICKED(ID7H17, On7h17)
	ON_BN_CLICKED(ID7H19, On7h19)
	ON_BN_CLICKED(ID7H20, On7h20)
	ON_BN_CLICKED(ID7H21, On7h21)
	ON_BN_CLICKED(ID7H22, On7h22)
	ON_BN_CLICKED(ID7H23, On7h23)
	ON_BN_CLICKED(ID7H24, On7h24)
	ON_BN_CLICKED(ID7H6a, On7H6a)
	ON_BN_CLICKED(ID7N, On7n)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

// Class method – Initialisation of dialog
BOOL C7_1Dlg::OnInitDialog()
{
	CDialog::OnInitDialog();
	extern CString uavname_dbsearch;
	extern CString uavmission_dbsearch;
	extern CString uavengine_dbsearch;
	extern double uavweight_dbsearch;
	extern double uavpower_dbsearch;
	extern double uavspeed_dbsearch;
	extern double uavendurance_dbsearch;
	extern double uavrange_dbsearch;
	extern double uavaltitude_dbsearch;
	extern double uavlength_dbsearch;
	extern double uavwingspan_dbsearch;
	extern double uavrotorspan_dbsearch;
	extern double uavdiameter_dbsearch;

	extern double ewt;//Empty weight
	extern double pwt;//Payload weight
	extern double lwt;//MTOW or launch weight
	extern double mt;//Mission time
	extern double fr;//Fuel reserve
	extern double en;//Endurance
	extern double sp;//Forward Speed
	extern double al;//Altitude
	extern double uavprojectedarea;// Interim variable to store the value

	//To calculate power later
	extern double ARD_method;//Area of rotor disc
	extern double TS_method;//Tip Speed
	extern double RR_method;//Rotor radius
	extern double SR_method;//Solidity of rotor
	extern double fuel_system;//To record fuel weight
	extern double engine_weight;//To record engine weight

	int fv;
	if(fr==0)
	fv=2;
```

```
else if(fr==1)
fv=20;
else //fr==2
fv=30;

CString a,b,c;
c.Format("%d",fv);
b="FUEL RESERVE, LB - ( ";
b+=c;
b+=" MINUTES)";

m_labelfr.SetWindowText(b);

//edit1 value
if((1500*ewt) < (8000*pwt))
{
        a.Format("%ld",int(1500*ewt));
        m_1=a;
        m_1+=" to ";
        a="";
        a.Format("%ld",int(8000*pwt));
        m_1+=a;
}
else
{
        a.Format("%ld",int(8000*pwt));
        m_1=a;
        m_1+=" to ";
        a="";
        a.Format("%ld",int(1500*ewt));
        m_1+=a;
}

//edit2 to be found
m_2.Format("%ld",int(120000 + (333.34*en*pwt)));

//edit3 value
m_3.Format("%g",(pwt+(0.5*uavpower_dbsearch)));
m_4.Format("%lf",uavpower_dbsearch);
m_5 = uavengine_dbsearch;
m_6.Format("%g",(0.45*uavpower_dbsearch*mt/3600));

//Check  fuel values feasibilit
if((0.45*uavpower_dbsearch*mt/3600) < 0.55*(lwt)) //.55 - fuel + .15 - engine
{
        m_6.Format("%g",(0.45*uavpower_dbsearch*mt/3600));
        m_6a.Format("%g",(0.45*uavpower_dbsearch*fv/60));
        m_6a2.Format("%g",((0.45*uavpower_dbsearch*mt/3600)+(0.45*uavpower_dbsearch*fv/60))
        );
        m_6a3.Format("%g",0.04*((0.45*uavpower_dbsearch*mt/3600)+(0.45*uavpower_dbsearch*f
        v/60)));
        m_6a4.Format("%g",1.04*((0.45*uavpower_dbsearch*mt/3600)+(0.45*uavpower_dbsearch*f
        v/60)));
        fuel_system=1.04*((0.45*uavpower_dbsearch*mt/3600)+(0.45*uavpower_dbsearch*fv/60)) ;
}
else
{
        a.Format("%g",lwt);
        MessageBox("Fuel weight = sfc x power x mission-time.  Fuel weight cannot be more than
        55% of gross weight. Current fuel weight: "+m_6+" lb and gross weight: "+a+" lb. You have
        to decrease the mission-time & endurance duration;  therefore are being redirected to input
        dialog.", "MISMATCHED VALUES");
        EndDialog(IDOK);
```

```
	CInputDlg a;
	a.DoModal();
}

//check if fuel is more than 37.8 litres
m_6a5.Format("%g", (((0.45*uavpower_dbsearch*mt/3600) +
(0.45*uavpower_dbsearch*fv/60))/2.2)*0.716);
if(((((0.45*uavpower_dbsearch*mt/3600)+(0.45*uavpower_dbsearch*fv/60))/2.2)*0.716) > 37.8)
m_6a6= "Safety regulations assert that fuel tank of 37.8 litres or greater capacity must have internal
baffles, or must have external support to resist surging. Besides, fuel tank must be separated from
engine compartment by a firewall";

else
m_6a6= "No need for internal baffles in fuel tank";
m_6a7.Format("%g",
(((0.45*uavpower_dbsearch*mt/3600)+(0.45*uavpower_dbsearch*fv/60))/2.2)*0.795);

if(((((0.45*uavpower_dbsearch*mt/3600)+(0.45*uavpower_dbsearch*fv/60))/2.2)*0.795) > 37.8)
m_6a8= "Safety regulations assert that fuel tank of 37.8 litres or greater capacity must have internal
baffles, or must have external support to resist surging. Besides, fuel tank must be separated from
engine compartment by a firewall";

else
m_6a8= "No need for internal baffles in fuel tank";
m_7.Format("%g",pwt);
m_8=m_6a4;
m_9.Format("%g",0.15*lwt);
engine_weight=0.15*lwt;
m_10a.Format("%g",(lwt-(pwt + 1.04*((0.45*uavpower_dbsearch*mt/3600) +
(0.45*uavpower_dbsearch*fv/60)) + (0.15*lwt))));
m_11.Format("%g",uavprojectedarea);
m_12.Format("%g",0.3);
m_13.Format("%g",4.0);
m_14.Format("%g",0.3*4.0*uavprojectedarea);
m_15= "COUNTER-ROTATING CO-AXIAL ROTOR SYSTEM";
//m_16 var not formed
m_17.Format("%g",(lwt+(0.3*4.0*uavprojectedarea))/4.0);
m_18.Format("%g",sqrt(((lwt+(0.3*4.0*uavprojectedarea))/4.0)/(3.14159265358979323846264338327
95)));//include cmath
m_19.Format("%g", 0.5);
m_19a.Format("%g",sp*0.681818182);
m_20.Format("%g",(sp*0.681818182)/(sqrt(((lwt+(0.3*4.0*uavprojectedarea))/4.0)/3.1416) * 0.5));
m_21.Format("%g",(sp*0.681818182)/0.5);
m_22.Format("%g",((al*0.3048)/1000));
double local= (((al*0.3048)/1000)*6356 / (6356+((al*0.3048)/1000)) );
m_23.Format("%g",local); // Z*E/Z+E 6356 is earth's radius
if(local < 44.3308)
//kg/m3 to lb/ft3 multiply by 0.0624
m_24.Format("%g",0.0624*(pow(((44.3308-local)/42.2665),4.2556)));
else
{
m_24="Out-of-range altitude";
}
m_25.Format("%g",fabs(pow(((44.3308-local)/4.94654),5.2556)));
m_26.Format("%ld",4);
m_27.Format("%g",0.23);
double solidity=fabs(((0.3*4.0*uavprojectedarea) + lwt)/(0.23*(pow((44.3308-local)/42.2665,4.2556))
* ((lwt+(0.3*4.0*uavprojectedarea))/4.0) * pow(((sp*0.681818182)/0.5),2)));
m_28.Format("%g",solidity);
UpdateData(FALSE);


//Filling the values
```

```
	ARD_method=(lwt+(0.3*4.0*uavprojectedarea))/4.0;
	TS_method=(sp*0.681818182)/0.5;
	RR_method=sqrt(((lwt+(0.3*4.0*uavprojectedarea))/4.0)/(3.14159265358979323846264338327795));
	SR_method=solidity;
	return TRUE;  // return TRUE unless you set the focus to a control
		// EXCEPTION: OCX Property Pages should return FALSE
}

void C7_1Dlg::On7b()
{
	EndDialog(IDCANCEL);
	CDatabaseSearchDLG a;
	a.DoModal();
}

void C7_1Dlg::On7h1()
{
	MessageBox("Empty weight cost is a commonly used metric in the aviation industry because it tends to
	remain constant across a variety of aircraft types. That number today is roughly $1500 per pound. Also,
	historically it is presumed that it costs around $8000 per pound of payload capacity. The two different
	values are ranged to form max and min limit.","HOW?");
}

void C7_1Dlg::On7h2()
{
	MessageBox("A link between performance and cost in terms of dollars per pound-hour has been
	framed. The in-air cost of UAV is calculated as = US$ (120K + 334*(Payload*Endurance))","HOW?");
}

void C7_1Dlg::On7h3()
{
	MessageBox("Useful load (in pounds) = Payload+(0.5*Engine-Power(shp))","HOW?");
}

void C7_1Dlg::On7h4()
{
	MessageBox("Fuel=sfc(lb/hp/hr) * engine-power(hp) * Mission-time(hour); where sfc = 0.5 for piston
	engines and 0.4 for turbine engines","HOW?");
}

void C7_1Dlg::On7h5()
{
	MessageBox("Reserve-fuel=sfc(lb/hp/hr) * engine-power(hp) * Reserve-time(hour); where sfc = 0.5 for
	piston engines and 0.4 for turbine engines","HOW?");
}

void C7_1Dlg::On7h6()
{
	MessageBox("A guide to suitable petrol/oil ratios for engines is hard to  formulate but as a world-wide
	standard, a ratio of 50:1 is considered healthy. Usually, highly developed engines with maximum
	engine speeds in excess of 7500rpm have ratio of 20:1 and modified engines with maximum engine
	speed less than 7500 rpm have ratio closed at 35:1. Considering these propositions, the ratio of 24:1
	seems fair.","HOW?");
}

void C7_1Dlg::On7h7()
{
	MessageBox("Initial guess at engine weight is from following equation: Engine-weight ~= 0.15*Gross-
	weight","HOW?");
}

void C7_1Dlg::On7h8()
{
```

```
	MessageBox("The projected area is evaluated from the plan view of the RWUAV. The plan view is
	divided into sectors and the areas of these sectors are evaluated geometrically. The aggregate of these
	areas is the projected area of the RWUAV, which will be presented in the profile diagram of the
	aircraft.","HOW?");
}

void C7_1Dlg::On7h9()
{
	MessageBox("A first estimate of downwash is obtained by assuming a drag coefficient of 0.3 for all
	RWUAV components in the remote wake.","HOW?");
}

void C7_1Dlg::On7h10()
{
	MessageBox("Disc Loading ranges from 4.0 to 12.0 (lb/ft*ft). As the constraint of the design process is
	to offer maximum hovering capabilities, disc loading is chosen to be at the lower end of the scale. Other
	advantages include low induced velocity and low autorotative rate of descent.","HOW?");
}

void C7_1Dlg::On7h11()
{
	MessageBox("Vertical Drag = Drag-Coefficient x Disk-Loading x Projected-Area","HOW?");
}

void C7_1Dlg::On7h12()
{
	MessageBox("The design chosen for RWUAV has a counter-coaxial rotor system. Both rotors will be
	of same shape and size, and will be rotating at same speed but in opposite direction. The advantages
	offered by this system are numerous. First, it is compact and saves space. Second, torque reaction is
	cancelled hence reducing the complexity in design calculations. Third, it allows for greater lifting
	capability as all the power is used by main rotor system. Fourth, it allows RWUAV to have a smaller
	fuselage and hence a narrower profile. Finally, lack of airframe structure necessary for the tail rotor
	system allows for greater payload capability with a low noise signature.","WHY?");
}

void C7_1Dlg::On7h13()
{
	MessageBox("DISC AREA = (GROSS WEIGHT + VERTICAL DRAG)/ DISC
	LOADING","HOW?");
}

void C7_1Dlg::On7h14()
{
	MessageBox("DISC AREA = Pi x R x R","HOW?");
}

void C7_1Dlg::On7h15()
{
	MessageBox("TIP SPEED of conventional helicopters range from 500 to 800 ft/sec and the 'tip speed
	ratio' at max forward speed does not exceed 0.5","HOW?");
}

void C7_1Dlg::On7h16()
{
	MessageBox("ROTATIONAL SPEED = FORWARD SPEED/(TIP RATIO * ROTOR
	RADIUS)","HOW?");
}

void C7_1Dlg::On7h17()
{
	MessageBox("TIP SPEED = ROTATIONAL SPEED * ROTOR RADIUS","HOW?");
}
```

```
void C7_1Dlg::On7h19()
{
	MessageBox(" Z=(E*H)/(E-H), where E = 6356  km, the radius of the earth (for 1976 ISA), H =
	geopotential altitude in km, & Z = geometric altitude, km","HOW?");
}

void C7_1Dlg::On7h20()
{
	MessageBox("D = POW((44.3308-H)/42.2665, 4.2556) * 0.0624, where   H = geopotential altitude in
	km and D = air density, lb/ft3","HOW?");
}

void C7_1Dlg::On7h21()
{
	MessageBox("P = POW((44.3308-H)/4.94654, 5.2556), where   H = geopotential altitude in km and P =
	actual air pressure, Pascals","HOW?");
}

void C7_1Dlg::On7h22()
{
	MessageBox("The radius-to-chord (R/C) ratio governs the number of blades.  R/C ratios of less than
	eight results in high tip losses, affecting the performance.  R/C ratios of above 24 provide structural
	flexibility being easy to bend and twist in flight.  At the preliminary design stage an aspect ratio of
	twelve is considered to evaluate the number of blades.  Alternatively, as number of blades is not a major
	design decision; four blades are generally accepted as conventional, for further design
	analysis.","HOW?");
}

void C7_1Dlg::On7h23()
{
	MessageBox("The number of blades is evaluated from the 'thrust coefficient'-to-'solidity ratio'.  The
	ratio ranges from 0.17 at sea level and varies with altitude to a maximum of 0.23. Since, we are
	performing analysis at max altitude, therefore max value is assumed.","HOW?");
}

void C7_1Dlg::On7h24()
{
	MessageBox("SOLIDITY=(GROSS-WEIGHT + VERTICAL DRAG)/0.23 * AIR-DENSITY *
	ROTOR-DISC-AREA * TIP-SPEED * TIP-SPEED","HOW?");
}

void C7_1Dlg::On7H6a()
{
	MessageBox("Each fuel system must be constructed and arranged to ensure a flow of fuel at a rate and
	pressure established for proper engine functioning under any likely operating condition, including the
	maneuvers for which certification is requested. The component recommendation is as follow:  1.  Each
	fuel tank must be able to withstand, without failure, the vibration, inertia, fluid, and structural loads to
	which it may be subjected in operation. 2. The maximum exposed surface temperature of any
	component in the fuel tank must be less, by a safe margin, than the lowest expected auto-ignition
	temperature of the fuel or fuel vapor in the tank.  3.  At least one-half inch of clear airspace must be
	provided between the tank and the firewall. 4. Each fuel tank must be supported so that loads resulting
	from the weight of fuel are not concentrated on unsupported tank skin. 5. Each compartment containing
	a fuel tank must be ventilated and drained overboard. 6. Each fuel tank, if permanently installed, must
	have a drainable sump which is effective in all normal ground and flight attitudes and with a capacity of
	0.10% of the tank capacity, or 120 ml, whichever is the greater. 7. Each fuel tank must be vented from
	the top of the tank. 8. There must be a fuel filter between the fuel tank outlet and the fuel pump inlet. 9.
	Each fuel line must be installed and supported to prevent excessive vibration and to withstand loads due
	to fuel pressure and accelerated flight conditions. 10. Each fuel drain must: (a) Be easily accessible; (b)
	Discharge clear of the aircraft; and (c) Have an automatic means for positive locking in the closed
	position.","FUEL SYSTEM DESIGN");
}

void C7_1Dlg::On7n()
```

```
{
	EndDialog(IDCANCEL);
	C7_2Dlg a;
	a.DoModal();

}
```

H.6 ‘Rotor system-II’ Class

The ‘Rotor system-II’ class is programmed to involve the selection of blade airfoil as follows:

```
// 7_2Dlg.cpp : implementation file
// Declaration of header files
#include "stdafx.h"
#include "VTUAV_Expert_System.h"
#include "7_2Dlg.h"
#include "7_1Dlg.h"
#include "7_3Dlg.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

// Declaration & initialisation of class variables
C7_2Dlg::C7_2Dlg(CWnd* pParent /*=NULL*/)
	: CDialog(C7_2Dlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(C7_2Dlg)
	m_edit1 = _T("");
	m_radio1 = 0;
	m_radio3 = 1;
	m_edit2 = _T("");
	//}}AFX_DATA_INIT
}

// Limiting variables values
void C7_2Dlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(C7_2Dlg)
	DDX_Control(pDX, IDC_EDIT2, m_edit2c);
	DDX_Control(pDX, IDC_EDIT1, m_editc);
	DDX_Control(pDX, IDC_SLIDER3, m_slider3);
	DDX_Control(pDX, IDC_SLIDER2, m_slider2);
	DDX_Control(pDX, IDC_SLIDER1, m_slider1);
	DDX_Text(pDX, IDC_EDIT1, m_edit1);
	DDX_Radio(pDX, IDC_RADIO1, m_radio1);
	DDX_Control(pDX, IDC_PICTURECLIP00, m_pic);
	DDX_Control(pDX, IDC_PICTURECLIP04, m_4);
	DDX_Control(pDX, IDC_PICTURECLIP06, m_6);
	DDX_Control(pDX, IDC_PICTURECLIP08, m_8);
	DDX_Control(pDX, IDC_PICTURECLIP20, m_20);
	DDX_Control(pDX, IDC_PICTURECLIP22, m_22);
	DDX_Control(pDX, IDC_PICTURECLIP24, m_24);
	DDX_Control(pDX, IDC_PICTURECLIP26, m_26);
	DDX_Control(pDX, IDC_PICTURECLIP28, m_28);
	DDX_Control(pDX, IDC_PICTURECLIP40, m_40);
	DDX_Control(pDX, IDC_PICTURECLIP42, m_42);
```

```
	DDX_Control(pDX, IDC_PICTURECLIP44, m_44);
	DDX_Control(pDX, IDC_PICTURECLIP46, m_46);
	DDX_Control(pDX, IDC_PICTURECLIP48, m_48);
	DDX_Control(pDX, IDC_PICTURECLIP60, m_60);
	DDX_Control(pDX, IDC_PICTURECLIP62, m_62);
	DDX_Control(pDX, IDC_PICTURECLIP64, m_64);
	DDX_Control(pDX, IDC_PICTURECLIP66, m_66);
	DDX_Control(pDX, IDC_PICTURECLIP68, m_68);
	DDX_Control(pDX, IDC_PICTURECLIP80, m_80);
	DDX_Control(pDX, IDC_PICTURECLIP82, m_82);
	DDX_Control(pDX, IDC_PICTURECLIP84, m_84);
	DDX_Control(pDX, IDC_PICTURECLIP86, m_86);
	DDX_Control(pDX, IDC_PICTURECLIP8812, m_88);
	DDX_Control(pDX, IDC_PICTURECLIP02, m_2);
	DDX_Radio(pDX, IDC_RADIO3, m_radio3);
	DDX_Text(pDX, IDC_EDIT2, m_edit2);
	//}}AFX_DATA_MAP
}

// Defining user-actions vis-a-vis system messages
BEGIN_MESSAGE_MAP(C7_2Dlg, CDialog)
	//{{AFX_MSG_MAP(C7_2Dlg)
	ON_NOTIFY(NM_RELEASEDCAPTURE, IDC_SLIDER1, OnReleasedcaptureSlider1)
	ON_NOTIFY(NM_RELEASEDCAPTURE, IDC_SLIDER2, OnReleasedcaptureSlider2)
	ON_NOTIFY(NM_RELEASEDCAPTURE, IDC_SLIDER3, OnReleasedcaptureSlider3)
	ON_BN_CLICKED(IDC_RADIO1, OnRadio1)
	ON_BN_CLICKED(IDC_RADIO2, OnRadio2)
	ON_BN_CLICKED(ID7M, On7m)
	ON_BN_CLICKED(ID7M2, On7m2)
	ON_BN_CLICKED(IDC_RADIO3, OnRadio3)
	ON_BN_CLICKED(IDC_RADIO4, OnRadio4)
	ON_BN_CLICKED(ID7_2B, On7_2b)
	ON_BN_CLICKED(ID7_2N, On2n)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

// Class method – Initialisation of dialog
BOOL C7_2Dlg::OnInitDialog()
{
	CDialog::OnInitDialog();
	// TODO: Add extra initialization here
	m_slider1.EnableWindow(TRUE);
	m_slider1.SetRange(0,8,FALSE);
	m_slider1.SetPos(0);
	m_slider1.SetTicFreq(2);
	m_slider2.EnableWindow(TRUE);
	m_slider2.SetRange(0,8,FALSE);
	m_slider2.SetPos(0);
	m_slider2.SetTicFreq(2);
	m_slider3.EnableWindow(TRUE);
	m_slider3.SetRange(0,98,FALSE);
	m_slider3.SetPos(12);
	m_slider3.SetTicFreq(2);
	UpdateData(TRUE);

	if(m_radio1==0)
	m_edit1= "NACA 0012 seems to be a good choice with camber value = 0, maximum camber position =
	0 and thickness value = 12.";
	else
	m_edit1= "Assymmetrical NACA combinations can be explored/selected by moving the sliders.
	However the third/thickness slider is disabled as thickness value of 12 is highly recommended as it has
	been thoroughly investigated and is extensively used." ;
```

```
	if(m_radio3==0)
	m_edit2="SQUARE TIP - easy to design, easy to manufacture and is a much cheaper option.";
	else
	m_edit2="SWEPTBACK TIPS - produce dynamic twist that improves the unbalanced air loading
	distribution in forward flight; uploaded tip on retreating side increases the nose down twist on that side
	that is beneficial for hover performances; finally the swept-back blade tip configuration tends to offer
	VTUAV speed stability.";

	//Show pic
	m_pic.EnableWindow(TRUE);
	m_pic.ShowWindow(SW_SHOW);

	//Disable sliders
	m_slider1.EnableWindow(FALSE);
	m_slider2.EnableWindow(FALSE);
	m_slider3.EnableWindow(FALSE);

	UpdateData(FALSE);
	return TRUE;  // return TRUE unless you set the focus to a control
		// EXCEPTION: OCX Property Pages should return FALSE
}


void C7_2Dlg::OnReleasedcaptureSlider1(NMHDR* pNMHDR, LRESULT* pResult)
{
	*pResult = 0;
	m_pic.ShowWindow(SW_SHOW);

	if(m_slider1.GetPos()==1)
	m_slider1.SetPos(m_slider1.GetPos()+1);
	if(m_slider1.GetPos()==3)
	m_slider1.SetPos(m_slider1.GetPos()+1);
	if(m_slider1.GetPos()==5)
	m_slider1.SetPos(m_slider1.GetPos()+1);
	if(m_slider1.GetPos()==7)
	m_slider1.SetPos(m_slider1.GetPos()+1);

	if(m_slider1.GetPos() == 0 && m_slider2.GetPos() == 2)
	{
		m_pic.ShowWindow(SW_HIDE);
		m_2.ShowWindow(SW_SHOW);
		m_4.ShowWindow(SW_HIDE);
		m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);
		m_22.ShowWindow(SW_HIDE);
		m_24.ShowWindow(SW_HIDE);
		m_26.ShowWindow(SW_HIDE);
		m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);
		m_42.ShowWindow(SW_HIDE);
		m_44.ShowWindow(SW_HIDE);
		m_46.ShowWindow(SW_HIDE);
		m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);
		m_62.ShowWindow(SW_HIDE);
		m_64.ShowWindow(SW_HIDE);
		m_66.ShowWindow(SW_HIDE);
		m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);
		m_82.ShowWindow(SW_HIDE);
		m_84.ShowWindow(SW_HIDE);
		m_86.ShowWindow(SW_HIDE);
```

```
		m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 0 && m_slider2.GetPos() == 4)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_SHOW); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);	m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);	m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);	m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 0 && m_slider2.GetPos() == 6)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_SHOW);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);	m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);	m_26.ShowWindow(SW_HIDE);
		m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);	m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 0 && m_slider2.GetPos() == 8)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_SHOW);
		m_20.ShowWindow(SW_HIDE);	m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);	m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);	m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 2 && m_slider2.GetPos() == 0)
{
```

```
            m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
            m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
            m_8.ShowWindow(SW_HIDE);
            m_20.ShowWindow(SW_SHOW); m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
            m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
            m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
            m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 2 && m_slider2.GetPos() == 2)
{
            m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
            m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
            m_8.ShowWindow(SW_HIDE);
            m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_SHOW);
m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
            m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
            m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
            m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 2 && m_slider2.GetPos() == 4)
{
            m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
            m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
            m_8.ShowWindow(SW_HIDE);
            m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_SHOW); m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
            m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
            m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
            m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 2 && m_slider2.GetPos() == 6)
{
            m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
            m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
            m_8.ShowWindow(SW_HIDE);
            m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_SHOW);
m_28.ShowWindow(SW_HIDE);
```

```
		m_40.ShowWindow(SW_HIDE);	m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 2 && m_slider2.GetPos() == 8)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);	m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);	m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_SHOW);
		m_40.ShowWindow(SW_HIDE);	m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 4 && m_slider2.GetPos() == 0)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);	m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);	m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_SHOW); m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 4 && m_slider2.GetPos() == 2)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);	m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);	m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);	m_42.ShowWindow(SW_SHOW);
m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
```

```
        m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 4 && m_slider2.GetPos() == 4)
{
        m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
        m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
        m_8.ShowWindow(SW_HIDE);
        m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
        m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_SHOW);           m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
        m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
        m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 4 && m_slider2.GetPos() == 6)
{
        m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
        m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
        m_8.ShowWindow(SW_HIDE);
        m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
        m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_SHOW);
m_48.ShowWindow(SW_HIDE);
        m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
        m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 4 && m_slider2.GetPos() == 8)
{
        m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
        m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
        m_8.ShowWindow(SW_HIDE);
        m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
        m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_SHOW);
        m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
        m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}
```

```
if(m_slider1.GetPos() == 6 && m_slider2.GetPos() == 0)
{
	m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
	m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
	m_8.ShowWindow(SW_HIDE);
	m_20.ShowWindow(SW_HIDE);  m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);  m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
	m_40.ShowWindow(SW_HIDE);  m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);  m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
	m_60.ShowWindow(SW_SHOW); m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);  m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
	m_80.ShowWindow(SW_HIDE);  m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);  m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 6 && m_slider2.GetPos() == 2)
{
	m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
	m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
	m_8.ShowWindow(SW_HIDE);
	m_20.ShowWindow(SW_HIDE);  m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);  m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
	m_40.ShowWindow(SW_HIDE);  m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);  m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
	m_60.ShowWindow(SW_HIDE);  m_62.ShowWindow(SW_SHOW);
m_64.ShowWindow(SW_HIDE);  m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
	m_80.ShowWindow(SW_HIDE);  m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);  m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 6 && m_slider2.GetPos() == 4)
{
	m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
	m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
	m_8.ShowWindow(SW_HIDE);
	m_20.ShowWindow(SW_HIDE);  m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);  m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
	m_40.ShowWindow(SW_HIDE);  m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);  m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
	m_60.ShowWindow(SW_HIDE);  m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_SHOW); m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
	m_80.ShowWindow(SW_HIDE);  m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);  m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 6 && m_slider2.GetPos() == 6)
{
	m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
	m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
	m_8.ShowWindow(SW_HIDE);
```

```
            m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
            m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
            m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_SHOW);
m_68.ShowWindow(SW_HIDE);
            m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 6 && m_slider2.GetPos() == 8)
{
            m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
            m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
            m_8.ShowWindow(SW_HIDE);
            m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
            m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
            m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_SHOW);
            m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 8 && m_slider2.GetPos() == 0)
{
            m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
            m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
            m_8.ShowWindow(SW_HIDE);
            m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
            m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
            m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
            m_80.ShowWindow(SW_SHOW); m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 8 && m_slider2.GetPos() == 2)
{
            m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
            m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
            m_8.ShowWindow(SW_HIDE);
            m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
            m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
```

```
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_SHOW);
m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 8 && m_slider2.GetPos() == 4)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);	m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);	m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);	m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_SHOW); m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 8 && m_slider2.GetPos() == 6)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);	m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);	m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);	m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_SHOW);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 8 && m_slider2.GetPos() == 8)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);	m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);	m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);	m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_SHOW);
```

```
	}

}


void C7_2Dlg::OnReleasedcaptureSlider2(NMHDR* pNMHDR, LRESULT* pResult)
{
	*pResult = 0;
	m_pic.ShowWindow(SW_SHOW);

	if(m_slider2.GetPos()==1)
	m_slider2.SetPos(m_slider2.GetPos()+1);
	if(m_slider2.GetPos()==3)
	m_slider2.SetPos(m_slider2.GetPos()+1);
	if(m_slider2.GetPos()==5)
	m_slider2.SetPos(m_slider2.GetPos()+1);
	if(m_slider2.GetPos()==7)
	m_slider2.SetPos(m_slider2.GetPos()+1);

	if(m_slider1.GetPos() == 0 && m_slider2.GetPos() == 2)
	{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_SHOW);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
	m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
	m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
	m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
	m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
	m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
	m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
	m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
	m_88.ShowWindow(SW_HIDE);
	}

	if(m_slider1.GetPos() == 0 && m_slider2.GetPos() == 4)
	{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_SHOW); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
	m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
	m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
	m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
	m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
	m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
	m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
	m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
	m_88.ShowWindow(SW_HIDE);
	}

	if(m_slider1.GetPos() == 0 && m_slider2.GetPos() == 6)
	{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_SHOW);
		m_8.ShowWindow(SW_HIDE);
```

```
        m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
        m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
        m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
        m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 0 && m_slider2.GetPos() == 8)
{
        m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
        m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
        m_8.ShowWindow(SW_SHOW);
        m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
        m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
        m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
        m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 2 && m_slider2.GetPos() == 0)
{
        m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
        m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
        m_8.ShowWindow(SW_HIDE);
        m_20.ShowWindow(SW_SHOW); m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
        m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
        m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
        m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 2 && m_slider2.GetPos() == 2)
{
        m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
        m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
        m_8.ShowWindow(SW_HIDE);
        m_20.ShowWindow(SW_HIDE); m_22.ShowWindow(SW_SHOW);
m_24.ShowWindow(SW_HIDE); m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
        m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
```

```
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 2 && m_slider2.GetPos() == 4)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);	m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_SHOW); m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);	m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 2 && m_slider2.GetPos() == 6)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);	m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);	m_26.ShowWindow(SW_SHOW);
m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);	m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 2 && m_slider2.GetPos() == 8)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);	m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);	m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_SHOW);
		m_40.ShowWindow(SW_HIDE);	m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
```

```
}

if(m_slider1.GetPos() == 4 && m_slider2.GetPos() == 0)
{
        m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
        m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
        m_8.ShowWindow(SW_HIDE);
        m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
        m_40.ShowWindow(SW_SHOW); m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
        m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
        m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 4 && m_slider2.GetPos() == 2)
{
        m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
        m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
        m_8.ShowWindow(SW_HIDE);
        m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
        m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_SHOW);
m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
        m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
        m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 4 && m_slider2.GetPos() == 4)
{
        m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
        m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
        m_8.ShowWindow(SW_HIDE);
        m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
        m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_SHOW);           m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
        m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
        m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 4 && m_slider2.GetPos() == 6)
{
```

```
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_SHOW);
m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 4 && m_slider2.GetPos() == 8)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_SHOW);
		m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}


if(m_slider1.GetPos() == 6 && m_slider2.GetPos() == 0)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_SHOW); m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 6 && m_slider2.GetPos() == 2)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
```

```
            m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
            m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
            m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_SHOW);
m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
            m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 6 && m_slider2.GetPos() == 4)
{
            m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
            m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
            m_8.ShowWindow(SW_HIDE);
            m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
            m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
            m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_SHOW); m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
            m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 6 && m_slider2.GetPos() == 6)
{
            m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
            m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
            m_8.ShowWindow(SW_HIDE);
            m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
            m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
            m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_SHOW);
m_68.ShowWindow(SW_HIDE);
            m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 6 && m_slider2.GetPos() == 8)
{
            m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
            m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
            m_8.ShowWindow(SW_HIDE);
            m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
            m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
```

```
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_SHOW);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 8 && m_slider2.GetPos() == 0)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);	m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);	m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);	m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_SHOW); m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 8 && m_slider2.GetPos() == 2)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);	m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);	m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);	m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_SHOW);
m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 8 && m_slider2.GetPos() == 4)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);	m_22.ShowWindow(SW_HIDE);
m_24.ShowWindow(SW_HIDE);	m_26.ShowWindow(SW_HIDE);
m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);	m_42.ShowWindow(SW_HIDE);
m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_HIDE);
m_84.ShowWindow(SW_SHOW); m_86.ShowWindow(SW_HIDE);
m_88.ShowWindow(SW_HIDE);
```

```
	}

	if(m_slider1.GetPos() == 8 && m_slider2.GetPos() == 6)
	{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
	m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
	m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
	m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
	m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
	m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
	m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
	m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_SHOW);
	m_88.ShowWindow(SW_HIDE);
	}

	if(m_slider1.GetPos() == 8 && m_slider2.GetPos() == 8)
	{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
	m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
	m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
	m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
	m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
	m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
	m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
	m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
	m_88.ShowWindow(SW_SHOW);
	}

}



void C7_2Dlg::OnReleasedcaptureSlider3(NMHDR* pNMHDR, LRESULT* pResult)
{
	*pResult = 0;
	CString a;
	a.Format("%d",m_slider3.GetPos());
	MessageBox(a);
}

void C7_2Dlg::OnRadio1()
{
	m_pic.ShowWindow(SW_SHOW);
	m_slider1.SetPos(0);
	m_slider2.SetPos(0);
	m_slider1.EnableWindow(FALSE);
	m_slider2.EnableWindow(FALSE);
	m_editc.SetWindowText("NACA 0012 seems to be a good choice with camber value = 0, maximum
	camber position = 0 and thickness value = 12.");
}
```

```
void C7_2Dlg::OnRadio2()
{
	m_slider1.EnableWindow(TRUE);
	m_slider2.EnableWindow(TRUE);
	m_editc.SetWindowText("Asymmetrical NACA combinations can be explored/selected by moving the
	sliders. However the third/thickness slider is disabled as thickness value of 12 is highly recommended
	as it has been thoroughly investigated and is extensively used.");
}

void C7_2Dlg::On7m()
{
	MessageBox("Airfoil sections are of two basic types, symmetrical and nonsymmetrical. (A)
	Symmetrical airfoils have identical upper and lower surfaces. They are suited to rotary-wing
	applications because they have almost no center of pressure travel. Travel remains relatively constant
	under varying angles of attack, affording the best lift-drag ratios for the full range of velocities from
	rotor blade root to tip. However, the symmetrical airfoil produces less lift than a nonsymmetrical airfoil
	and also has relatively undesirable stall characteristics. The rotorcraft blade must adapt to a wide range
	of airspeeds and angles of attack during each revolution of the rotor. The symmetrical airfoil delivers
	acceptable performance under those alternating conditions. Other benefits are lower cost and ease of
	construction as compared to the nonsymmetrical airfoil. (B) Nonsymmetrical (cambered) airfoils may
	have a wide variety of upper and lower surface designs. Advantages of the nonsymmetrical airfoil are
	increased lift-drag ratios and more desirable stall characteristics.","AIRFOIL SECTION TYPES");
}

void C7_2Dlg::On7m2()
{
	MessageBox("Advanced airfoil design methodology including Navier-Stokes CFD codes to design
	multi-element airfoils with leading edge slots. The unique rotorcraft challenge is to achieve the high lift
	of a slotted airfoil to delay stall on the helicopter rotor retreating blade without incurring a significant
	drag penalty at the low angles of attack of the advancing blade in high-speed forward flight. Another
	high-lift airfoil approach is to introduce a variable geometry, such as a deployable leading edge slat or
	variable leading edge contour. Variable leading edge camber or droop would be especially beneficial
	since no drag penalty would be incurred for azimuthal locations operating at low lift, although such
	concepts pose demanding actuator and structural requirements. Finally, new approaches to active flow
	control based on low-mass flow, oscillatory blowing concepts have emerged.","Advanced Airfoil
	Design");
}

void C7_2Dlg::OnRadio3()
{
	m_edit2c.SetWindowText("SQUARE TIP - easy to design, easy to manufacture and is a much cheaper
	option.");
}

void C7_2Dlg::OnRadio4()
{
	m_edit2c.SetWindowText("SWEPTBACK TIPS - produce dynamic twist that improves the unbalanced
	air loading distribution in forward flight; uploaded tip on retreating side increases the nose down twist
	on that side that is beneficial for hover performances; finally the swept-back blade tip configuration
	tends to offer VTUAV speed stability.");
}

void C7_2Dlg::On7_2b()
{
	EndDialog(IDCANCEL);
	C7_1Dlg a;
	a.DoModal();
}

void C7_2Dlg::On2n()
{
	EndDialog(IDCANCEL);
	C7_3Dlg a;
```

```
	a.DoModal();
}
```

H.7 ‘Drag’ Class

The ‘Drag’ class is programmed to evaluate coefficients of parasite and profile drag as follows:

```
// 7_4Dlg.cpp : implementation file
// Declaration of header files
#include "stdafx.h"
#include "VTUAV_Expert_System.h"
#include "7_4Dlg.h"
#include "7_3Dlg.h"
#include "7_5Dlg.h"
#include <cmath>  // equivalent of math.h
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

// Declaration & initialisation of class variables
C7_4Dlg::C7_4Dlg(CWnd* pParent /*=NULL*/)
	: CDialog(C7_4Dlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(C7_4Dlg)
	m_14 = _T("");
	m_15 = _T("");
	m_1 = _T("");
	m_2 = _T("");
	m_3 = _T("");
	m_4 = _T("");
	m_5 = _T("");
	m_6 = _T("");
	m_7 = _T("");
	m_8 = _T("");
	m_9a = _T("");
	m_10a = _T("");
	m_11 = _T("");
	m_12 = _T("");
	m_13 = _T("");
	m_16 = _T("");
	m_17 = _T("");
	m_18 = _T("");
	m_19 = _T("");
	m_20 = _T("");
	m_21 = _T("");
	m_22 = _T("");
	m_23 = _T("");
	m_24 = _T("");
	m_25 = _T("");
	//}}AFX_DATA_INIT
}

// Limiting variables values
void C7_4Dlg::DoDataExchange(CDataExchange* pDX)
{
```

```
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(C7_4Dlg)
	DDX_Text(pDX, IDC_EDIT14, m_14);
	DDX_Text(pDX, IDC_EDIT15, m_15);
	DDX_Text(pDX, IDC_EDIT1, m_1);
	DDX_Text(pDX, IDC_EDIT2, m_2);
	DDX_Text(pDX, IDC_EDIT3, m_3);
	DDX_Text(pDX, IDC_EDIT4, m_4);
	DDX_Text(pDX, IDC_EDIT5, m_5);
	DDX_Text(pDX, IDC_EDIT6, m_6);
	DDX_Text(pDX, IDC_EDIT7, m_7);
	DDX_Text(pDX, IDC_EDIT8, m_8);
	DDX_Text(pDX, IDC_EDIT9a, m_9a);
	DDX_Text(pDX, IDC_EDIT10a, m_10a);
	DDX_Text(pDX, IDC_EDIT11, m_11);
	DDX_Text(pDX, IDC_EDIT12, m_12);
	DDX_Text(pDX, IDC_EDIT13, m_13);
	DDX_Text(pDX, IDC_EDIT16, m_16);
	DDX_Text(pDX, IDC_EDIT17, m_17);
	DDX_Text(pDX, IDC_EDIT18, m_18);
	DDX_Text(pDX, IDC_EDIT19, m_19);
	DDX_Text(pDX, IDC_EDIT20, m_20);
	DDX_Text(pDX, IDC_EDIT21, m_21);
	DDX_Text(pDX, IDC_EDIT22, m_22);
	DDX_Text(pDX, IDC_EDIT23, m_23);
	DDX_Text(pDX, IDC_EDIT24, m_24);
	DDX_Text(pDX, IDC_EDIT25, m_25);
	//}}AFX_DATA_MAP
}

// Defining user-actions vis-a-vis system messages
BEGIN_MESSAGE_MAP(C7_4Dlg, CDialog)
	//{{AFX_MSG_MAP(C7_4Dlg)
	ON_BN_CLICKED(ID7_3B, On3b)
	ON_BN_CLICKED(ID7H1, On7h1)
	ON_BN_CLICKED(ID7H11, On7h11)
	ON_BN_CLICKED(ID7H2, On7h2)
	ON_BN_CLICKED(ID7H3, On7h3)
	ON_BN_CLICKED(ID7H4a, On7H4a)
	ON_BN_CLICKED(ID7H5, On7h5)
	ON_BN_CLICKED(ID7H6, On7h6)
	ON_BN_CLICKED(ID7H7, On7h7)
	ON_BN_CLICKED(ID7H8, On7h8)
	ON_BN_CLICKED(ID7H9, On7h9)
	ON_BN_CLICKED(ID7H10, On7h10)
	ON_BN_CLICKED(ID7H12, On7h12)
	ON_BN_CLICKED(ID7H13, On7h13)
	ON_BN_CLICKED(ID7H14, On7h14)
	ON_BN_CLICKED(ID7H15, On7h15)
	ON_BN_CLICKED(ID7H16, On7h16)
	ON_BN_CLICKED(ID7H17, On7h17)
	ON_BN_CLICKED(ID7H18, On7h18)
	ON_BN_CLICKED(ID7H19, On7h19)
	ON_BN_CLICKED(ID7H20, On7h20)
	ON_BN_CLICKED(ID7H21, On7h21)
	ON_BN_CLICKED(ID7_3N, On3n)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

// Class method – Initialisation of dialog
BOOL C7_4Dlg::OnInitDialog()
{
	CDialog::OnInitDialog();
```

```
extern CString uavname_dbsearch;
extern CString uavmission_dbsearch;
extern CString uavengine_dbsearch;
extern double uavweight_dbsearch;
extern double uavpower_dbsearch;
extern double uavspeed_dbsearch;
extern double uavendurance_dbsearch;
extern double uavrange_dbsearch;
extern double uavaltitude_dbsearch;
extern double uavlength_dbsearch;
extern double uavwingspan_dbsearch;
extern double uavrotorspan_dbsearch;
extern double uavdiameter_dbsearch;
extern double ewt;//empty weight
extern double pwt;//payload weight
extern double lwt;//launch weight
extern double mt;//mission time
extern double fr;//fuel reserve
extern double en;//endurance
extern double sp;//speed
extern double al;//altitude
extern double le;//length
extern double sn;//span, infact width
extern double uavprojectedarea;// store the value

//Variables for calculating power
extern double DEN_method;//Density
extern double PRD_method;//profile drag
extern double PAD_method;//parasite drag

CString a;

m_1.Format("%g",al);
m_2.Format("%g",sp*1.46666667);//change mph to fps and 1 mile / hour = 1.46666667 foot / second
m_3.Format("%g",le);


//Calculating the air variables
//Local variable declaration
double TEMPSL=518.67;
double RHOSL=0.00237689;
double PRESSSL=2116.22;
double saTheta=1.0;
double saSigma=1.0;
double saDelta=1.0;

double h=al*1.46666667;//convert mph to fps
double v=sp;
double rl=le;

//program for calculation
if(h<232940)
{
        saTheta=1.434843-(h/337634);
        saSigma=pow(0.79899-(h/606330),11.20114);
        saDelta=pow(0.838263-(h/577922),12.20114);
}

if(h<167323)
{
        saTheta=0.939268;
        saSigma=0.00116533*exp((h-154200)/-25992);
```

```
		saDelta=0.00109456*exp((h-154200)/-25992);
}

if(h<154199)
{
		saTheta=0.482561+(h/337634);
		saSigma=pow(0.857003+(h/190115),-13.20114);
		saDelta=pow(0.898309+(h/181373),-12.20114);
}

if(h<104987)
{
		saTheta=0.682457+(h/945374);
		saSigma=pow(0.978261+(h/659515),-35.16319);
		saDelta=pow(0.988626+(h/652600),-34.16319);
}

if(h<65617)
{
		saTheta=0.751865;
		saSigma=0.297076*exp((36089-h)/20806);
		saDelta=0.223361*exp((36089-h)/20806);
}

if(h<36089)
{
		saTheta=1.0-(h/145442);
		saSigma=pow(1.0-(h/145442),4.255876);
		saDelta=pow(1.0-(h/145442),5.255876);
}

double tempVal=TEMPSL*saTheta;
double rhoVal=RHOSL*saSigma;
double pVal=PRESSSL*saDelta;
double soundVal=sqrt(1.4*1716.56*(tempVal));
double machVal=v/soundVal;
double qVal=0.7*pVal*machVal*machVal;
double viscVal=0.0226968*pow(tempVal,1.5)/((tempVal)+198.72)/1000000.0;
double reynolds=v*rl*rhoVal/viscVal;
double cfturb=0.455/pow((log(reynolds)/log(10)),2.58);
double cflam=1.328/sqrt(reynolds);

m_4.Format("%g",tempVal);
m_5.Format("%g",rhoVal);
m_6.Format("%g",pVal);
m_7.Format("%g",qVal);
m_8.Format("%g",soundVal);
m_9a.Format("%g",viscVal);
m_10a.Format("%g",machVal);
m_11.Format("%g",reynolds);
m_12.Format("%g",cflam);
m_13.Format("%g",cfturb);
m_14="0.00818";
m_15.Format("%g",0.00818*(pow((lwt-pwt),(2/3))));
m_16.Format("%g",1 + (1.5*pow(sn/le,1.5)) + (7*pow(sn/le,3)));

double f_value=0, PDE;
if(reynolds < 100000)
{
		m_17.Format("%g",1.3*pow(reynolds,-0.05));
		f_value=1.3*pow(reynolds,-0.05);
}
else
```

```
	{
		m_17.Format("%g",0.45*pow(log(reynolds),-2.50));
		f_value=0.45*pow(log(reynolds),-2.50);
	}

	PDE=1.15 * (1 + (1.5*pow(sn/le,1.5)) + (7*pow(sn/le,3))) * (f_value);
	m_18.Format("%g", PDE);
	m_19.Format("%g",(0.00818*(pow((lwt-pwt),(2/3))))+PDE);
	m_20.Format("%g",0.23);
	m_21.Format("%g",5.9);
	m_22.Format("%g",6*0.23/5.9);
	m_23.Format("%g",0.0087 + (0.0126*(6*0.23/5.9)) + (0.400*(6*0.23/5.9)*(6*0.23/5.9)) );
	m_24.Format("%g",0.010 + (0.3*(6*0.23/5.9)*(6*0.23/5.9)));
	m_25.Format("%g",((0.0087 + (0.0126*(6*0.23/5.9)) + (0.400*(6*0.23/5.9)*(6*0.23/5.9)))+(0.010 +
	(0.3*(6*0.23/5.9)*(6*0.23/5.9))))/2);
	UpdateData(FALSE);


	DEN_method=rhoVal;//Density
	PRD_method=((0.0087 + (0.0126*(6*0.23/5.9)) + (0.400*(6*0.23/5.9)*(6*0.23/5.9)))+(0.010 +
	(0.3*(6*0.23/5.9)*(6*0.23/5.9))))/2 ;//profile drag
	PAD_method=(0.00818*(pow((lwt-pwt),(2/3))))+PDE ;//parasite drag
	return TRUE;  // return TRUE unless you set the focus to a control
		// EXCEPTION: OCX Property Pages should return FALSE
}

// Class method – Navigate to previous dialog
void C7_4Dlg::On3b()
{
	EndDialog(IDCANCEL);
	C7_3Dlg a;
	a.DoModal();
}

void C7_4Dlg::On7h1()
{
	MessageBox("Temp=518.67*st;where st is computed from flying altitude. Also, Celsius= ((Rankine-
	491.67)*0.5555)+273.15 ","HOW?");
}

void C7_4Dlg::On7h11()
{
	MessageBox("COEFFICIENT-OF-PARASITE-DRAG-OF-CLEAN-VTUAV-WITH-INTERNAL-
	USEFUL-LOAD = k*pow((W-P),0.667); where 'k' is parasite_drag-gross_weight constant, 'W' is Gross
	weight and 'P' is mission system weight external  ","HOW?");
}

void C7_4Dlg::On7h2()
{
	MessageBox("Density=0.00237689*ss; where ss is computed from flying altitude","HOW?");
}

void C7_4Dlg::On7h3()
{
	MessageBox("Pressure=2116.22*sd; where sd is computed from flying altitude","HOW?");
}

void C7_4Dlg::On7H4a()
{
	MessageBox("Dynamic Pressure = 0.7 * Pressure * MACH_no * MACH_no","HOW?");
}

void C7_4Dlg::On7h5()
```

```
{
	MessageBox("Sound Speed=sqrt(1.4*1716.56*(Temp))","HOW?");
}

void C7_4Dlg::On7h6()
{
	MessageBox("Viscosity=(0.0226968*pow(tempVal,1.5))/(((tempVal)+198.72)/1000000.0)","HOW?");
}

void C7_4Dlg::On7h7()
{
	MessageBox("Mach Num=Speed/Sound_Speed","HOW?");
}

void C7_4Dlg::On7h8()
{
	MessageBox("Reynolds Num=Speed*Reference_Length *Air_Density/Air_Viscosity", "HOW?");
}

void C7_4Dlg::On7h9()
{
	MessageBox("Laminar Cf = 1.328/sqrt(Reynolds_num)","HOW?");
}

void C7_4Dlg::On7h10()
{
	MessageBox("Turbulent Cf = 0.455/pow((log(Reynolds_num)/log(10)),2.58)","HOW?");
}

void C7_4Dlg::On7h12()
{
	MessageBox("FORM FACTOR = 1 + (1.5*pow(diameter_external_load/length_external_load,1.5)) +
	(7*pow(diameter_external_load/length_external_load,3))","HOW?");
}

void C7_4Dlg::On7h13()
{
	MessageBox("IF(Reynolds_num < 100000) then Coefficient_Friction = 1.3*pow(Reynolds_num,-0.05)
	ELSE Coefficient_Friction= 0.45* pow (log (Reynolds_num), -2.50)","HOW?");
}

void C7_4Dlg::On7h14()
{
	MessageBox("COEFFICIENT OF PARASITE DRAG EXTERNAL = 1.15 * FORM-FACTOR *
	COEFFICIENT-OF-FRICTION","HOW?");
}

void C7_4Dlg::On7h15()
{
	MessageBox(" COEFFICIENT-OF-PARASITE-DRAG = COEFFICIENT-OF-PARASITE-DRAG-OF-
	CLEAN-VTUAV-WITH-INTERNAL-USEFUL-LOAD + COEFFICIENT-OF-PARASITE-DRAG-
	EXTERNAL ","HOW?");
}

void C7_4Dlg::On7h16()
{
	MessageBox("The number of blades is evaluated from the 'thrust coefficient'-to-'solidity ratio'.  The
	ratio ranges from 0.17 at sea level and varies with altitude to a maximum of 0.23. Since, we are
	performing analysis at max altitude, therefore max value is assumed.","HOW?");
}

void C7_4Dlg::On7h17()
{
```

```
	MessageBox("For conventional airfoils at low Mach number the slope of the lift curve is '5.73' per
	radian. It increases slightly with Mach number and may be assumed to be '5.9' for preliminary
	design","HOW?");
}

void C7_4Dlg::On7h18()
{
	MessageBox("AVERAGE ANGLE OF ATTACK = 6 * THRUST_COEFFICIENT-TO-
	SOLIDITY_RATIO / SLOPE_OF_LIFT_CURVE","HOW?");
}

void C7_4Dlg::On7h19()
{
	MessageBox("COEFFICIENT_OF_PROFILE_DRAG- METHOD 1 = 0.0087 +
	0.0126*(AVERAGE_ANGLE_OF_ATTACK) +
	0.400*(AVERAGE_ANGLE_OF_ATTACK)^2","HOW?");
}

void C7_4Dlg::On7h20()
{
	MessageBox("COEFFICIENT_OF_PROFILE_DRAG- METHOD 2 = 0.01 +
	0.3*(AVERAGE_ANGLE_OF_ATTACK)^2","HOW?");
}

void C7_4Dlg::On7h21()
{
	MessageBox("","HOW?");
}

void C7_4Dlg::On3n()
{
	EndDialog(IDCANCEL);
	C7_5Dlg a;
	a.DoModal();
}
```

H.8 ‘Power’, ‘Database-II’, & ‘Airframe-section weight’ Classes

The ‘Power’, ‘Database-II’, and ‘Airframe-section weight’ classes are programmed in a single dialog for evaluation of the power required at maximum speed to sustain forward flight; reselection of engine; and computation of structural weight of various airframe sections of a VTUAV as follows:

```
// 7_5Dlg.cpp : implementation file
// Declaration of header files

#include "stdafx.h"
#include "VTUAV_Expert_System.h"
#include "7_5Dlg.h"
#include "7_4Dlg.h"
#include "7_1Dlg.h"
#include "7_6Dlg.h"
#include <cmath> //equivalent of math.h
#include "_recordset.h"
#ifdef _DEBUG
#define new DEBUG_NEW
```

```
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

double check=0;
extern double check_final;
extern double uavpower_dbsearch;//reset the power value
extern double engine_weight;//recording the selected engine weight
extern double final_power;//record computed engine power
extern double extreme_result;

//variables for calculating power
extern double ARD_method;//Area of rotor disc
extern double DEN_method;//Density
extern double TS_method;//Tip speed
extern double RR_method;//Rotor radius
extern double SR_method;//Solidity of rotor
extern double PRD_method;//Profile drag
extern double PAD_method;//Parasite drag

extern double pwt;//Payload weight
extern double lwt;//Launch weight
extern double sp;//Forward speed
extern double le;// Fuselage length
extern double uavprojectedarea;// store the value
extern double fuel_system;//recording fuel weight

// Declaration & initialisation of class variables
C7_5Dlg::C7_5Dlg(CWnd* pParent /*=NULL*/)
	: CDialog(C7_5Dlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(C7_5Dlg)
	m_1 = _T("");
	m_2 = _T("");
	m_3 = _T("");
	m_4 = _T("");
	m_5 = _T("");
	m_6 = _T("");
	m_7 = _T("");
	m_8 = _T("");
	m_9 = _T("");
	m_11 = _T("");
	m_10 = _T("");
	m_12 = _T("");
	m_13 = _T("");
	m_14 = _T("");
	m_15 = _T("");
	m_16 = _T("");
	m_17 = _T("");
	m_18 = _T("");
	m_19 = _T("");
	m_20 = _T("");
	m_21 = _T("");
	m_22 = _T("");
	m_23 = _T("");
	m_24 = _T("");
	m_25 = _T("");
	//}}AFX_DATA_INIT
}

// Limiting variables values
void C7_5Dlg::DoDataExchange(CDataExchange* pDX)
{
```

```
CDialog::DoDataExchange(pDX);
//{{AFX_DATA_MAP(C7_5Dlg)
DDX_Control(pDX, ID7_3N, m_next);
DDX_Control(pDX, IDC_EDIT25, m_25c);
DDX_Control(pDX, IDC_EDIT24a, m_24c);
DDX_Control(pDX, IDC_EDIT23, m_23c);
DDX_Control(pDX, IDC_EDIT22a, m_22c);
DDX_Control(pDX, IDC_EDIT21, m_21c);
DDX_Control(pDX, IDC_EDIT20, m_20c);
DDX_Control(pDX, IDC_EDIT19a, m_19c);
DDX_Control(pDX, IDC_EDIT18, m_18c);
DDX_Control(pDX, IDC_EDIT17, m_17c);
DDX_Control(pDX, IDC_EDIT16, m_16c);
DDX_Control(pDX, IDC_EDIT15, m_15c);
DDX_Control(pDX, IDC_EDIT14, m_14c);
DDX_Control(pDX, IDC_BUTTON2, m_but2);
DDX_Control(pDX, IDC_BUTTON1, m_but1);
DDX_Text(pDX, IDC_EDIT1, m_1);
DDX_Text(pDX, IDC_EDIT2, m_2);
DDX_Text(pDX, IDC_EDIT3, m_3);
DDX_Text(pDX, IDC_EDIT4a, m_4);
DDX_Text(pDX, IDC_EDIT5a, m_5);
DDX_Text(pDX, IDC_EDIT6a, m_6);
DDX_Text(pDX, IDC_EDIT7a, m_7);
DDX_Text(pDX, IDC_EDIT8a, m_8);
DDX_Text(pDX, IDC_EDIT9a, m_9);
DDX_Text(pDX, IDC_EDIT11a, m_11);
DDX_Text(pDX, IDC_EDIT10a, m_10);
DDX_Text(pDX, IDC_EDIT12, m_12);
DDX_Text(pDX, IDC_EDIT13, m_13);
DDX_Text(pDX, IDC_EDIT14, m_14);
DDX_Text(pDX, IDC_EDIT15, m_15);
DDX_Text(pDX, IDC_EDIT16, m_16);
DDX_Text(pDX, IDC_EDIT17, m_17);
DDX_Text(pDX, IDC_EDIT18, m_18);
DDX_Text(pDX, IDC_EDIT19a, m_19);
DDX_Text(pDX, IDC_EDIT20, m_20);
DDX_Text(pDX, IDC_EDIT21, m_21);
DDX_Text(pDX, IDC_EDIT22a, m_22);
DDX_Text(pDX, IDC_EDIT23, m_23);
DDX_Text(pDX, IDC_EDIT24a, m_24);
DDX_Text(pDX, IDC_EDIT25, m_25);
DDX_Control(pDX, IDC_LABEL1a, m_l1);
DDX_Control(pDX, IDC_LABEL1b, m_l2);
DDX_Control(pDX, IDC_LABEL1c, m_l3);
DDX_Control(pDX, IDC_LABEL1d, m_l4);
DDX_Control(pDX, IDC_LABEL1e, m_l5);
DDX_Control(pDX, IDC_LABEL1f, m_l6);
DDX_Control(pDX, IDC_LABEL1g, m_l7);
DDX_Control(pDX, IDC_LABEL1h, m_l8);
DDX_Control(pDX, IDC_LABEL1i, m_l9);
DDX_Control(pDX, IDC_LABEL1j, m_l10);
DDX_Control(pDX, IDC_LABEL1k, m_l11);
DDX_Control(pDX, IDC_LABEL1l, m_l12);
DDX_Control(pDX, IDC_ADOSEARCH, m_ado);
DDX_Control(pDX, IDC_ADOSEARCH1, m_ado1);
DDX_Control(pDX, IDC_DATAGRIDAS, m_dgas);
DDX_Control(pDX, IDC_DATAGRIDDS1, m_dgds);
DDX_Control(pDX, IDC_ADOSEARCH3, m_ado2);
DDX_Control(pDX, IDC_DATALISTID1, m_ls1);
DDX_Control(pDX, IDC_DATALISTID2, m_ls2);
DDX_Control(pDX, IDC_ADOSEARCH4, m_ado3);
//}}AFX_DATA_MAP
```

```
}

// Defining user-actions vis-a-vis system messages
BEGIN_MESSAGE_MAP(C7_5Dlg, CDialog)
	//{{AFX_MSG_MAP(C7_5Dlg)
	ON_BN_CLICKED(ID7H1, On7h1)
	ON_BN_CLICKED(ID7_3B, On3b)
	ON_BN_CLICKED(IDC_BUTTON1, OnButton1)
	ON_BN_CLICKED(IDC_BUTTON2, OnButton2)
	ON_BN_CLICKED(IDC_BUTTONengine, OnBUTTONengine)
	ON_BN_CLICKED(IDC_BUTTONengine1, OnBUTTONengine1)
	ON_BN_CLICKED(ID7_3N, On3n)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

// Class method – Initialisation of dialog
BOOL C7_5Dlg::OnInitDialog()
{
	CDialog::OnInitDialog();
	//Local variables converted into SI units
	double weight=lwt/2.2;//weight in kg
	double disc_area=ARD_method*0.09290304;//m^2 and 1(foot^2)/sec = 0.09290304(meter^2)/sec
	double density=DEN_method*536.523;// in kg/m^3 and 1 sl/ft3 = 536.523 kg/m3
	double tip_speed=TS_method*0.3048;// in m/sec and  1 foot / second = 0.3048 meter / second
	double forward_velocity=sp*0.44704;//in mps and 1 mile / hour = 0.44704 meter / second
	double rotor_radius=RR_method*0.3048;//in m and 1 foot = 0.3048 meters
	double solidity=SR_method;
	double profile_drag=PRD_method;
	double parasite_drag=PAD_method;

	m_1.Format("%g",weight);
	m_2.Format("%g",disc_area);
	m_3.Format("%g",density);
	m_4.Format("%g",tip_speed);
	m_5.Format("%g",forward_velocity);
	m_6.Format("%g",rotor_radius);
	m_7.Format("%g",solidity);
	m_8.Format("%g",profile_drag);
	m_9.Format("%g",parasite_drag);

	//Enter loop first time only
	if(extreme_result==0)
	{
	check=(weight*weight)/(2*disc_area*density*forward_velocity);//in KW
	check+=(solidity*profile_drag*density*disc_area*pow(tip_speed,2)*(1+(4.65*0.5*0.5)))/8;
	check=check*1.15;//miscellaneous power is 15% extra
	extreme_result=extreme_result + 1;
	}

	else
	{
	check=check_final;
	}

	CString Neetu, Neetu1;
	Neetu.Format("%g", check);
	//Value check-used in extreme cases
	if(check > 2.0*uavpower_dbsearch)
	{
		check=1.80*uavpower_dbsearch;
		Neetu1.Format("%g", check);
```

```
		MessageBox("Extreme case max algorithm followed. The program generates maximum power
		value of "+Neetu+" kw and in range value of "+Neetu1+" kw. The in-range value is being
		automatically selected.","BOUNDARY VALUES");
		check_final=check;
}

if(check < 0.5*uavpower_dbsearch)
{
		check=0.60*uavpower_dbsearch;
		Neetu1.Format("%g", check);
		MessageBox("Extreme case min algorithm followed. The program generates minimum power
		value of "+Neetu+" kw and in range value of "+Neetu1+" kw. The in-range value is being
		automatically selected.","BOUNDARY VALUES");
		check_final=check;
}

m_10.Format("%g",check);
check=check*1.341;//in hp and Power (HP) = Power (kW) x 1.341
m_11.Format("%g",check);
m_12.Format("%g",uavpower_dbsearch);
m_13.Format("%g",fabs((check)-uavpower_dbsearch));
final_power=check;


//make them invisible
m_14c.ShowWindow(SW_HIDE);
m_15c.ShowWindow(SW_HIDE);
m_16c.ShowWindow(SW_HIDE);
m_17c.ShowWindow(SW_HIDE);
m_18c.ShowWindow(SW_HIDE);
m_19c.ShowWindow(SW_HIDE);
m_20c.ShowWindow(SW_HIDE);
m_21c.ShowWindow(SW_HIDE);
m_22c.ShowWindow(SW_HIDE);
m_23c.ShowWindow(SW_HIDE);
m_24c.ShowWindow(SW_HIDE);
m_25c.ShowWindow(SW_HIDE);

//Weight distribution
double balance;
if((0.75 * 6.9 * pow((lwt/1000),0.49) * pow(le,0.61) * pow(uavprojectedarea,0.25))<lwt)
{
		m_14c.ShowWindow(SW_SHOW);
		m_14.Format("%g",0.75 * 6.9 * pow((lwt/1000),0.49) * pow(le,0.61) *
		pow(uavprojectedarea,0.25) );
		balance=lwt-((0.75 * 6.9 * pow((lwt/1000),0.49) * pow(le,0.61) *
		pow(uavprojectedarea,0.25)));
}

if((0.75 * 2 * 0.026 * pow(2*RR_method,0.66) * 0.4265 * pow(RR_method,1.3) *
pow(TS_method,0.67)) < balance)
{
		m_15c.ShowWindow(SW_SHOW);
		m_15.Format("%g", 0.75 * 2 * 0.026 * pow(2*RR_method,0.66) * 0.4265 *
		pow(RR_method,1.3) * pow(TS_method,0.67));//chord length is assumed at 0.13m or
		3.2808399 *.13 = 0.426509187 ft. Also rotor span is twice the radius
		balance-=0.75 * 2 * 0.026 * pow(2*RR_method,0.66) * 0.4265 * pow(RR_method,1.3) *
		pow(TS_method,0.67) ;
}

if(((0.75 * 0.041 * pow(engine_weight,1.1)) + (0.33*pow(uavprojectedarea,1.3)))<balance)
{
		m_16c.ShowWindow(SW_SHOW);
```

```
		m_16.Format("%g",(0.75 * 0.041 * pow(engine_weight,1.1)) +
		(0.33*pow(uavprojectedarea,1.3)));
		balance-=(0.75 * 0.041 * pow(engine_weight,1.1)) + (0.33*pow(uavprojectedarea,1.3));
}

if(engine_weight < balance)
{
		m_17c.ShowWindow(SW_SHOW);
		m_17.Format("%g",engine_weight);
		balance-=engine_weight;
}

if((0.5*13.6*pow((check),0.82) * pow(5.5,0.37) / pow((TS_method/RR_method),0.64)) <balance)
{
		m_18c.ShowWindow(SW_SHOW);
		m_18.Format("%g", 0.5*13.6*pow((check),0.82) * pow(5.5,0.37) /
		pow((TS_method/RR_method),0.64));//rpm assumed at 5500
		balance-=0.5*13.6*pow((check),0.82) * pow(5.5,0.37) / pow((TS_method/RR_method),0.64);
}

if((0.75*2*pow(engine_weight,0.59))<balance)
{
		m_19c.ShowWindow(SW_SHOW);
		m_19.Format("%g",0.75*2*pow(engine_weight,0.59));
		balance -=0.75*2*pow(engine_weight,0.59);
}

if(fuel_system<balance)
{
		m_20c.ShowWindow(SW_SHOW);
		m_20.Format("%g",fuel_system);
		balance-=fuel_system;
}

if((3.2*pow((check),0.65) / pow((lwt/1000),0.4))<balance)// formula is 9.6*pow((check),0.65) /
pow((lwt/1000),0.4) but i have changed it
{
		m_21c.ShowWindow(SW_SHOW);
		m_21.Format("%g",3.2*pow((check),0.65) / pow((lwt/1000),0.4));
		balance-=3.2*pow((check),0.65) / pow((lwt/1000),0.4);
}

if(pwt<balance)
{
		m_22c.ShowWindow(SW_SHOW);
		m_22.Format("%g",pwt);
		balance-=pwt;
}

UpdateData(FALSE);

if(((check)-uavpower_dbsearch)!=0)
{
		m_but1.ShowWindow(SW_SHOW);
		m_but2.ShowWindow(SW_HIDE);
}

else
{
		m_but1.ShowWindow(SW_HIDE);
		m_but2.ShowWindow(SW_SHOW);
}
```

```
	return TRUE;
}

void C7_5Dlg::On7h1()
{
	MessageBox("Power = (weight*weight)/(2*disc_area*density*forward_velocity) +
	((solidity*profile_drag*density*disc_area*pow(tip_speed,2)*(1+(4.65*0.5*0.5)))/8) +
	((density*parasite_drag*disc_area*pow(forward_velocity,3))/2)","How?");
}

void C7_5Dlg::On3b()
{
	EndDialog(IDCANCEL);
	C7_4Dlg a;
	a.DoModal();
}

void C7_5Dlg::OnButton1()
{
	MessageBox("The assumed power value and the computed power value are different. The methodology
	needs to be iterated. You will be automatically redirected to first methodology dialog. Please observe
	new values", "Iteration Required");
	uavpower_dbsearch = check;
	EndDialog(IDCANCEL);
	C7_1Dlg a;
	a.DoModal();
}

void C7_5Dlg::OnButton2()
{
	double val=check;
	CString query,query1,lwts;

	query = "SELECT * FROM UAVENGINELIST WHERE (POWER > ";
	lwts.Format("%g",val);
	query+=lwts;
	query+=") ORDER BY POWER, WEIGHT";

	query1= "SELECT * FROM UAVENGINELIST WHERE (POWER < ";
	query1+=lwts;
	query1+=") ORDER BY POWER DESC, WEIGHT";

	m_ado.SetRecordSource(query);
	m_ado.Refresh();
	m_ado1.SetRecordSource(query1);
	m_ado1.Refresh();
}

void C7_5Dlg::OnBUTTONengine()
{
	char  *stopstring;
	m_23c.ShowWindow(SW_SHOW);
	m_24c.ShowWindow(SW_SHOW);
	m_25c.ShowWindow(SW_SHOW);

	CString query_check;
	query_check = "SELECT * FROM UAVENGINELIST WHERE MODEL ='";
	query_check+=m_dgas.GetText();
	query_check+="'";
	m_ado1.SetRecordSource(query_check);
	m_ado1.Refresh();

	if(m_ado1.GetRecordset().GetRecordCount() > 0)
```

```
{
	m_23=m_dgas.GetText();
	query_check = "SELECT WEIGHT FROM UAVENGINELIST WHERE MODEL ='";
	query_check+=m_dgas.GetText();
	query_check+="'";
	m_ado1.SetRecordSource(query_check);
	m_ado1.Refresh();
	m_24=m_dgds.GetText();
	engine_weight= strtod(m_dgds.GetText(),&stopstring);

	query_check = "SELECT POWER FROM UAVENGINELIST WHERE MODEL ='";
	query_check+=m_dgas.GetText();
	query_check+="'";
	m_ado1.SetRecordSource(query_check);
	m_ado1.Refresh();
	m_25=m_dgds.GetText();

	//Name of closest uav
	query_check = "SELECT TOP 1 NAME FROM Querytable WHERE POWER >= ";
	query_check+= m_dgds.GetText();
	query_check+= " ORDER BY POWER";
	m_ado2.SetRecordSource(query_check);
	m_ado2.Refresh();

	query_check = "SELECT TOP 1 NAME FROM Querytable WHERE POWER < ";
	query_check+= m_dgds.GetText();
	query_check+= " ORDER BY POWER DESC";
	m_ado3.SetRecordSource(query_check);
	m_ado3.Refresh();

	//Resetting the other datagrid
	query_check="SELECT POWER FROM UAVENGINELIST WHERE MODEL = 'Ankush'";
	m_ado1.SetRecordSource(query_check);
	m_ado1.Refresh();
}

else
{
	MessageBox("Please select the name field only. You may have to click the search button once
	more.","Warning");
}


//2nd part - recalculate the weight values
//weight distribution
double balance;

if((0.75 * 6.9 * pow((lwt/1000),0.49) * pow(le,0.61) * pow(uavprojectedarea,0.25))<lwt)
{
//m_l1.ShowWindow(true);
	m_14c.ShowWindow(SW_SHOW);
	m_14.Format("%g",0.75 * 6.9 * pow((lwt/1000),0.49) * pow(le,0.61) *
	pow(uavprojectedarea,0.25) );
	balance=lwt-((0.75 * 6.9 * pow((lwt/1000),0.49) * pow(le,0.61) *
	pow(uavprojectedarea,0.25)));
}

if((0.75 * 2 * 0.026 * pow(2*RR_method,0.66) * 0.4265 * pow(RR_method,1.3) *
pow(TS_method,0.67)) < balance)
{
	//m_l2.ShowWindow(true);
	m_15c.ShowWindow(SW_SHOW);
```

```
		m_15.Format("%g", 0.75 * 2 * 0.026 * pow(2*RR_method,0.66) * 0.4265 *
		pow(RR_method,1.3) * pow(TS_method,0.67));//chord length is assumed at 0.13m or
		3.2808399 *.13 = 0.426509187 ft. Also rotor span is twice the radius
		balance-=0.75 * 2 * 0.026 * pow(2*RR_method,0.66) * 0.4265 * pow(RR_method,1.3) *
		pow(TS_method,0.67) ;
}

if(((0.75 * 0.041 * pow(engine_weight,1.1)) + (0.33*pow(uavprojectedarea,1.3)))<balance)
{
		//m_l3.ShowWindow(true);
		m_16c.ShowWindow(SW_SHOW);
		m_16.Format("%g",(0.75 * 0.041 * pow(engine_weight,1.1)) +
		(0.33*pow(uavprojectedarea,1.3)));
		balance-=(0.75 * 0.041 * pow(engine_weight,1.1)) + (0.33*pow(uavprojectedarea,1.3));
}

if(engine_weight < balance)
{
		//m_l4.ShowWindow(true);
		m_17c.ShowWindow(SW_SHOW);
		m_17.Format("%g",engine_weight);
		balance-=engine_weight;
}

if((0.5*13.6*pow((check),0.82) * pow(5.5,0.37) / pow((TS_method/RR_method),0.64))<balance)
{
		//m_l5.ShowWindow(true);
		m_18c.ShowWindow(SW_SHOW);
		m_18.Format("%g", 0.5*13.6*pow((check),0.82) * pow(5.5,0.37) /
		pow((TS_method/RR_method),0.64));//rpm assumed at 5500
		balance-=0.5*13.6*pow((check),0.82) * pow(5.5,0.37) / pow((TS_method/RR_method),0.64);
}

if((0.75*2*pow(engine_weight,0.59))<balance)
{
		//m_l6.ShowWindow(true);
		m_19c.ShowWindow(SW_SHOW);
		m_19.Format("%g",0.75*2*pow(engine_weight,0.59));
		balance-=0.75*2*pow(engine_weight,0.59);
}

if(fuel_system<balance)
{
		//m_l7.ShowWindow(true);
		m_20c.ShowWindow(SW_SHOW);
		m_20.Format("%g",fuel_system);
		balance-=fuel_system;
}


if((3.2*pow((check),0.65) / pow((lwt/1000),0.4))<balance)
{
		//m_l8.ShowWindow(true);
		m_21c.ShowWindow(SW_SHOW);
		m_21.Format("%g",3.2*pow((check),0.65) / pow((lwt/1000),0.4));
		balance-=3.2*pow((check),0.65) / pow((lwt/1000),0.4);
}

if(pwt<balance)
{
		//m_l9.ShowWindow(true);
		m_22c.ShowWindow(SW_SHOW);
		m_22.Format("%g",pwt);
```

```
			balance-=pwt;
		}

		//proceed further
		m_next.ShowWindow(SW_SHOW);
		MessageBox("Please Click on Next Button to determine C.G.","Proceed Further");
		UpdateData(FALSE);
}

void C7_5Dlg::OnBUTTONengine1()
{
		char  *stopstring;//pointer to end of number in a string
		m_23c.ShowWindow(SW_SHOW);
		m_24c.ShowWindow(SW_SHOW);
		m_25c.ShowWindow(SW_SHOW);

		CString query_check;
		query_check = "SELECT * FROM UAVENGINELIST WHERE MODEL ='";
		query_check+=m_dgds.GetText();
		query_check+="'";
		m_ado.SetRecordSource(query_check);
		 m_ado.Refresh();

		if(m_ado.GetRecordset().GetRecordCount() > 0)
		{
			m_23=m_dgds.GetText();
			query_check = "SELECT WEIGHT FROM UAVENGINELIST WHERE MODEL ='";
			query_check+=m_dgds.GetText();
			query_check+="'";
			m_ado.SetRecordSource(query_check);
			m_ado.Refresh();
			m_24=m_dgas.GetText();
			engine_weight= strtod(m_dgas.GetText(),&stopstring);

			query_check = "SELECT POWER FROM UAVENGINELIST WHERE MODEL ='";
			query_check+=m_dgds.GetText();
			query_check+="'";
			m_ado.SetRecordSource(query_check);
			m_ado.Refresh();
			m_25=m_dgas.GetText();

			//name of closest uav
			query_check = "SELECT TOP 1 NAME FROM Querytable WHERE POWER >= ";
			query_check+= m_dgas.GetText();
			query_check+= " ORDER BY POWER";
			m_ado2.SetRecordSource(query_check);
			m_ado2.Refresh();

			query_check = "SELECT TOP 1 NAME FROM Querytable WHERE POWER < ";
			query_check+= m_dgas.GetText();
			query_check+= " ORDER BY POWER DESC";
			m_ado3.SetRecordSource(query_check);
			m_ado3.Refresh();

			//Resetting the other datagrid
			query_check="SELECT POWER FROM UAVENGINELIST WHERE MODEL = 'Ankush'";
			m_ado.SetRecordSource(query_check);
			m_ado.Refresh();
		}

		else
		{
```

```
		MessageBox("Please select the name field only. You may have to click the search button once
		more.","Warning");
}


//2nd part - recalculate the weight values
//weight distribution
double balance;

if((0.75 * 6.9 * pow((lwt/1000),0.49) * pow(le,0.61) * pow(uavprojectedarea,0.25))<lwt)
{
		//m_l1.ShowWindow(true);
		m_14c.ShowWindow(SW_SHOW);
		m_14.Format("%g",0.75 * 6.9 * pow((lwt/1000),0.49) * pow(le,0.61) *
		pow(uavprojectedarea,0.25) );
		balance=lwt-((0.75 * 6.9 * pow((lwt/1000),0.49) * pow(le,0.61) *
		pow(uavprojectedarea,0.25)));
}

if((0.75 * 2 * 0.026 * pow(2*RR_method,0.66) * 0.4265 * pow(RR_method,1.3) *
pow(TS_method,0.67)) < balance)
{
		//m_l2.ShowWindow(true);
		m_15c.ShowWindow(SW_SHOW);
		m_15.Format("%g", 0.75 * 2 * 0.026 * pow(2*RR_method,0.66) * 0.4265 *
		pow(RR_method,1.3) * pow(TS_method,0.67));//chord length is assumed at 0.13m or
		3.2808399 *.13 = 0.426509187 ft. Also rotor span is twice the radius
		balance-=0.75 * 2 * 0.026 * pow(2*RR_method,0.66) * 0.4265 * pow(RR_method,1.3) *
		pow(TS_method,0.67) ;
}

if(((0.75 * 0.041 * pow(engine_weight,1.1)) + (0.33*pow(uavprojectedarea,1.3)))<balance)
{
		//m_l3.ShowWindow(true);
		m_16c.ShowWindow(SW_SHOW);
		m_16.Format("%g",(0.75 * 0.041 * pow(engine_weight,1.1)) +
		(0.33*pow(uavprojectedarea,1.3)));
		balance-=(0.75 * 0.041 * pow(engine_weight,1.1)) + (0.33*pow(uavprojectedarea,1.3));
}

if(engine_weight < balance)
{
		//m_l4.ShowWindow(true);
		m_17c.ShowWindow(SW_SHOW);
		m_17.Format("%g",engine_weight);
		balance-=engine_weight;
}

if((0.5*13.6*pow((check),0.82) * pow(5.5,0.37) / pow((TS_method/RR_method),0.64))<balance)
{
		//m_l5.ShowWindow(true);
		m_18c.ShowWindow(SW_SHOW);
		m_18.Format("%g", 0.5*13.6*pow((check),0.82) * pow(5.5,0.37) /
		pow((TS_method/RR_method),0.64));//rpm assumed at 5500
		balance-=0.5*13.6*pow((check),0.82) * pow(5.5,0.37) / pow((TS_method/RR_method),0.64);
}

if((0.75*2*pow(engine_weight,0.59))<balance)
{
		m_19c.ShowWindow(SW_SHOW);
		m_19.Format("%g",0.75*2*pow(engine_weight,0.59));
		balance-=0.75*2*pow(engine_weight,0.59);
}
```

```
	if(fuel_system<balance)
	{
		m_20c.ShowWindow(SW_SHOW);
		m_20.Format("%g",fuel_system);
		balance-=fuel_system;
	}

	if((3.2*pow((check),0.65) / pow((lwt/1000),0.4))<balance
	{
		m_21c.ShowWindow(SW_SHOW);
		m_21.Format("%g",3.2*pow((check),0.65) / pow((lwt/1000),0.4));
		balance-=3.2*pow((check),0.65) / pow((lwt/1000),0.4);
	}

	if(pwt<balance)
	{
		m_22c.ShowWindow(SW_SHOW);
		m_22.Format("%g",pwt);
		balance-=pwt;
	}

	//Proceed further
	m_next.ShowWindow(SW_SHOW);
	MessageBox("Please Click on Next Button to determine C.G.","Proceed Further");
	UpdateData(FALSE);
}

void C7_5Dlg::On3n()
{
	EndDialog(IDCANCEL);
	C7_6Dlg a;
	a.DoModal();
}
```

H.9 'Balancing' & 'Design-output' Classes

The 'Balancing', and 'Design-output' classes are programmed together and involve 'Weight and balance' exercise to refine the inboard and external profile of the VTUAV to position the cg within the acceptable limits as follows:

```
// 7_6Dlg.cpp : implementation file
// Declaration of header files
#include "stdafx.h"
#include "VTUAV_Expert_System.h"
#include "7_6Dlg.h"
#include "7_5Dlg.h"
#include "7_7Dlg.h"
#include "cmath"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

// Declaration & initialisation of class variables
extern double uavpower_dbsearch;//reset the power value
extern double engine_weight;//recording engine weight
```

```
extern double uavprojectedarea;// store the value
extern double fuel_system;//recording fuel weight

//Variables for calculating power
extern double ARD_method;//Area of rotor disc
extern double DEN_method;//Density
extern double TS_method;//Tip Speed
extern double RR_method;//Rotor radius
extern double SR_method;//Solidity of rotor
extern double PRD_method;//profile drag
extern double PAD_method;//parasite drag
extern double pwt;//payload weight
extern double lwt;//launch weight
extern double sp;//speed
extern double le;//length
extern double final_power;//record computed engine power
extern double wsn;//wing span
extern double SENSORARRAY[1];// record of individual weight
extern int i1;
extern double w1;
extern double TRANSPONDER[6];
extern int i2;
extern double w2;
extern double TRANSMITTER[2];
extern int i3;
extern double w3;
extern double TERMINAL[2];
extern int i4;
extern double w4;
extern double RECORDER[2];
extern int i5;
extern double w5;
extern double INTERCEPTOR[1];
extern int i6;
extern double w6;
extern double IMAGINGPROCESSOR[12];
extern int i7;
extern double w7;
extern double OBSERVATIONDISPLAYUNIT[5];
extern int i8;
extern double w8;
extern double CAMERAMODULARACCESSORIESSYSTEM[15];
extern int i9;
extern double w9;
extern double MODULARPAYLOAD[4];
extern int i10;
extern double w10;
extern double RWRESMSIGNITSYSTEM[5];
extern int i11;
extern double w11;
extern double RADARDECOYJAMMERS[2];
extern int i12;
extern double w12;
extern double FLIRS[7];
extern int i13;
extern double w13;
extern double ONBOARDRADAR[9];
extern int i14;
extern double w14;
double total_moment;
double total_weight;
double c;//c.g value
```

// Limiting variables values

```
C7_6Dlg::C7_6Dlg(CWnd* pParent /*=NULL*/)
	: CDialog(C7_6Dlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(C7_6Dlg)
	m_Ba = _T("");
	m_Ra = _T("");
	m_Da = _T("");
	m_Ma = _T("");
	m_Pa = _T("");
	m_Na = _T("");
	m_Ea = _T("");
	m_Fa = _T("");
	m_Bb = _T("");
	m_Bc = _T("");
	m_Db = _T("");
	m_Dc = _T("");
	m_Eb = _T("");
	m_Ec = _T("");
	m_Fb = _T("");
	m_Fc = _T("");
	m_Mb = _T("");
	m_Mc = _T("");
	m_Nb = _T("");
	m_Nc = _T("");
	m_Pb = _T("");
	m_Pc = _T("");
	m_Rb = _T("");
	m_Rc = _T("");
	m_M1 = _T("");
	m_M2 = _T("");
	m_W1 = _T("");
	m_W2 = _T("");
	m_CG1 = _T("");
	m_CG2 = _T("");
	m_V1 = _T("");
	m_V2 = _T("");
	m_san = _T("");
	m_saw = _T("");
	m_sas = _T("");
	m_sam = _T("");
	m_trm = _T("");
	m_trn = _T("");
	m_trs = _T("");
	m_trw = _T("");
	m_trnm = _T("");
	m_trnn = _T("");
	m_trns = _T("");
	m_trnw = _T("");
	m_tem = _T("");
	m_ten = _T("");
	m_tes = _T("");
	m_tew = _T("");
	m_rem = _T("");
	m_ren = _T("");
	m_res = _T("");
	m_rew = _T("");
	m_inm = _T("");
	m_inn = _T("");
	m_ins = _T("");
	m_inw = _T("");
	m_ipm = _T("");
	m_ipn = _T("");
```

```
	m_ips = _T("");
	m_ipw = _T("");
	m_oum = _T("");
	m_oun = _T("");
	m_ous = _T("");
	m_ouw = _T("");
	m_csm = _T("");
	m_csn = _T("");
	m_css = _T("");
	m_csw = _T("");
	m_mpn = _T("");
	m_mpm = _T("");
	m_mps = _T("");
	m_mpw = _T("");
	m_rsm = _T("");
	m_rsn = _T("");
	m_rss = _T("");
	m_rsw = _T("");
	m_rjm = _T("");
	m_rjn = _T("");
	m_rjs = _T("");
	m_rjw = _T("");
	m_flm = _T("");
	m_fln = _T("");
	m_fls = _T("");
	m_flw = _T("");
	m_ram = _T("");
	m_ran = _T("");
	m_ras = _T("");
	m_raw = _T("");
	m_M3 = _T("");
	m_W3 = _T("");
	m_radio = 0;
	//}}AFX_DATA_INIT
}

void C7_6Dlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(C7_6Dlg)
	DDX_Text(pDX, IDC_EDITBa, m_Ba);
	DDX_Text(pDX, IDC_EDITRa, m_Ra);
	DDX_Text(pDX, IDC_EDITDa, m_Da);
	DDX_Text(pDX, IDC_EDITMa, m_Ma);
	DDX_Text(pDX, IDC_EDITPa, m_Pa);
	DDX_Text(pDX, IDC_EDITNa, m_Na);
	DDX_Text(pDX, IDC_EDITEa, m_Ea);
	DDX_Text(pDX, IDC_EDITFa, m_Fa);
	DDX_Text(pDX, IDC_EDITBb, m_Bb);
	DDX_Text(pDX, IDC_EDITBc, m_Bc);
	DDX_Text(pDX, IDC_EDITDb, m_Db);
	DDX_Text(pDX, IDC_EDITDc, m_Dc);
	DDX_Text(pDX, IDC_EDITEb, m_Eb);
	DDX_Text(pDX, IDC_EDITEc, m_Ec);
	DDX_Text(pDX, IDC_EDITFb, m_Fb);
	DDX_Text(pDX, IDC_EDITFc, m_Fc);
	DDX_Text(pDX, IDC_EDITMb, m_Mb);
	DDX_Text(pDX, IDC_EDITMc, m_Mc);
	DDX_Text(pDX, IDC_EDITNb, m_Nb);
	DDX_Text(pDX, IDC_EDITNc, m_Nc);
	DDX_Text(pDX, IDC_EDITPb, m_Pb);
	DDX_Text(pDX, IDC_EDITPc, m_Pc);
	DDX_Text(pDX, IDC_EDITRb, m_Rb);
```

```
DDX_Text(pDX, IDC_EDITRc, m_Rc);
DDX_Text(pDX, IDC_EDITM1, m_M1);
DDX_Text(pDX, IDC_EDITM2, m_M2);
DDX_Text(pDX, IDC_EDITW1, m_W1);
DDX_Text(pDX, IDC_EDITW2, m_W2);
DDX_Text(pDX, IDC_EDITCG1, m_CG1);
DDX_Text(pDX, IDC_EDITCG2, m_CG2);
DDX_Text(pDX, IDC_EDITV1, m_V1);
DDX_Text(pDX, IDC_EDITV2, m_V2);
DDX_Text(pDX, IDC_EDITsan, m_san);
DDX_Text(pDX, IDC_EDITsaw, m_saw);
DDX_Text(pDX, IDC_EDITsas, m_sas);
DDX_Text(pDX, IDC_EDITsam, m_sam);
DDX_Text(pDX, IDC_EDITtrm, m_trm);
DDX_Text(pDX, IDC_EDITtrn, m_trn);
DDX_Text(pDX, IDC_EDITtrs, m_trs);
DDX_Text(pDX, IDC_EDITtrw, m_trw);
DDX_Text(pDX, IDC_EDITtrnm, m_trnm);
DDX_Text(pDX, IDC_EDITtrnn, m_trnn);
DDX_Text(pDX, IDC_EDITtrns, m_trns);
DDX_Text(pDX, IDC_EDITtrnw, m_trnw);
DDX_Text(pDX, IDC_EDITtem, m_tem);
DDX_Text(pDX, IDC_EDITten, m_ten);
DDX_Text(pDX, IDC_EDITtes, m_tes);
DDX_Text(pDX, IDC_EDITtew, m_tew);
DDX_Text(pDX, IDC_EDITrem, m_rem);
DDX_Text(pDX, IDC_EDITren, m_ren);
DDX_Text(pDX, IDC_EDITres, m_res);
DDX_Text(pDX, IDC_EDITrew, m_rew);
DDX_Text(pDX, IDC_EDITinm, m_inm);
DDX_Text(pDX, IDC_EDITinn, m_inn);
DDX_Text(pDX, IDC_EDITins, m_ins);
DDX_Text(pDX, IDC_EDITinw, m_inw);
DDX_Text(pDX, IDC_EDITipm, m_ipm);
DDX_Text(pDX, IDC_EDITipn, m_ipn);
DDX_Text(pDX, IDC_EDITips, m_ips);
DDX_Text(pDX, IDC_EDITipw, m_ipw);
DDX_Text(pDX, IDC_EDITоum, m_oum);
DDX_Text(pDX, IDC_EDITоun, m_oun);
DDX_Text(pDX, IDC_EDITоus, m_ous);
DDX_Text(pDX, IDC_EDITоuw, m_ouw);
DDX_Text(pDX, IDC_EDITcsm, m_csm);
DDX_Text(pDX, IDC_EDITcsn, m_csn);
DDX_Text(pDX, IDC_EDITcss, m_css);
DDX_Text(pDX, IDC_EDITcsw, m_csw);
DDX_Text(pDX, IDC_EDITmpn, m_mpn);
DDX_Text(pDX, IDC_EDITmpm, m_mpm);
DDX_Text(pDX, IDC_EDITmps, m_mps);
DDX_Text(pDX, IDC_EDITmpw, m_mpw);
DDX_Text(pDX, IDC_EDITrsm, m_rsm);
DDX_Text(pDX, IDC_EDITrsn, m_rsn);
DDX_Text(pDX, IDC_EDITrss, m_rss);
DDX_Text(pDX, IDC_EDITrsw, m_rsw);
DDX_Text(pDX, IDC_EDITrjm, m_rjm);
DDX_Text(pDX, IDC_EDITrjn, m_rjn);
DDX_Text(pDX, IDC_EDITrjs, m_rjs);
DDX_Text(pDX, IDC_EDITrjw, m_rjw);
DDX_Text(pDX, IDC_EDITflm, m_flm);
DDX_Text(pDX, IDC_EDITfln, m_fln);
DDX_Text(pDX, IDC_EDITfls, m_fls);
DDX_Text(pDX, IDC_EDITflw, m_flw);
DDX_Text(pDX, IDC_EDITram, m_ram);
DDX_Text(pDX, IDC_EDITran, m_ran);
```

```
	DDX_Text(pDX, IDC_EDITras, m_ras);
	DDX_Text(pDX, IDC_EDITraw, m_raw);
	DDX_Text(pDX, IDC_EDITM3, m_M3);
	DDX_Text(pDX, IDC_EDITW3, m_W3);
	DDX_Radio(pDX, IDC_RADIO1, m_radio);
	DDX_Control(pDX, IDC_PICTURECLIP1, m_pic1);
	DDX_Control(pDX, IDC_PICTURECLIP1a, m_pic2);
	//}}AFX_DATA_MAP
}

// Defining user-actions vis-a-vis system messages
BEGIN_MESSAGE_MAP(C7_6Dlg, CDialog)
	//{{AFX_MSG_MAP(C7_6Dlg)
	ON_BN_CLICKED(ID7H2, On7h2)
	ON_BN_CLICKED(ID7P, On7p)
	ON_BN_CLICKED(ID7_3B, On3b)
	ON_BN_CLICKED(IDC_RADIO2, OnRadio2)
	ON_BN_CLICKED(IDC_RADIO3, OnRadio3)
	ON_BN_CLICKED(IDC_RADIO1, OnRadio1)
	ON_BN_CLICKED(ID7F, On7f)
	ON_BN_CLICKED(IDC_RADIO4, OnRadio4)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()


void C7_6Dlg::On7h2()
{
	MessageBox("Payload Positioning Aims: 1) To distribute the load symmetrically around the center of
	gravity of the aircraft. 2)To mount the payload such that there are no large moment arms. 3)Payload to
	be placed on the underside of the fuselage, so the infra-red camera can be pointed down to view the
	target area. 4)Mounted such that vibrations absorbed by the Infra-red camera are minimized.","AIM");

}

void C7_6Dlg::On7p()
{
	MessageBox("Longitudinal Position of the center of gravity of the empty VTUAV is calculated from
	the sum of the static moments about some arbitrary point contributed by each group that makes up the
	empty weight divided by that weight. The arbitrary point should be selected ahead of and below the
	nose so that all parts of the aircraft will have positive locations. A variation under 7% is an acceptable
	result.","PROCEDURE");
}

// Class method – Initialisation of dialog
BOOL C7_6Dlg::OnInitDialog()
{
	CDialog::OnInitDialog();
	//C.G without payload - recalculate the weight values
	//weight distribution

	double balance, check;
	check=final_power;

	//weight var
	double body_weight=0, rotor_weight=0, nacelle_weight=0;
	double eng_weight=0, drysystem_weight=0, propulsionsubsys_weight=0;
	double fuel_weight=0, electricalsys_weight=0;

	//distance var
	double body_station=0, rotor_station=0, nacelle_station=0;
	double eng_station=0, drysystem_station=0, propulsionsubsys_station=0;
	double fuel_station=0, electricalsys_station=0;
```

```
if((0.75 * 6.9 * pow((lwt/1000),0.49) * pow(le,0.61) * pow(uavprojectedarea,0.25))<lwt)
{
	body_weight=0.75 * 6.9 * pow((lwt/1000),0.49) * pow(le,0.61) * pow(uavprojectedarea,0.25);
	balance=lwt-((0.75 * 6.9 * pow((lwt/1000),0.49) * pow(le,0.61) *
	pow(uavprojectedarea,0.25)));
}

if((0.75 * 2 * 0.026 * pow(2*RR_method,0.66) * 0.4265 * pow(RR_method,1.3) *
pow(TS_method,0.67)) < balance)
{
	rotor_weight = 0.75 * 2 * 0.026 * pow(2*RR_method,0.66) * 0.4265 * pow(RR_method,1.3)
	* pow(TS_method,0.67);//chord length is assumed at 0.13m or 3.2808399 *.13 = 0.426509187
	ft. Also rotor span is twice the radius
	balance-=0.75 * 2 * 0.026 * pow(2*RR_method,0.66) * 0.4265 * pow(RR_method,1.3) *
	pow(TS_method,0.67) ;
}

if(((0.75 * 0.041 * pow(engine_weight,1.1)) + (0.33*pow(uavprojectedarea,1.3)))<balance)
{
	nacelle_weight = (0.75 * 0.041 * pow(engine_weight,1.1)) +
	(0.33*pow(uavprojectedarea,1.3));
	balance-=(0.75 * 0.041 * pow(engine_weight,1.1)) + (0.33*pow(uavprojectedarea,1.3));
}

if(engine_weight < balance)
{
	eng_weight = engine_weight;
	balance-=engine_weight;
}

if((0.5*13.6*pow((check),0.82) * pow(5.5,0.37) / pow((TS_method/RR_method),0.64))<balance)
{
	drysystem_weight = 0.5*13.6*pow((check),0.82) * pow(5.5,0.37) /
	pow((TS_method/RR_method),0.64);//rpm assumed at 5500
	balance-=0.5*13.6*pow((check),0.82) * pow(5.5,0.37) / pow((TS_method/RR_method),0.64);
}

if((0.75*2*pow(engine_weight,0.59))<balance)
{
	propulsionsubsys_weight = 0.75*2*pow(engine_weight,0.59);
	balance-=0.75*2*pow(engine_weight,0.59);
}

if(fuel_system<balance)
{
	fuel_weight = fuel_system;
	balance-=fuel_system;
}

if((3.2*pow((check),0.65) / pow((lwt/1000),0.4))<balance)
{
	electricalsys_weight = 3.2*pow((check),0.65) / pow((lwt/1000),0.4);
	balance-=3.2*pow((check),0.65) / pow((lwt/1000),0.4);
}

//check if the leftover weight can be adjusted
if(balance>0)
{

	if(body_weight == 0)
	body_weight = balance;
```

```
		if(rotor_weight == 0)
		rotor_weight = balance;

		if(nacelle_weight == 0)
		nacelle_weight = balance;

		if(eng_weight == 0)
		eng_weight = balance;

		if(drysystem_weight == 0)
		drysystem_weight = balance;

		if(propulsionsubsys_weight == 0)
		propulsionsubsys_weight = balance;

		if(fuel_weight == 0)
		fuel_weight = balance;

		if(electricalsys_weight == 0)
		 electricalsys_weight = balance;
}

m_Ba.Format("%g", body_weight);
m_Ra.Format("%g", rotor_weight);
m_Da.Format("%g", drysystem_weight);
m_Ma.Format("%g", eng_weight);
m_Pa.Format("%g", propulsionsubsys_weight);
m_Na.Format("%g", nacelle_weight);
m_Fa.Format("%g", fuel_weight);
m_Ea.Format("%g", electricalsys_weight);

//determine the type of aircraft and find the cg of the body
//D'Gopher VTUAV
if(wsn>0)
{
		c= 2*le/3;
		body_station = c;
		rotor_station = 0.95*c;
		eng_station = 0.96*c;
		propulsionsubsys_station = 0.97*c;
		nacelle_station = 0.98*c;
		fuel_station = 0.88*c;
		drysystem_station = 1.20*c;
		electricalsys_station =1.15*c;
}

//SEAMOS VTUAV
else
{
		c= le/2;
		body_station = c;
		rotor_station = 0.95*c;
		eng_station = 0.96*c;
		propulsionsubsys_station = 0.97*c;
		nacelle_station = 0.98*c;
		fuel_station = 0.88*c;
		drysystem_station = 1.20*c;
		electricalsys_station =1.15*c;
}

m_Bb.Format("%g", body_station);
m_Rb.Format("%g", rotor_station);
m_Db.Format("%g", drysystem_station);
```

```
m_Mb.Format("%g", eng_station);
m_Pb.Format("%g", propulsionsubsys_station);
m_Nb.Format("%g", nacelle_station);
m_Fb.Format("%g", fuel_station);
m_Eb.Format("%g", electricalsys_station);

//calculate moment
m_Bc.Format("%g", body_weight * body_station);
m_Rc.Format("%g", rotor_weight * rotor_station);
m_Dc.Format("%g", drysystem_weight * drysystem_station);
m_Mc.Format("%g", eng_weight * eng_station);
m_Pc.Format("%g", propulsionsubsys_weight * propulsionsubsys_station);
m_Nc.Format("%g", nacelle_weight * nacelle_station);
m_Fc.Format("%g", fuel_weight * fuel_station);
m_Ec.Format("%g", electricalsys_weight * electricalsys_station);


total_moment=(body_weight * body_station) + (rotor_weight * rotor_station) + (drysystem_weight *
drysystem_station) + (eng_weight * eng_station) + (propulsionsubsys_weight *
propulsionsubsys_station) + (nacelle_weight * nacelle_station) + (fuel_weight * fuel_station) +
(electricalsys_weight * electricalsys_station);
total_weight=(body_weight) + (rotor_weight) + (drysystem_weight) + (eng_weight) +
(propulsionsubsys_weight) + (nacelle_weight) + (fuel_weight) + (electricalsys_weight);

m_M1.Format("%g",total_moment);
m_W1.Format("%g",total_weight);
m_CG1.Format("%g",total_moment/total_weight);
m_V1.Format("%g",fabs(  ((total_moment/total_weight)-(c)) * 100 / c ));

//C.G with payload
m_san.Format("%d",i1);
m_saw.Format("%g",w1);
m_sas.Format("%g",1.10*c);
m_sam.Format("%g", w1 * (1.10*c));
m_trn.Format("%d",i2);
m_trw.Format("%g",w2);
m_trs.Format("%g",1.30*c);
m_trm.Format("%g", w2 * (1.30*c));
m_trnn.Format("%d",i3);
m_trnw.Format("%g",w3);
m_trns.Format("%g",1.25*c);
m_trnm.Format("%g", w3 * (1.25*c));
m_ten.Format("%d",i4);
m_tew.Format("%g",w4);
m_tes.Format("%g",1.20*c);
m_tem.Format("%g", w4 * (1.20*c));
m_ren.Format("%d",i5);
m_rew.Format("%g",w5);
m_res.Format("%g",1.15*c);
m_rem.Format("%g", w5 * (1.15*c));
m_inn.Format("%d",i6);
m_inw.Format("%g",w6);
m_ins.Format("%g",1.03*c);
m_inm.Format("%g", w6 * (1.03*c));
m_ipn.Format("%d",i7);
m_ipw.Format("%g",w7);
m_ips.Format("%g",0.99*c);
m_ipm.Format("%g", w7 * (0.99*c));
m_oun.Format("%d",i8);
m_ouw.Format("%g",w8);
m_ous.Format("%g",0.94*c);
m_oum.Format("%g", w8 * (0.94*c));
m_csn.Format("%d",i9);
```

```
		m_csw.Format("%g",w9);
		m_css.Format("%g",0.88*c);
		m_csm.Format("%g", w9 * (0.88*c));
		m_mpn.Format("%d",i10);
		m_mpw.Format("%g",w10);
		m_mps.Format("%g",1.22*c);
		m_mpm.Format("%g", w10 * (1.22*c));
		m_rsn.Format("%d",i11);
		m_rsw.Format("%g",w11);
		m_rss.Format("%g",0.85*c);
		m_rsm.Format("%g", w11 * (0.85*c));
		m_rjn.Format("%d",i12);
		m_rjw.Format("%g",w12);
		m_rjs.Format("%g",c);
		m_rjm.Format("%g", w12 * (c));
		m_fln.Format("%d",i13);
		m_flw.Format("%g",w13);
		m_fls.Format("%g",1.05*c);
		m_flm.Format("%g", w13 * (1.05*c));
		m_ran.Format("%d",i14);
		m_raw.Format("%g",w14);
		m_ras.Format("%g",c);
		m_ram.Format("%g", w14 * (c));

		//finally
		double payload_moment, payload_weight, final_moment, final_weight;
		payload_moment= (w1 * (1.10*c)) + (w2 * (1.30*c)) + (w3 * (1.25*c)) + (w4 * (1.20*c)) + (w5 *
		(1.15*c)) + (w6 * (1.03*c)) + (w7 * (0.99*c)) + (w8 * (0.94*c)) + (w9 * (0.88*c)) +(w10 * (1.22*c))
		+(w11 * (0.85*c)) +(w12 * (c)) +(w13 * (1.05*c)) +(w14 * (c))  ;
		payload_weight= w1+w2+w3+w4+w5+w6+w7+w8+w9+w10+w11+w12+w13+w14;
		final_moment=payload_moment + total_moment ;
		final_weight=payload_weight + total_weight ;
		m_M2.Format("%g",payload_moment);
		m_W2.Format("%g",payload_weight);
		m_M3.Format("%g",final_moment);
		m_W3.Format("%g",final_weight);
		m_CG2.Format("%g",final_moment/final_weight);
		m_V2.Format("%g",fabs(  ((final_moment/final_weight)-(c)) * 100 / c ));

		//show the graph
		m_pic1.ShowWindow(SW_SHOW);
		UpdateData(FALSE);
		return TRUE;
}

void C7_6Dlg::On3b()
{
		EndDialog(IDCANCEL);
		C7_5Dlg a;
		a.DoModal();
}

void C7_6Dlg::OnRadio1()
{
		m_pic1.ShowWindow(SW_SHOW);
		m_pic2.ShowWindow(SW_HIDE);
}


void C7_6Dlg::OnRadio2()
{
		m_pic2.ShowWindow(SW_SHOW);
		m_pic1.ShowWindow(SW_HIDE);
```

```
}

void C7_6Dlg::OnRadio3()
{
	m_pic2.ShowWindow(SW_HIDE);
	m_pic1.ShowWindow(SW_HIDE);

	//Generate the profile
	//Method 2 -  Dont have to capture and release DC - Device Context
	CClientDC dlgDC(this);
	CRect rc;
	GetClientRect(&rc);
	CPen penBlue(PS_DASHDOTDOT, 1, RGB(0,0,255));
	CPen *pOldPen = NULL;
	pOldPen = dlgDC.SelectObject(&penBlue);
	CBrush brGrey(RGB(128,128,128));
	CBrush *pOldBrush=NULL;
	pOldBrush=dlgDC.SelectObject(&brGrey);

	if(wsn>0)
	{
		dlgDC.MoveTo(410+170,570);
		dlgDC.LineTo(655+170,670);
		dlgDC.LineTo(755+170,670);
		dlgDC.LineTo(755+170,470);
		dlgDC.LineTo(655+170,470);
		dlgDC.LineTo(410+170,570);
		dlgDC.LineTo(430+170,530);
		dlgDC.LineTo(675+170,633);
		dlgDC.LineTo(655+170,670);
		dlgDC.LineTo(675+170,633);
		dlgDC.LineTo(775+170,633);
		dlgDC.LineTo(755+170,670);
		dlgDC.LineTo(775+170,633);
		dlgDC.LineTo(775+170,433);
		dlgDC.LineTo(755+170,470);
		dlgDC.LineTo(775+170,433);
		dlgDC.LineTo(675+170,433);
		dlgDC.LineTo(655+170,470);
		dlgDC.LineTo(675+170,433);
		dlgDC.LineTo(675+170,633);
		dlgDC.LineTo(675+170,433);
		dlgDC.LineTo(430+170,530);
		dlgDC.Chord(402+170,390,753+170,683,382+170,575,687+170,720);
		dlgDC.AngleArc(609+170,633,206,74,80);
		dlgDC.TextOut(660+170,530,"cg");
		dlgDC.TextOut(560,690,"LAYOUT OF COUNTER ROTATING VTUAV W/O ROTOR
		SYSTEM");

		dlgDC.Ellipse(410+85-20,485-40,410+85+30-20,515-40);
		dlgDC.MoveTo(410+85-20,570-40);
		CRect a;
		a.TopLeft().x=410+85-20;
		a.TopLeft().y=600-40;
		a.BottomRight().x=410+85+30-20;
		a.BottomRight().y=500-40;
		dlgDC.Draw3dRect(a,0x00FF1122,0x00FFFFFF);
		//first rotor
		CPoint x1,y1,z1;
		x1.x=510-20;
		x1.y=445;
		y1.x=615-20;
		y1.y=385;
```

```
z1.x=595-20;
z1.y=365;

CPoint point[]={x1,y1,z1};
dlgDC.Polygon(point,3);
x1.x=510-20;
x1.y=445;
y1.x=435-20;
y1.y=525;
z1.x=415-20;
z1.y=505;
CPoint point1[]={x1,y1,z1};
dlgDC.Polygon(point1,3);
x1.x=510-20;
x1.y=445;
y1.x=435-20;
y1.y=360;
z1.x=415-20;
z1.y=380;
CPoint point2[]={x1,y1,z1};
dlgDC.Polygon(point2,3);
x1.x=510-20;
x1.y=445;
y1.x=615-20;
y1.y=505;
z1.x=595-20;
z1.y=525;
CPoint point3[]={x1,y1,z1};
dlgDC.Polygon(point3,3);

//2nd rotor
x1.x=510-20;
x1.y=445+30;
y1.x=615-20;
y1.y=385+30;
z1.x=595-20;
z1.y=365+30;

CPoint point4[]={x1,y1,z1};
dlgDC.Polygon(point4,3);
x1.x=510-20;
x1.y=445+30;
y1.x=435-20;
y1.y=525+30;
z1.x=415-20;
z1.y=505+30;

CPoint point5[]={x1,y1,z1};
dlgDC.Polygon(point5,3);
x1.x=510-20;
x1.y=445+30;
y1.x=435-20;
y1.y=360+30;
z1.x=415-20;
z1.y=380+30;

CPoint point6[]={x1,y1,z1};
dlgDC.Polygon(point6,3);
x1.x=510-20;
x1.y=445+30;
y1.x=615-20;
y1.y=505+30;
z1.x=595-20;
```

```
		z1.y=525+30;

		CPoint point7[]={x1,y1,z1};
		dlgDC.Polygon(point7,3);
		dlgDC.TextOut(405,590,"COUNTER ROTATING");
		dlgDC.TextOut(415,610,"ROTOR SYSTEM");
}

else
{
		dlgDC.MoveTo(695,570);
		dlgDC.AngleArc(695,570,80,90,180);
		dlgDC.LineTo(695,490);
		dlgDC.LineTo(855,490);
		dlgDC.LineTo(855,650);
		dlgDC.LineTo(695,650);
		dlgDC.MoveTo(855,570);
		dlgDC.AngleArc(855,570,80,270,180);

		dlgDC.MoveTo(695+10,570-25);
		dlgDC.AngleArc(695+10,570-25,80,90,180);
		dlgDC.LineTo(695+10,490-25);
		dlgDC.LineTo(855+10,490-25);
		dlgDC.LineTo(855+10,650-25);
		dlgDC.LineTo(695+10,650-25);
		dlgDC.MoveTo(855+10,570-25);
		dlgDC.AngleArc(855+10,570-25,80,270,180);

		dlgDC.MoveTo(695,490);
		dlgDC.LineTo(695+10,490-25);
		dlgDC.MoveTo(855,490);
		dlgDC.LineTo(855+10,490-25);
		dlgDC.MoveTo(855,650);
		dlgDC.LineTo(855+10,650-25);
		dlgDC.MoveTo(695,650);
		dlgDC.LineTo(695+10,650-25);

		dlgDC.TextOut(560,690,"LAYOUT OF COUNTER ROTATING VTUAV W/O ROTOR
		SYSTEM");

		dlgDC.Ellipse(410+85-20,485-40,410+85+30-20,515-40);
		dlgDC.MoveTo(410+85-20,570-40);
		CRect a;
		a.TopLeft().x=410+85-20;
		a.TopLeft().y=600-40;
		a.BottomRight().x=410+85+30-20;
		a.BottomRight().y=500-40;

		dlgDC.Draw3dRect(a,0x00FF1122,0x00FFFFFF);

		//first rotor
		CPoint x1,y1,z1;
		x1.x=510-20;
		x1.y=445;
		y1.x=615-20;
		y1.y=385;
		z1.x=595-20;
		z1.y=365;

		CPoint point[]={x1,y1,z1};
		dlgDC.Polygon(point,3);

		x1.x=510-20;
```

```
x1.y=445;
y1.x=435-20;
y1.y=525;
z1.x=415-20;
z1.y=505;

CPoint point1[]={x1,y1,z1};
dlgDC.Polygon(point1,3);
x1.x=510-20;
x1.y=445;
y1.x=435-20;
y1.y=360;
z1.x=415-20;
z1.y=380;

CPoint point2[]={x1,y1,z1};
dlgDC.Polygon(point2,3);
x1.x=510-20;
x1.y=445;
y1.x=615-20;
y1.y=505;
z1.x=595-20;
z1.y=525;

CPoint point3[]={x1,y1,z1};
dlgDC.Polygon(point3,3);

//2nd rotor
x1.x=510-20;
x1.y=445+30;
y1.x=615-20;
y1.y=385+30;
z1.x=595-20;
z1.y=365+30;

CPoint point4[]={x1,y1,z1};
dlgDC.Polygon(point4,3);

x1.x=510-20;
x1.y=445+30;
y1.x=435-20;
y1.y=525+30;
z1.x=415-20;
z1.y=505+30;

CPoint point5[]={x1,y1,z1};
dlgDC.Polygon(point5,3);

x1.x=510-20;
x1.y=445+30;
y1.x=435-20;
y1.y=360+30;
z1.x=415-20;
z1.y=380+30;

CPoint point6[]={x1,y1,z1};
dlgDC.Polygon(point6,3);
x1.x=510-20;
x1.y=445+30;
y1.x=615-20;
y1.y=505+30;
z1.x=595-20;
z1.y=525+30;
```

```
			CPoint point7[]={x1,y1,z1};
			dlgDC.Polygon(point7,3);

			dlgDC.TextOut(405,590,"COUNTER ROTATING");
			dlgDC.TextOut(415,610,"ROTOR SYSTEM");
		}
}

void C7_6Dlg::On7f()
{
		UpdateData(TRUE);

		char  *stopstring;//pointer to end of number in a string
		m_sam.Format("%g", w1*strtod(m_sas,&stopstring));
		m_trm.Format("%g", w2*strtod(m_trs,&stopstring));
		m_trnm.Format("%g", w3*strtod(m_trns,&stopstring));
		m_tem.Format("%g", w4*strtod(m_tes,&stopstring));
		m_rem.Format("%g", w5*strtod(m_res,&stopstring));
		m_inm.Format("%g", w6*strtod(m_ins,&stopstring));
		m_ipm.Format("%g", w7*strtod(m_ips,&stopstring));
		m_oum.Format("%g", w8*strtod(m_ous,&stopstring));
		m_csm.Format("%g", w9*strtod(m_css,&stopstring));
		m_mpm.Format("%g", w10*strtod(m_mps,&stopstring));
		m_rsm.Format("%g", w11*strtod(m_rss,&stopstring));
		m_rjm.Format("%g", w12*strtod(m_rjs,&stopstring));
		m_flm.Format("%g", w13*strtod(m_fls,&stopstring));
		m_ram.Format("%g", w14*strtod(m_ras,&stopstring));

		// finally
		double payload_moment, payload_weight, final_moment, final_weight;
		payload_moment = (w1 * strtod(m_sas,&stopstring)) + (w2 * strtod(m_trs,&stopstring))
			+ (w3 * strtod(m_trns,&stopstring)) + (w4 * strtod(m_tes,&stopstring))
			+ (w5 * strtod(m_res,&stopstring)) + (w6 * strtod(m_ins,&stopstring))
			+ (w7 * strtod(m_ips,&stopstring)) + (w8 * strtod(m_ous,&stopstring))
			+ (w9 * strtod(m_css,&stopstring)) + (w10 * strtod(m_mps,&stopstring))
			+ (w11 * strtod(m_rss,&stopstring)) + (w12 * strtod(m_rjs,&stopstring))
			+(w13 * strtod(m_fls,&stopstring)) + (w14 * strtod(m_ras,&stopstring));

		payload_weight= w1+w2+w3+w4+w5+w6+w7+w8+w9+w10+w11+w12+w13+w14;
		final_moment=payload_moment + strtod(m_M1,&stopstring) ;
		final_weight=payload_weight + strtod(m_W1,&stopstring) ;
		m_M2.Format("%g",payload_moment);
		m_W2.Format("%g",payload_weight);
		m_M3.Format("%g",final_moment);
		m_W3.Format("%g",final_weight);
		m_CG2.Format("%g",final_moment/final_weight);
		m_V2.Format("%g",fabs(  ((final_moment/final_weight)-(c)) * 100 / c ));
		UpdateData(FALSE);

		if(fabs(((final_moment/final_weight)-(c)) * 100 / c ) > 7)
		MessageBox("Since the variation is >7%, hence the solution may not be acceptable.");
}

void C7_6Dlg::OnRadio4()
{
		C7_7Dlg a;
		a.DoModal();
}
```

H.10 ‘Non-editable database’ & ‘Editable database’ Classes

The ‘Non-editable database’ and ‘Editable database’ classes provide read-only/update facilities of the design parameters of various VTUAVs in-development/in-service and their power-plants as follows:

```
// BviDlg.cpp : implementation file
// Declaration of header files
#include "stdafx.h"
#include "VTUAV_Expert_System.h"
#include "BviDlg.h"
#include "column.h"
#include "columns.h"
#include "datagrid.h"
#include "_recordset.h"
#include "FindBVIDlg.h"
#include "Find1BVIDlg.h"
#include "Find2BVIDlg.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

int return1;
int return2;

// Declaration & initialisation of class variables
CBviDlg::CBviDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CBviDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CBviDlg)
	m_editva = _T("");
	m_editva1 = _T("");
	m_e5str = _T("");
	m_editva2 = _T("");
	m_e4str = _T("");
	//}}AFX_DATA_INIT
}

// Limiting variables values
void CBviDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CBviDlg)
	DDX_Control(pDX, IDC_EDIT3, m_3c);
	DDX_Control(pDX, IDC_EDIT5, m_e5c);
	DDX_Control(pDX, IDC_BUTTON1, m_ltmt);
	DDX_Control(pDX, IDC_EDIT4, m_c);
	DDX_Control(pDX, IDC_BUTTONFIND, m_find);
	DDX_Control(pDX, IDC_EDIT1, m_editco);
	DDX_Control(pDX, IDC_ADOVTUAVMAIN, m_adovtmain);
	DDX_Control(pDX, IDC_DATAGRID1, m_dg);
	DDX_Text(pDX, IDC_EDIT1, m_editva);
	DDX_Text(pDX, IDC_EDIT2, m_editva1);
	DDX_Text(pDX, IDC_EDIT5, m_e5str);
	DDX_Text(pDX, IDC_EDIT3, m_editva2);
	DDX_Text(pDX, IDC_EDIT4, m_e4str);
	//}}AFX_DATA_MAP
}
```

```
// Defining user-actions vis-a-vis system messages
BEGIN_MESSAGE_MAP(CBviDlg, CDialog)
	//{{AFX_MSG_MAP(CBviDlg)
	ON_BN_CLICKED(IDC_BUTTONFIND, OnButtonfind)
	ON_EN_CHANGE(IDC_EDIT4, OnChangeEdit4)
	ON_BN_CLICKED(IDC_BUTTON1, OnButton1)
	ON_EN_CHANGE(IDC_EDIT5, OnChangeEdit5)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

BEGIN_EVENTSINK_MAP(CBviDlg, CDialog)
	//{{AFX_EVENTSINK_MAP(CBviDlg)
	ON_EVENT(CBviDlg, IDC_DATAGRID1, 218 /* RowColChange */, OnRowColChangeDatagrid1,
	VTS_PVARIANT VTS_I2)
	//}}AFX_EVENTSINK_MAP
END_EVENTSINK_MAP()

// Class method – Initialisation of dialog
BOOL CBviDlg::OnInitDialog()
{
	CDialog::OnInitDialog();
	m_adovtmain.GetRecordset().MoveLast();
	m_adovtmain.GetRecordset().MoveFirst();
	m_find.SetWindowText("Find");

	CEdit* ed = (CEdit*)GetDlgItem(IDC_EDIT1);
	CEdit*ed1 = (CEdit*)GetDlgItem(IDC_EDIT2);
	long num1 =  m_dg.GetColumns().GetCount();

	if(num1<10)
	{
		ed->SetWindowText(CString(num1+48));
	}

	else
	{
		int quo = num1/10;
		int rem = num1%10;
		ed->SetWindowText(CString(quo+48) + CString(rem+48));
	}
		long num2 = m_adovtmain.GetRecordset().GetRecordCount();
	if (num2<10)
	{
		ed1->SetWindowText(CString(num2+48));
	}


	else if(num2<100)
	{
		int quo = num2/10;
		int rem = num2%10;
		ed1->SetWindowText(CString(quo+48) + CString(rem+48));
	}

	else if(num2<1000)
	{
		int quo = num2/10;
		int quo1 = quo/10;
		int quo2 = quo%10;
		int rem = num2%10;
		ed1->SetWindowText(CString(quo1+48) + CString(quo2+48) + CString(rem+48));
	}
```

```
else
{
	int quo = num2/100;
	int quo1 = quo/10;
	int quo2 = quo%10;
	int quoa = num2%100;
	int quo3 = quoa/10;
	int rem = num2%10;
	ed1->SetWindowText(CString(quo1+48) + CString(quo2+48) + CString(quo3+48) +
	CString(rem+48));
}

CEdit*ed2 = (CEdit*)GetDlgItem(IDC_EDIT3);
long num3 = m_adovtmain.GetRecordset().GetAbsolutePosition();

if (num3<10)
{
	ed2->SetWindowText(CString(num3+48));
}
else if(num3<100)
{
	int quo = num3/10;
	int rem = num3%10;
	ed2->SetWindowText(CString(quo+48) + CString(rem+48));
}
else if(num3<1000)
{
	int quo = num3/10;
	int quo1 = quo/10;
	int quo2 = quo%10;
	int rem = num3%10;
	ed2->SetWindowText(CString(quo1+48) + CString(quo2+48) + CString(rem+48));
}
else
{
	int quo = num3/100;
	int quo1 = quo/10;
	int quo2 = quo%10;
	int quoa = num3%100;
	int quo3 = quoa/10;
	int rem = num3%10;
	ed2->SetWindowText(CString(quo1+48) + CString(quo2+48) + CString(quo3+48) +
	CString(rem+48));

}

//set the properties
m_dg.SetAllowUpdate(FALSE);
m_dg.SetAllowAddNew(FALSE);
m_dg.SetAllowDelete(FALSE);

extern int modify_check;
//add new
if(modify_check == 1)
{
	m_dg.SetAllowUpdate(TRUE); //this is essential to set the add new property
	m_dg.SetAllowAddNew(TRUE);
	m_adovtmain.GetRecordset().SetAbsolutePosition(m_adovtmain.GetRecordset().GetRecordC
	ount());
	UpdateData(FALSE);
}
```

```
	//update
	if(modify_check == 2)
	{
		m_dg.SetAllowUpdate(TRUE); //only update the existing record values
	}

	//delete
	if(modify_check == 3)
	{
		m_dg.SetAllowDelete(TRUE); //delete the record
	}
		return TRUE;  // return TRUE unless you set the focus to a control

}

void CBviDlg::OnRowColChangeDatagrid1(VARIANT FAR* LastRow, short LastCol)
{
	CEdit*ed2 = (CEdit*)GetDlgItem(IDC_EDIT3);//ed2 should have been named as edrow
	long num3 = m_adovtmain.GetRecordset().GetAbsolutePosition();
	if (num3<10)
	{
		ed2->SetWindowText(CString(num3+48));
	}

	else if(num3<100)
	{
		int quo = num3/10;
		int rem = num3%10;
		ed2->SetWindowText(CString(quo+48) + CString(rem+48));
	}

	else if(num3<1000)
	{
		int quo = num3/10;
		int quo1 = quo/10;
		int quo2 = quo%10;
		int rem = num3%10;
		ed2->SetWindowText(CString(quo1+48) + CString(quo2+48) + CString(rem+48));
	}

	else
	{
		int quo = num3/100;
		int quo1 = quo/10;
		int quo2 = quo%10;
		int quoa = num3%100;
		int quo3 = quoa/10;
		int rem = num3%10;
		ed2->SetWindowText(CString(quo1+48) + CString(quo2+48) + CString(quo3+48) +
		CString(rem+48));
	}

}

// Class method – Search function
void CBviDlg::OnButtonfind()
{
	m_e5c.ShowWindow(SW_HIDE);
	m_c.ShowWindow(SW_HIDE);
	m_ltmt.ShowWindow(SW_SHOW);
	m_find.EnableWindow(FALSE);
	CFindBVIDlg a;
	a.DoModal();
```

```
	if(a.result_findbvidlg  ==  1)
	{
		m_e5c.ShowWindow(SW_SHOW);
		Find1BVIDlg a;
		if (a.DoModal() == IDOK)
		return1 = 1;
		else return1 = 0;
	}

	if(a.result_findbvidlg == 2)
	{
		m_c.ShowWindow(SW_SHOW);
		CFind2BVIDlg a;
		if (a.DoModal() == IDOK){
		return2 = 1;
	 }

	else
	{
		return2 = 0;
	}
}

void CBviDlg::OnChangeEdit4()
{
	m_find.EnableWindow(FALSE);
	UpdateData(TRUE);
	CString x ;
	CString y;
	y = m_e4str;

	if(return2 == 1)
	{
		x = "SELECT TOP 18 * FROM All_UAV WHERE MISSION Like '";
		x+=y;
		x+="%'";
	}


	else
	{
		x = "SELECT TOP 18 * FROM All_UAV WHERE MISSION = '";
		x+=y;
		x+="'";
	}

	m_adovtmain.SetRecordSource(x);
	m_adovtmain.Refresh();
}

void CBviDlg::OnButton1()
{
	m_find.EnableWindow(TRUE);
	m_find.SetWindowText("Find");
	m_e5c.ShowWindow(SW_HIDE);
	m_c.ShowWindow(SW_HIDE);
	CString x = "SELECT * FROM All_UAV";
	m_adovtmain.SetRecordSource(x);
	m_adovtmain.Refresh();
	m_c.ShowWindow(SW_HIDE);
 }
```

```
void CBviDlg::OnChangeEdit5()
{
	m_find.EnableWindow(FALSE);
	UpdateData(TRUE);
	CString str = m_e5str;
	CString x = " SELECT TOP 18 * FROM All_UAV WHERE NAME ";
	CFind2BVIDlg b;

	if (return1 == 1)
	{
		x+= "Like '";
		x+= str;
		x+= "%'";
	}

	else
	{
		x+= "='";
		x+=str;
		x+= "'";
	}

	m_adovtmain.SetRecordSource(x);
	m_adovtmain.Refresh();
	m_editva.Format("%ld",m_dg.GetColumns().GetCount());
	m_editva1.Format("%ld",m_adovtmain.GetRecordset().GetRecordCount());

	if(m_adovtmain.GetRecordset().GetAbsolutePosition()!= -1)
	m_editva2.Format("%ld",m_adovtmain.GetRecordset().GetAbsolutePosition());

	else
	m_editva2.Format("%ld",m_adovtmain.GetRecordset().GetAbsolutePosition()+1);
	UpdateData(FALSE);
}
void CBviDlg::OnOK()
{
	int pnum=1;
	CAdodc* ad =(CAdodc*)GetDlgItem(IDC_ADOVTUAVMAIN);
	VARIANT a;
	a = ad->GetRecordset().GetBookmark();
	a.intVal = ad->GetRecordset().GetAbsolutePosition();

	CDataList* dl1 =(CDataList*)GetDlgItem(IDC_DATALISTID1);
	CDataList* dl2 =(CDataList*)GetDlgItem(IDC_DATALISTID2);
	CDataList* dl3 =(CDataList*)GetDlgItem(IDC_DATALISTID3);
	CDataList* dl4 =(CDataList*)GetDlgItem(IDC_DATALISTID4);
	CDataList* dl5 =(CDataList*)GetDlgItem(IDC_DATALISTID5);
	CDataList* dl6 =(CDataList*)GetDlgItem(IDC_DATALISTID6);
	CDataList* dl7 =(CDataList*)GetDlgItem(IDC_DATALISTID7);
	CDataList* dl8 =(CDataList*)GetDlgItem(IDC_DATALISTID8);
	CDataList* dl9 =(CDataList*)GetDlgItem(IDC_DATALISTID9);
	CDataList* dl10 =(CDataList*)GetDlgItem(IDC_DATALISTID10);
	CDataList* dl11 =(CDataList*)GetDlgItem(IDC_DATALISTID11);

	//1. Construct a CPrintDialog object
	CPrintDialog dlgPrint(FALSE,PD_NOSELECTION,this);
	//PD_ALLPAGES flag was used by me at first instance
	MessageBox("Please select landscape orientation and A4 paper size. Elsewise, print will not work
	correctly","Compulsory Printer Settings");

	//dlgPrint.m_pd.Flags
	//2. The printdialog has to be initiated and displayed
```

```
if(dlgPrint.DoModal()==IDOK)
{
	//3. Attach the printer DC from the dialog to a CDC object
	CDC dcPrint;
	dcPrint.Attach(dlgPrint.GetPrinterDC());

	//4. Create and Fill a DOCINFO structure
	DOCINFO myPrintJob;
	myPrintJob.cbSize = sizeof(myPrintJob);
	myPrintJob.lpszDocName = "UAV_Design_PrintJob";
	myPrintJob.lpszOutput = NULL;
	myPrintJob.lpszDatatype = NULL;
	myPrintJob.fwType = NULL;

	 //5. Start the printing document. StartDoc() function tells window to start spooling and
	EndDoc tells it that the spooling is done
	if(dcPrint.StartDoc(&myPrintJob)>=0)
	//Document is checking if their is sufficient memory/diskspace to proceed or not.
	{
		//5.1 Start the page
		dcPrint.StartPage();

		//5.2 Start the drawing
		dcPrint.TextOut(3000,30,"UAV DESIGN ASSIST DATASHEET");
		dcPrint.TextOut(0,100,"ID");//dl1->GetBoundColumn());
		dcPrint.TextOut(300,100,"Name");//dl1->GetBoundColumn());
		dcPrint.TextOut(2100,100,"Length"); //dl2->GetBoundColumn());
		dcPrint.TextOut(2600,100,"Wing");//dl3->GetBoundColumn());
		dcPrint.TextOut(3000,100,"Rotor");//dl4->GetBoundColumn());
		dcPrint.TextOut(3500,100,"Body");//dl5->GetBoundColumn());
		dcPrint.TextOut(3900,100,"Wt.");//dl6->GetBoundColumn());
		dcPrint.TextOut(4300,100,"Power.");//dl7->GetBoundColumn());
		dcPrint.TextOut(4900,100,"Endur.");//dl8->GetBoundColumn());
		dcPrint.TextOut(5400,100,"Range");//dl9->GetBoundColumn());
		dcPrint.TextOut(5900,100,"Altitude");//dl10->GetBoundColumn());
		dcPrint.TextOut(0,200,"");//dl1->GetBoundColumn());
		dcPrint.TextOut(300,200,"");//dl1->GetBoundColumn());
		dcPrint.TextOut(2100,200,"(ft)"); //dl2->GetBoundColumn());
		dcPrint.TextOut(2600,200,"(ft)");//dl3->GetBoundColumn());
		dcPrint.TextOut(3000,200,"(ft)");//dl4->GetBoundColumn());
		dcPrint.TextOut(3500,200,"(ft)");//dl5->GetBoundColumn());
		dcPrint.TextOut(3900,200,"(lb)");//dl6->GetBoundColumn());
		dcPrint.TextOut(4300,200,"(hp)");//dl7->GetBoundColumn());
		dcPrint.TextOut(4900,200,"(hr)");//dl8->GetBoundColumn());
		dcPrint.TextOut(5400,200,"(mi)");//dl9->GetBoundColumn());
		dcPrint.TextOut(5900,200,"(ft)");//dl10->GetBoundColumn());

		//use a bookmark
		ad->GetRecordset().MoveFirst();

		int y=400;
		for(int i=0;i<ad->GetRecordset().GetRecordCount(); i++)
		{
			CString str;
			str.Format("%ld",ad->GetRecordset().GetAbsolutePosition());
			dcPrint.TextOut(0,y,str);
			dcPrint.TextOut(300,y,dl1->GetBoundText());
			dcPrint.TextOut(2100,y,dl2->GetText());// >GetBoundText());
			dcPrint.TextOut(2600,y,dl3->GetText());// >GetBoundText());
			dcPrint.TextOut(3000,y,dl4->GetText());// >GetBoundText());
			dcPrint.TextOut(3500,y,dl5->GetText());// >GetBoundText());
			dcPrint.TextOut(3900,y,dl6->GetText());// >GetBoundText());
			dcPrint.TextOut(4300,y,dl7->GetText());// >GetBoundText());
```

```
		dcPrint.TextOut(4900,y,dl8->GetText());// >GetBoundText());
		dcPrint.TextOut(5400,y,dl9->GetText());// >GetBoundText());
		dcPrint.TextOut(5900,y,dl10->GetText());// >GetBoundText())

		ad->GetRecordset().MoveNext();
		y+=175;

		if((y/4300)>0)
		{
			CString pagem,page;
			page="Page ";
			pagem.Format("%d",pnum++);
			page+=pagem;
			dcPrint.TextOut(3000,4400,page);
			dcPrint.EndPage();
			dcPrint.StartPage();
			dcPrint.TextOut(0,100,"ID");
			dcPrint.TextOut(300,100,"Name");Column());
			dcPrint.TextOut(2100,100,"Length");
			dcPrint.TextOut(2600,100,"Wing");
			dcPrint.TextOut(3000,100,"Rotor");
			dcPrint.TextOut(3500,100,"Body");
			dcPrint.TextOut(3900,100,"Wt.");
			dcPrint.TextOut(4300,100,"Power");
			dcPrint.TextOut(4900,100,"Endur.");
			dcPrint.TextOut(5400,100,"Range");
			dcPrint.TextOut(5900,100,"Altitude");
			dcPrint.TextOut(0,200,"");
			dcPrint.TextOut(300,200,"");
			dcPrint.TextOut(2100,200,"(ft)");
			dcPrint.TextOut(2600,200,"(ft)");
			dcPrint.TextOut(3000,200,"(ft)");
			dcPrint.TextOut(3500,200,"(ft)");
			dcPrint.TextOut(3900,200,"(lb)");
			dcPrint.TextOut(4300,200,"(hp)");
			dcPrint.TextOut(4900,200,"(hr)");
			dcPrint.TextOut(5400,200,"(mi)");
			dcPrint.TextOut(5900,200,"(ft)");
			y=400;
		}

}
ad->GetRecordset().SetAbsolutePosition(a.lVal);
 //Using Bookmarks.can also use (a.intval) no warnings will be generated
//ad->GetRecordset().MoveFirst();

if((y/4300) >0)
// This loop is for checking the space to print my warning below
{

		CString pagem,page;
		page="Page";
		pagem.Format("%d",++pnum);
		page+=pagem;
		dcPrint.TextOut(3500,4400, page);
		dcPrint.EndPage();
		dcPrint.StartPage();
		y=4450;
}

dcPrint.TextOut(10,y+80,"This Text has been Printed from the VTUAV DESIGN
ASSIST SOFTWARE's DATABASE COLLECTION. Software");
```

```
dcPrint.TextOut(10,y+160,"Manufacturer's and Distributors do not attest to the
accuracy of this information and thereby are not");
dcPrint.TextOut(10,y+240,"liable for any resulting damages/charges. ");

SYSTEMTIME a;
GetSystemTime(&a);
CString afinal,a1,a2,a3;
a3=".";
afinal="The Document has been printed on ";

if(a.wDayOfWeek==0)
        a1="Sunday";
if(a.wDayOfWeek==1)
        a1="Monday";
if(a.wDayOfWeek==2)
        a1="Tuesday";
if(a.wDayOfWeek==3)
        a1="Wednesday";
if(a.wDayOfWeek==4)
        a1="Thursday";
if(a.wDayOfWeek==5)
{a1="Friday"; a3=" Have A Great WeekEnd";}
if(a.wDayOfWeek==6)
a1="Saturday";

afinal+=a1;
afinal+=",The ";

a1="";
a1.Format("%d",a.wDay);//Word is a 16bit integer/equivalent to short int
if((a.wDay) >= 4 && (a.wDay)<= 20) a1+="th";
if((a.wDay) >= 24 && (a.wDay)<= 30)   a1+="th";
if(a.wDay == 1 || a.wDay == 21 || a.wDay == 31) a1+="st";
if(a.wDay == 2 || a.wDay == 22) a1+="nd";
if(a.wDay == 3 || a.wDay == 23) a1+="rd";

a1+="of";

if(a.wMonth == 1) a2="January";
if(a.wMonth == 2) a2="February";
if(a.wMonth == 3) a2="March";
if(a.wMonth == 4) a2="April";
if(a.wMonth == 5) a2="May";
if(a.wMonth == 6) a2="June";
if(a.wMonth == 7) a2="July";
if(a.wMonth == 8) a2="August";
if(a.wMonth == 9) a2="September";
if(a.wMonth == 10) a2="October";
if(a.wMonth == 11) a2="November";
if(a.wMonth == 12) a2="December";

//a2.Format("%d", a.wMonth);
a1+=a2;
a1+=",";
a2="";
a2.Format("%d", a.wYear);
a1+=a2;
a1+=a3;
afinal+=a1;
dcPrint.TextOut(10,y+320,afinal);

//5.3 Throw the page
dcPrint.EndPage();
```

```
                    //5.4 Close the document.
                    dcPrint.EndDoc();
                }

                    //6. Donot forget to delete printer devide context
                    dcPrint.DeleteDC();
        }

}
```

H.11 'Print' Class

The print class involves print of database and design configuration as follows:

```
BOOL CVTUAV_Expert_SystemView::OnPreparePrinting(CPrintInfo* pInfo)
{
        // Default preparation
        pInfo->SetMinPage(1);
        pInfo->SetMaxPage(2);
        pInfo->m_pPD->m_pd.nCopies=1; // Default number of copies is 1

         if(!pInfo->m_bPreview)
        AfxMessageBox("The Program is initiating your default Printer. For your information the default
        number of copies of each page is 1. The maximum number of pages are set to 2. Please manually
        override to change the settings. Also, you cannot print more than 5 copies of each page. This setting is
        non-negotiable.");


        do
        {
         if(DoPreparePrinting(pInfo) == FALSE)
         return FALSE;

         //Warn user if too many copies are specified
         if(pInfo->m_pPD->GetCopies() > 5)
         MessageBox("Please choose less than 6 copies","Warning",MB_OK);

        }while(pInfo->m_pPD->GetCopies()>5);// Cant print more than 5 copies of the program

         if(!pInfo->m_bPreview)
        AfxMessageBox("Please re-verify your printer settings. The selected Device is  " +pInfo->m_pPD-
        >GetDeviceName()+ " on  " +pInfo->m_pPD->GetPortName()+ " port and with  " +pInfo->m_pPD-
        >GetDriverName()+ " driver.",  MB_OK);// == IDCANCEL)

        return DoPreparePrinting(pInfo);
}
```

H.12 'Mission categories' Class

The class provides the option to select the desired category of mission requirements based on the operational needs as follows:

// MRDlg.cpp : implementation file

```
// Declaration of header files
#include "stdafx.h"
#include "VTUAV_Expert_System.h"
#include "MRDlg.h"
#include "MR1ADlg.h"
#include "MR1BDlg.h"
#include "MR1CDlg.h"
#include "MR1DDlg.h"
#include "MR1EDlg.h"
#include "InputDlg.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

CMRDlg::CMRDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CMRDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CMRDlg)
	//}}AFX_DATA_INIT
}

// Declaration & initialisation of class variables
void CMRDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CMRDlg)
	DDX_Control(pDX, IDMR5, m_mr5);
	DDX_Control(pDX, IDMR4, m_mr4);
	DDX_Control(pDX, IDMR3, m_mr3);
	DDX_Control(pDX, IDMR2, m_mr2);
	DDX_Control(pDX, IDMR1, m_mr1);
	//}}AFX_DATA_MAP
}

// Limiting variables values
BEGIN_MESSAGE_MAP(CMRDlg, CDialog)
	//{{AFX_MSG_MAP(CMRDlg)
	ON_BN_CLICKED(IDMR1, OnMr1)
	ON_BN_CLICKED(IDMR2, OnMr2)
	ON_BN_CLICKED(IDMR3, OnMr3)
	ON_BN_CLICKED(IDMR4, OnMr4)
	ON_BN_CLICKED(IDMR5, OnMr5)
	ON_BN_CLICKED(IDC_BUTTON1, OnButton1)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

int CMRDlg::DoModal()
{
	return CDialog::DoModal();
}

void CMRDlg::OnMr1()
{
	EndDialog(IDOK);
	CMR1ADlg a;
	a.DoModal();
}

void CMRDlg::OnMr2()
{
	EndDialog(IDOK);
```

```
	CMR1BDlg a;
	a.DoModal();
}

void CMRDlg::OnMr3()
{
	EndDialog(IDOK);
	CMR1CDlg a;
	a.DoModal();
}

void CMRDlg::OnMr4()
{
	EndDialog(IDOK);
	CMR1DDlg a;
	a.DoModal();
}

void CMRDlg::OnMr5()
{
	EndDialog(IDOK);
	CMR1EDlg a;
	a.DoModal();
}

void CMRDlg::OnButton1()
{
	EndDialog(IDOK);
	CInputDlg a;
	a.DoModal();
}
```

H.13 'Mission requirements' Class

The class involves mission requirements based on the selected mission category. The class is programmed as follows as an illustration for 'Generic' mission category:

```
// MR1ADlg.cpp : implementation file
// Declaration of header files
#include "stdafx.h"
#include "VTUAV_Expert_System.h"
#include "MR1ADlg.h"
#include "MRDlg.h"
#include "MR1A1Dlg.h"
#include "MR1A2Dlg.h"
#include "MR1A3Dlg.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

CMR1ADlg::CMR1ADlg(CWnd* pParent /*=NULL*/)
	: CDialog(CMR1ADlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CMR1ADlg)
		// NOTE: the ClassWizard will add member initialization here
	//}}AFX_DATA_INIT
```

```
}


void CMR1ADlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CMR1ADlg)
		// NOTE: the ClassWizard will add DDX and DDV calls here
	//}}AFX_DATA_MAP
}

// Limiting variables values
BEGIN_MESSAGE_MAP(CMR1ADlg, CDialog)
	//{{AFX_MSG_MAP(CMR1ADlg)
	ON_BN_CLICKED(IDMR1A1, OnMr1a1)
	ON_BN_CLICKED(IDMR1A2, OnMr1a2)
	ON_BN_CLICKED(IDMR1A3, OnMr1a3)
	ON_BN_CLICKED(IDC_BUTTON2, OnButton2)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

void CMR1ADlg::OnMr1a1()
{
	EndDialog(IDCANCEL);
	CMR1A1Dlg a;
	a.DoModal();
}

void CMR1ADlg::OnMr1a2()
{
	EndDialog(IDCANCEL);
	CMR1A2Dlg a;
	a.DoModal();
}

void CMR1ADlg::OnMr1a3()
{
	EndDialog(IDCANCEL);
	CMR1A3Dlg a;
	a.DoModal();
}

void CMR1ADlg::OnButton2()
{
	EndDialog(IDCANCEL);
	CMRDlg a;
	a.DoModal();
}

BEGIN_EVENTSINK_MAP(CMR1ADlg, CDialog)
	//{{AFX_EVENTSINK_MAP(CMR1ADlg)
	//}}AFX_EVENTSINK_MAP
END_EVENTSINK_MAP()
```

H.14 ‘Application’ Class

The class responds to menu and taskbar interactions as follows:

```
// VTUAV_Expert_System.h : main header file for the VTUAV_Expert_System application
#if
!defined(AFX_VTUAV_Expert_System_H__9090EBC3_9518_11D8_A133_0004AC6E2E50__INCLUDED_)
#define AFX_VTUAV_Expert_System_H__9090EBC3_9518_11D8_A133_0004AC6E2E50__INCLUDED_
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
#ifndef __AFXWIN_H__
	#error include 'stdafx.h' before including this file for PCH
#endif

// Declaration of header files
#include "resource.h"       // main symbols
#include "VTUAV_Expert_System_i.h"

class CVTUAV_Expert_SystemApp : public CWinApp
{
public:
	CVTUAV_Expert_SystemApp();

	// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CVTUAV_Expert_SystemApp)
	public:
	virtual BOOL InitInstance();
	virtual BOOL InitApplication();
	virtual int ExitInstance();
	//}}AFX_VIRTUAL

	// Implementation
	//{{AFX_MSG(CVTUAV_Expert_SystemApp)
	afx_msg void OnAppAbout();
	afx_msg void OnDatabaseViewBvi();
	afx_msg void OnDatabaseViewDim();
	afx_msg void OnDatabaseViewOperst();
	afx_msg void OnDatabaseViewPerf();
	afx_msg void OnDatabaseViewPwrpt();
	afx_msg void OnDatabaseViewWt();
	afx_msg void OnMemoryUsage();
	afx_msg void OnWinappExplorer();
	afx_msg void OnWinappNotepad();
	afx_msg void OnWinappProgman();
	afx_msg void OnWinappWordpad();
	afx_msg void OnWinappCalculator();
	afx_msg void OnViewExplorer();
	afx_msg void OnFileFtp();
	afx_msg void OnViewFilemanager();
	afx_msg void OnWinappDrwatson();
	afx_msg void OnWinappRegistoryscanner();
	afx_msg void OnWinappSfc();
	afx_msg void OnDatabaseAddnewrecord();
	afx_msg void OnDatabaseUpdateexisitngrecord();
	afx_msg void OnDatabaseDeleteexisitingrecord();
	afx_msg void OnDesignprogramRun();
	afx_msg void OnDatabaseDisplayviewsEnginemodeclassificationElectricalengine();
	afx_msg void OnDatabaseDisplayviewsEnginemodeclassificationGliding();
```

```
        afx_msg void OnDatabaseDisplayviewsEnginemodeclassificationNelecengine();
        afx_msg void OnDatabaseEnginelistModify();
        afx_msg void OnDatabaseEnginelistView();
        afx_msg void OnIdsMemoryDevice();
        afx_msg void OnAppAboutHelp();
        afx_msg void OnHelpFinder();
        afx_msg void OnDesignprogramShadeoutthe3dprofile();
        afx_msg void OnDesignprogramDotoutthe3dprofile();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()

private:
        BOOL m_bATLInited;
        BOOL InitATL();
        int ncount;
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.
#endif //
!defined(AFX_VTUAV_Expert_System_H__9090EBC3_9518_11D8_A133_0004AC6E2E50__INCLUDED_)
```

H.15 'View' Class

The class responds to interactions involving input devices including mouse and keyboard as follows:

```
// VTUAV_Expert_SystemView.h : interface of the CVTUAV_Expert_SystemView class
#if
!defined(AFX_VTUAV_Expert_SystemVIEW_H__9090EBCB_9518_11D8_A133_0004AC6E2E50__INCLU
DED_)
#define
AFX_VTUAV_Expert_SystemVIEW_H__9090EBCB_9518_11D8_A133_0004AC6E2E50__INCLUDED_
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000


class CVTUAV_Expert_SystemView : public CView
{
        protected: // create from serialization only
        CVTUAV_Expert_SystemView();
        DECLARE_DYNCREATE(CVTUAV_Expert_SystemView)

// Attributes
public:
        CVTUAV_Expert_SystemDoc* GetDocument();

// Operations
public:

// Overrides
        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CVTUAV_Expert_SystemView)
        public:
        virtual void OnDraw(CDC* pDC);  // overridden to draw this view
        virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
        virtual void OnPrepareDC(CDC* pDC, CPrintInfo* pInfo = NULL);
```

```
protected:
virtual BOOL OnPreparePrinting(CPrintInfo* pInfo);
virtual void OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo);
virtual void OnEndPrinting(CDC* pDC, CPrintInfo* pInfo);
virtual void OnActivateView(BOOL bActivate, CView* pActivateView, CView* pDeactiveView);
virtual void OnPrint(CDC* pDC, CPrintInfo* pInfo);
//}}AFX_VIRTUAL

// Implementation
public:
virtual ~CVTUAV_Expert_SystemView();
#ifdef _DEBUG
virtual void AssertValid() const;
virtual void Dump(CDumpContext& dc) const;
#endif

protected:
// Generated message map functions
protected:
//{{AFX_MSG(CVTUAV_Expert_SystemView)
afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
afx_msg void OnMouseMove(UINT nFlags, CPoint point);
afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
afx_msg void OnRButtonDown(UINT nFlags, CPoint point);
//}}AFX_MSG
DECLARE_MESSAGE_MAP()
private:
CRect m_printrect;
int py;
int px;
};

#ifndef _DEBUG  // debug version in VTUAV_Expert_SystemView.cpp
inline CVTUAV_Expert_SystemDoc* CVTUAV_Expert_SystemView::GetDocument()
   { return (CVTUAV_Expert_SystemDoc*)m_pDocument; }
#endif

//{{AFX_INSERT_LOCATION}}
#endif //
!defined(AFX_VTUAV_Expert_SystemVIEW_H__9090EBCB_9518_11D8_A133_0004AC6E2E50__INCLU
DED_)
```

H.16 'Document' Class

The class involves database usage vis-à-vis the connection string as follows:

```
// VTUAV_Expert_SystemDoc.h : interface of the CVTUAV_Expert_SystemDoc class
#if
!defined(AFX_VTUAV_Expert_SystemDOC_H__9090EBC9_9518_11D8_A133_0004AC6E2E50__INCLUD
ED_)
#define
AFX_VTUAV_Expert_SystemDOC_H__9090EBC9_9518_11D8_A133_0004AC6E2E50__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

class CVTUAV_Expert_SystemDoc : public CDocument
{
```

```
        protected: // create from serialization only
        CVTUAV_Expert_SystemDoc();
        DECLARE_DYNCREATE(CVTUAV_Expert_SystemDoc)

// Attributes
public:

// Operations
public:

// Overrides
        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CVTUAV_Expert_SystemDoc)
        public:
        virtual BOOL OnNewDocument();
        virtual void Serialize(CArchive& ar);
        //}}AFX_VIRTUAL

// Implementation
public:
        virtual ~CVTUAV_Expert_SystemDoc();
        #ifdef _DEBUG
        virtual void AssertValid() const;
        virtual void Dump(CDumpContext& dc) const;
        #endif

protected:

protected:
        //{{AFX_MSG(CVTUAV_Expert_SystemDoc)
        // NOTE - the ClassWizard will add and remove member functions here.
        // DO NOT EDIT what you see in these blocks of generated code !
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif //
!defined(AFX_VTUAV_Expert_SystemDOC_H__9090EBC9_9518_11D8_A133_0004AC6E2E50__INCLUD
ED_)
```

APPENDIX J

J.1 'Design-input' class

The program code of the class validates the acquired design requirements and constraints as follows:

```
// File name: InputDlg.cpp
// Inclusion of header files
#include "stdafx.h"
#include "VTUAV_Expert_System.h"
#include "InputDlg.h"
#include "WelcomeDlg.h"
#include "MRDlg.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

// Declaration & initialisation of class variables
CInputDlg::CInputDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CInputDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CInputDlg)
	m_input1 = 0.0;
	m_input2 = 0.0;
	m_input3 = 0.0;
	m_input4 = 0.0;
	m_input5 = 0.0;
	m_input6 = 0.0;
	m_input7 = 0.0;
	m_input8 = 0.0;
	m_input9 = 0.0;
	m_input9a = 0.0;
	m_cf = 0;
	m_time = 1;
	m_yn = 1;
	m_input11 = 0.0;
	//}}AFX_DATA_INIT
}

// Limiting variables values
Void CInputDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CInputDlg)
	DDX_Text(pDX, IDC_EDIT1, m_input1);
	DDV_MinMaxDouble(pDX, m_input1, 4.9, 9586.);
	DDX_Text(pDX, IDC_EDIT2, m_input2);
	DDV_MinMaxDouble(pDX, m_input2, 1.7, 6414.);
	DDX_Text(pDX, IDC_EDIT3, m_input3);
	DDV_MinMaxDouble(pDX, m_input3, 6.6, 16000.);
	DDX_Text(pDX, IDC_EDIT4, m_input4);
	DDV_MinMaxDouble(pDX, m_input4, 21., 253.);
	DDX_Text(pDX, IDC_EDIT5, m_input5);
	DDV_MinMaxDouble(pDX, m_input5, 6.e-002, 40.);
	DDX_Text(pDX, IDC_EDIT6, m_input6);
```

```
        DDV_MinMaxDouble(pDX, m_input6, 1.15, 6762.);
        DDX_Text(pDX, IDC_EDIT7, m_input7);
        DDV_MinMaxDouble(pDX, m_input7, 400., 330000.);
        DDX_Text(pDX, IDC_EDIT8, m_input8);
        DDV_MinMaxDouble(pDX, m_input8, 0.5, 115.);
        DDX_Text(pDX, IDC_EDIT9, m_input9);
        DDV_MinMaxDouble(pDX, m_input9, 0.5, 247.);
        DDX_Text(pDX, IDC_EDIT9a, m_input9a);
        DDV_MinMaxDouble(pDX, m_input9a, 0.2, 34.);
        DDX_Radio(pDX, IDC_RADIO1, m_cf);
        DDX_Radio(pDX, IDC_RADIO5, m_time);
        DDX_Radio(pDX, IDC_RADIO3, m_yn);
        DDX_Text(pDX, IDC_EDIT11, m_input11);
        DDX_Control(pDX, IDC_LABEL34, m_label);
        //}}AFX_DATA_MAP
}

// Defining user actions vis-a-vis system messages
BEGIN_MESSAGE_MAP (CInputDlg, CDialog)
        //{{AFX_MSG_MAP(CInputDlg)
        ON_BN_CLICKED (IDIH, OnIh)
        ON_BN_CLICKED (IDIN, OnIn)
        ON_BN_CLICKED (IDIOFC, OnIofc)
        ON_BN_CLICKED (IDIMI, OnImi)
        ON_EN_CHANGE (IDC_EDIT2, OnChangeEdit2)
        ON_EN_CHANGE (IDC_EDIT4, OnChangeEdit4)
        ON_EN_CHANGE (IDC_EDIT1, OnChangeEdit1)
        ON_BN_CLICKED (IDC_RADIO3, OnRadio3)
        ON_BN_CLICKED (IDC_RADIO4, OnRadio4)
        ON_BN_CLICKED (IDMPE, OnMpe)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP ()

// Class method – Dialog initialisation
BOOL CInputDlg::OnInitDialog()
{

        CDialog::OnInitDialog();
        Extern double ewt; //Empty weight
        Extern double pwt; //Payload weight
        Extern double lwt; //MTOW or launch weight
        Extern double sp; //Forward speed
        Extern double en; //Endurance
        Extern double ra; //Range
        Extern double al; //Altitude
        Extern double le; //Fuselage length
        Extern double sn; //Fuselage span or width
        Extern double di; //Rotor diameter
        Extern double sr; //Safety regulations
        Extern double aw; //Stub wing or additional wing
        Extern double fr; //Fuel reserve weight
        Extern double wsn; //Wing span

        // Retrieve values during back navigation
        If ((ewt > 0))
        {
                m_input1=ewt;
                m_input2=pwt;
                m_input3=lwt;
                m_input4=sp;
```

```
                m_input5=en;
                m_input6=ra;
                m_input7=al;
                m_input8=le;
                m_input9=sn;
                m_input9a=di;
                m_input11=wsn;
                m_cf=int(sr);
                m_yn=int(aw);
                m_time=int(fr);
                UpdateData(FALSE);
        }
        If (m_yn==0)
        {
                CWnd* a=(CWnd*)GetDlgItem(IDC_EDIT11);
                a->ShowWindow(SW_SHOW);
        }
        return TRUE;
}

// Class method – Dialog display
Void CInputDlg::OnIh ()
{
        CDialog::EndDialog(IDOK);
        CWelcomeDlg a;
        a.DoModal();
}

// Class method – Input validation
Void CInputDlg::OnIn()
{
        // Recalling global (public) variable declarations
        Extern double ewt; //Empty weight
        Extern double pwt; //Payload weight
        Extern double lwt; //MTOW or launch weight
        Extern double ewt1; //Interim empty weight
        Extern double pwt1; //Interim payload weight
        Extern double lwt1;  //Interim MTOW or launch weight
        Extern double sp; //Forward speed
        Extern double en; //Endurance
        Extern double ra; //Range
        Extern double al; //Altitude
        Extern double le; //Fuselage length
        Extern double sn; //Fuselage span or width
        Extern double di; //Rotor diameter
        Extern double sr; //Safety regulations
        Extern double aw; //Stub wing or additional wing
        Extern double fr; //Fuel reserve weight
        Extern double wsn; //Wing span
        Extern double uavprojectedarea; // Projected area of the fuselage
        UpdateData(TRUE);


        bool proceed=FALSE, proceed1=FALSE;
        double var;
        CString buffer;

        // Validation of design parameters
        if((m_input1 >= 4.9) && (m_input1 <= 9586) && (m_input2 >= 1.7) && (m_input2 <=
        6414) && (m_input3 >= 6.6) && (m_input3 <= 16000) && (m_input4 >= 21) && (m_input4
```

```
<= 253) && (m_input5 >= 0.06) && (m_input5 <= 40) && (m_input6 >= 1.15) &&
(m_input6 <= 6762) && (m_input7 >= 400) && (m_input7 <= 330000) && (m_input8 >=
0.5) && (m_input8 <= 115) && (m_input9 >= 0.5) && (m_input9 <= 247) && (m_input9a
>= 0.2) && (m_input9a <= 34) )
{
	if((m_input3 / 2.2) <= 8)
	{
		var=1.064 * ((0.0177 * (m_input3 / 2.2) * (m_input3 / 2.2)) + (0.1132 *
		(m_input3 / 2.2)) + 0.0854);
		var=var*2.2;
			if (var >= m_input2)
			{
				proceed =TRUE;
			}
			else
			{
				buffer.Format("%lf",var);
				if(MessageBox("The maximum payload weight for
				MTOW is " +buffer+ " lb. Please reduce the payload or
				alternatively increase the empty weight of the aircraft. Do
				you want to view the involved formulas? ","Unexpected
				Payload Values" , MB_YESNO |
				MB_ICONEXCLAMATION  | MB_DEFBUTTON1 ) ==
				IDYES)
				{
					MessageBox("Micro UAV: Payload-
					max=1.064(0.0177(MTOW)^2 +
					0.1132(MTOW) + 0.0854); Small UAV:
					Payload-max=1.235(0.0025(MTOW)^2 +
					0.0184(MTOW) + 3.6847); Large UAV:
					Payload-max=1.393(1E-05(MTOW)^2 +
					0.1264(MTOW) + 22.25)","Launch Wt., Empty
					Wt. and Payload Wt. co-relation" , MB_OK |
					MB_ICONQUESTION  |MB_DEFBUTTON1 );
				}
			}
	}
	else if(((m_input3 / 2.2) > 8) && ((m_input3 / 2.2) <= 100))
	{
		var=1.235 * ((0.0025 * (m_input3 / 2.2) * (m_input3 / 2.2)) + (0.0184 *
		(m_input3 / 2.2)) + 3.6847);
		var=var*2.2;
		if (var >= m_input2)
		{
			proceed=TRUE;
		}
		else
		{
			buffer.Format("%lf",var);
			if(MessageBox("The maximum payload weight for MTOW is "
			+buffer+ " lb. Please reduce the payload or alternatively increase
			the empty weight of the aircraft. Do you want to view the involved
			formulas? ","Unexpected Payload Values", MB_YESNO |
			MB_ICONEXCLAMATION | MB_DEFBUTTON1) == IDYES)
			{
				MessageBox("Micro UAV: Payload-
				max=1.064(0.0177(MTOW)^2 + 0.1132(MTOW) +
				0.0854); Small UAV: Payload-
				max=1.235(0.0025(MTOW)^2 + 0.0184(MTOW) +
				3.6847); Large UAV: Payload-max=1.393(1E-
```

```
                                        05(MTOW)^2 + 0.1264(MTOW) + 22.25)","Launch Wt.,
                                        Empty Wt. and Payload Wt. co-relation" , MB_OK |
                                        MB_ICONQUESTION  |MB_DEFBUTTON1 );
                                }
                        }
                }
                else
                {
                        var=1.393 * ((1E-05 * (m_input3 / 2.2) * (m_input3 / 2.2)) + (0.1264 *
                        (m_input3 / 2.2)) + 22.25);
                        var=var*2.2;
                        if (var >= m_input2)
                        {
                                proceed=TRUE;
                        }
                        else
                        {
                                buffer.Format("%lf",var);
                                if(MessageBox("The maximum payload weight for MTOW is "
                                +buffer+ " lb. Please reduce the payload or alternatively increase
                                the empty weight of the aircraft. Do you want to view the involved
                                formulas? ","Unexpected Payload Values" , MB_YESNO |
                                MB_ICONEXCLAMATION  | MB_DEFBUTTON1 ) == IDYES)
                                {
                                        MessageBox("Micro UAV: Payload-
                                        max=1.064(0.0177(MTOW)^2 + 0.1132(MTOW) +
                                        0.0854); Small UAV: Payload-
                                        max=1.235(0.0025(MTOW)^2 + 0.0184(MTOW) +
                                        3.6847); Large UAV: Payload-max=1.393(1E-
                                        05(MTOW)^2 + 0.1264(MTOW) + 22.25)","Launch Wt.,
                                        Empty Wt. and Payload Wt. co-relation" , MB_OK |
                                        MB_ICONQUESTION  |MB_DEFBUTTON1 );
                                }
                         }
                }
                if(((m_input4*m_input5/m_input6) >= 1) && ((m_input4*m_input5/m_input6) <=
                4))
                {
                        proceed1=TRUE;
                }
                else
                {
                        MessageBox("Speed, endurance and range values are not as expected. The
                        lower limit for (Speed*Endurance/Range) value is 1 and the upper limit for
                        (Speed*Endurance/Range) value is 4. Please adjust the values
                        accordingly","Unexpected Speed, Range and Endurance Co-relation" ,
                        MB_OK | MB_ICONEXCLAMATION  | MB_DEFBUTTON1 );
                }
}

// Save the validated design parameters
if(proceed && proceed1 && (m_input1 >= 4.9) && (m_input1 <= 9586) && (m_input2 >=
1.7) && (m_input2 <= 6414) && (m_input3 >= 6.6) && (m_input3 <= 16000) &&
(m_input4 >= 21) && (m_input4 <= 253) && (m_input5 >= 0.06) && (m_input5 <= 40) &&
(m_input6 >= 1.15) && (m_input6 <= 6762) && (m_input7 >= 400) && (m_input7 <=
330000) && (m_input8 >= 0.5) && (m_input8 <= 115) && (m_input9 >= 0.5) &&
(m_input9 <= 247) && (m_input9a >= 0.2) && (m_input9a <= 34) )
{
        ewt=m_input1;
        pwt=m_input2;
```

```
wt=m_input3;
sp=m_input4;
en=m_input5;
ra=m_input6;
al=m_input7;
le=m_input8;
sn=m_input9;
di=m_input9a;
sr=m_cf;
aw=m_yn;
fr=m_time;
wsn=m_input11;

// Conversion to SI system – Pound to kg
ewt1=ewt/2.2;
pwt1=pwt/2.2;
lwt1=lwt/2.2;
CString sewt1,spwt1,slwt1;
sewt1.Format("%lf", ewt1);
spwt1.Format("%lf", pwt1);
slwt1.Format("%lf", lwt1);

// CASA Safety Regulation
if(m_cf==0)
{
	if(lwt1<=(0.1))
	{
		MessageBox("1. Since the MTOW of the UAV is less than 100 grams, the aircraft should be classified as a MICRO UAV. 2. Similar to remotely controlled model aircrafts, micro UAVs are exempt from most of regulatory requirements (registration and marking, use of radio and certification) applying to larger UAVs but they are subject to rules governing hazardous operation, flight in controlled airspace and operations exceeding 400ft AGL. 3. A need for a telephony device is strongly stated. Please ensure that you later select it from payload classification section. 4. Finally, all the local state laws are applicable. Please obtain a copy of them from the respective offices before the design is put into practical implementation","CASA REGULATION CLASSIFICATION - MICRO UAV", MB_OK | MB_ICONINFORMATION |MB_DEFBUTTON1);
	}
	else if((lwt1>0.1) && (lwt1<=100))
	{
		if(lwt1>0.1 && lwt1<=8)
		{
			MessageBox("CASA classifies micro UAV aircraft to be of maximum 100 gms which is inconsistent with the world standard of upto 8kg MTOW","SMALL UAV CLASSIFICATION", MB_OK | MB_ICONINFORMATION |MB_DEFBUTTON1);
		}
		MessageBox("1. Since the MTOW of the UAV is more than 100 grams and less than 100 kg, the aircraft should be classified as a SMALL UAV. 2. They may only fly outside an approved area if they are kept clear of populous areas and controlled airspace. If flying above 400ft AGL, the operation must have CASA approval. 3. A need for a telephony device is strongly stated. Please ensure that you later select it from payload classification section. 4. Finally, all the local state laws are applicable. Please obtain a copy
```

```
                of them from the respective offices before the design is put into
                practical implementation","CASA REGULATION
                CLASSIFICATION - SMALL UAV", MB_OK |
                MB_ICONINFORMATION  |MB_DEFBUTTON1);
        }
        else
        {
                MessageBox("1. Since the MTOW of the UAV is more than 100
                kg, the aircraft should be classified as a LARGE UAV. Usually the
                weight classification for large UAVs starts at 150 Kg but in case of
                rotorcraft UAVs it starts at 100 kg. 2. Large UAVs are effectively
                treated as conventional aircraft for registration purposes and are
                operated under a special certificate of airworthiness (restricted
                category) or an experimental certificate.  3. Additional regulatory
                requirements include a UAV operator certificate (similar to an
                AOC), controller certificate (similar to a pilot licence) and rules for
                maintenance, use of radiotelephone and ATC procedures. There are
                also operational and equipment specifications. 4. Finally, all the
                local state laws are applicable. Please obtain a copy of them from
                the respective offices before the design is put into practical
                implementation.","CASA REGULATION CLASSIFICATION -
                LARGE UAV", MB_OK | MB_ICONINFORMATION
                |MB_DEFBUTTON1);
        }
}

// Other Safety Standards
if(m_cf==1)
{
        if(lwt1<=8)
        {
                MessageBox("1. Since the MTOW of the UAV is less than 8 kg,
                the aircraft should be classified as a MICRO UAV. 2.Like remotely
                controlled model aircraft, micro UAVs aircraft are exempt from
                most of the regulatory requirements (registration and marking, use
                of radio and certification) applying to larger UAVs. But they are
                subject to rules governing hazardous operation, flight in controlled
                airspace and operations exceeding 400ft AGL. 3. A need for a
                telephony device is strongly stated. Please ensure that you later
                select it from payload classification section. 4. Finally, all the local
                state laws are applicable. Please obtain a copy of them from the
                respective offices before the design is put into practical
                implementation","WORLD SAFETY STANDARDS - MICRO
                UAV", MB_OK | MB_ICONINFORMATION
                |MB_DEFBUTTON1);
        }
        else if(lwt1>8 && lwt1<=100)
        {
                MessageBox("1. Since the MTOW of the UAV is more than 8 kg
                and less than 100 kg, the aircraft should be classified as a SMALL
                UAV. 2. They may only fly outside an approved area if they are
                kept clear of populous areas and controlled airspace. If flying
                above 400ft AGL, the operation must have local authorities
                approval. 3. A need for a telephony device is strongly stated. Please
                ensure that you later select it from payload classification section. 4.
                Finally, all the local state laws are applicable. Please obtain a copy
                of them from the respective offices before the design is put into
                practical implementation","WORLD SAFETY STANDARDS -
                SMALL UAV", MB_OK | MB_ICONINFORMATION
                |MB_DEFBUTTON1);
```

```
                }
                else
                {
                        MessageBox("1. Since the MTOW of the UAV is more than 100
                        kg, the aircraft should be classified as a LARGE UAV. Usually the
                        weight classification for large UAVs starts at 150 Kg but in case of
                        rotorcraft UAVs it starts at 100 kg. 2. Large UAVs are effectively
                        treated as conventional aircraft for registration purposes and are
                        operated under a special certificate of airworthiness (restricted
                        category) or an experimental certificate.  3. Additional regulatory
                        requirements include a UAV operator certificate (similar to an
                        AOC), controller certificate (similar to a pilot licence) and rules for
                        maintenance, use of radiotelephone and ATC procedures. There are
                        also operational and equipment specifications. 4. Finally, all the
                        local state laws are applicable. Please obtain a copy of them from
                        the respective offices before the design is put into practical
                        implementation.","WORLD SAFETY STANDARDS - LARGE
                        UAV", MB_OK | MB_ICONINFORMATION
                        |MB_DEFBUTTON1);
                }
        }
        MessageBox("The SI value for the empty weight is "+sewt1+" kg. The SI value for
        the payload weight is "+spwt1+" kg. The SI value for the launch weight is "+slwt1+"
        kg","SI UNITS CONVERSION" , MB_OK | MB_ICONEXCLAMATION  |
        MB_DEFBUTTON1);

        // Calculation of the projected area
        uavprojectedarea=0.0;
        if (aw == 0)
        {
                // Stub wings required
                uavprojectedarea = 0.5*wsn*le;
        }
        else
        {
                // Stub wing not required
                uavprojectedarea = (2*0.5*3.1416*(le/6)*(le/6)) + (le*sn);
        }
        if(m_yn ==0)
        {
                if(wsn>0)
                {
                        CDialog::EndDialog(IDOK);
                        CMRDlg a;
                        a.DoModal();
                }
                else
                {
                        MessageBox("Please enter Wing Span value","Incorrect Values");
                }
        }
        else
        {
        CDialog::EndDialog(IDOK);
        CMRDlg a;
        a.DoModal();
        }
    }
}
```

// Class method – Constraints display

```
void CInputDlg::OnIofc()
{
        MessageBox("A> Disc loading at minimum end of scale, valued at 4.0 lb/ft*ft  B> Drag
        coefficient = 0.3  c> Number of blades = 4  d> Coefficient of thrust to solidity = 0.17  e>
        Blade aspect ratio = 12.0 f> Initial estimate of blade chord = 0.0127m.  g> Tip speed ratio =
        0.5","Fixed Design Constraints" , MB_OK | MB_ICONINFORMATION
        |MB_DEFBUTTON1);
}
```

// Class method –Safety regulation display

```
void CInputDlg::OnImi()
{
        MessageBox(" 1. UAVs to be classified as either micro(less than 100 grams), small (between
        100g and 100 kg) or large(more than 100kg) with separate rules applicable to each category. 2.
        Micro UAVs are largely exempt from regulation while a small UAV may be flown by an
        unqualified person in certain conditions without any form of certification. A large UAV is a
        de-facto manned aircraft and must be certificated and registered and its controllers qualified.
        3. There are also requirements on the use of a flight radiotelephone; 4. Height restrictions for
        unmanned moored balloons, kites, model aircraft and rockets to be raised from 300ft Above
        Ground Level (AGL) to 400ft AGL in line with overseas practice. 5. Requirements under
        relevant state and territory laws will continue to apply." ,"CASR Part 101 - Unmanned aircraft
        and rocket operations", MB_OK | MB_ICONFIRMATION  |MB_DEFBUTTON1);

        MessageBox(" 1. UAVs to be classified as either micro(less than 8 kg), small (between 8kg
        and 100kg) or large(more than 100kg) with separate rules applicable to each category. 2.
        Micro UAVs are largely exempt from regulation while a small UAV may be flown by an
        unqualified person in certain conditions without any form of certification. A large UAV is a
        de-facto manned aircraft and must generally be certificated and registered and its controllers
        qualified. 3. There are also requirements on the use of Position Lights, Anti-Collision Lights,
        Transponder, Radios, Navigation Systems, UAV System and Altitude Displays, Flight and
        Voice Recorder, and Built-in Test mechanisms. 4. Height restrictions for model aircraft are set
        at 400ft AGL 5. Requirements under relevant state and territory laws will continue to apply."
        ,"World Safety Standards - Unmanned aircrafts", MB_OK | MB_ICONINFORMATION
        |MB_DEFBUTTON1);

}
```

// Class method – String to decimal conversion

```
void CInputDlg::OnChangeEdit2()
{
        CEdit* a=(CEdit*)GetDlgItem(IDC_EDIT1);
        CEdit* b=(CEdit*)GetDlgItem(IDC_EDIT2);
        CEdit* c=(CEdit*)GetDlgItem(IDC_EDIT3);
        CString x, y;
        a->GetWindowText(x);
        b->GetWindowText(y);
        char  *stopstring;
        double z, z1;
        z = strtod( x, &stopstring);
        z1 = strtod( y, &stopstring);
        z=z+z1;
        y.Format("%lf",z);
        c->SetWindowText(y);
}
```

// Class method – String to decimal conversion

```
void CInputDlg::OnChangeEdit4()
{
        CEdit* a=(CEdit*)GetDlgItem(IDC_EDIT1);
```

```
        CEdit* b=(CEdit*)GetDlgItem(IDC_EDIT2);
        CString x, y;
        a->GetWindowText(x);
        b->GetWindowText(y);
        char  *stopstring;
        double z, z1;
         z = strtod( x, &stopstring);
        z1 = strtod( y, &stopstring);

        if(z<z1)
        {
                MessageBox("Payload weight cannot be more than the empty weight","Please Enter
                Values Carefully");
                a->SetFocus();
        }
}
```

// Class method – String to decimal conversion

```
void CInputDlg::OnChangeEdit1()
{
        CEdit* a=(CEdit*)GetDlgItem(IDC_EDIT1);
        CEdit* b=(CEdit*)GetDlgItem(IDC_EDIT2);
        CEdit* c=(CEdit*)GetDlgItem(IDC_EDIT3);
        CString x, y;
        a->GetWindowText(x);
        b->GetWindowText(y);
        char  *stopstring;
        double z, z1;
        z = strtod( x, &stopstring);
        z1 = strtod( y, &stopstring);
        z=z+z1;
        y.Format("%lf",z);
        c->SetWindowText(y);
}
```

// Class method – Show other dialog

```
void CInputDlg::OnRadio3()
{
        CWnd* a=(CWnd*)GetDlgItem(IDC_EDIT11);
        a->ShowWindow(SW_SHOW);
}
```

// Class method – Show other dialog

```
void CInputDlg::OnRadio4()
{
        CWnd* a=(CWnd*)GetDlgItem(IDC_EDIT11);
        a->ShowWindow(SW_HIDE);
}
```

J.2 ‘Mission time’ class

The ‘Mission-time’ class estimates and validates the time required to complete the slated mission. The program code of the ‘Mission-time’ class based on the reconnaissance and surveillance mission is as follows:

// File name: MR1A1Dlg.cpp
// Header Files

```
#include "stdafx.h"
#include "VTUAV_Expert_System.h"
#include "MR1A1Dlg.h"
#include "MR1ADlg.h"
#include "UavdesignfcsDlg.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
```

// Declaration & initialisation of class variables

```
CMR1A1Dlg::CMR1A1Dlg(CWnd* pParent /*=NULL*/)
	: CDialog(CMR1A1Dlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CMR1A1Dlg)
	m_1 = 0;
	m_10 = 0;
	m_11 = 0;
	m_12 = 0;
	m_13 = 0;
	m_14 = 0;
	m_15 = 0;
	m_2 = 0;
	m_3 = 0;
	m_4 = 0;
	m_5 = 0;
	m_6 = 0;
	m_7 = 0;
	m_8 = 0;
	m_9 = 0;
	m_16a = 0.0;
	//}}AFX_DATA_INIT
}
```

// Limiting variables values

```
void CMR1A1Dlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CMR1A1Dlg)
	DDX_Control(pDX, IDC_EDIT16a, m_16c);
	DDX_Text(pDX, IDC_EDIT1, m_1);
	DDV_MinMaxInt(pDX, m_1, 0, 60);
	DDX_Text(pDX, IDC_EDIT10, m_10);
	DDV_MinMaxInt(pDX, m_10, 0, 60);
	DDX_Text(pDX, IDC_EDIT11, m_11);
	DDV_MinMaxInt(pDX, m_11, 0, 60);
	DDX_Text(pDX, IDC_EDIT12, m_12);
	DDV_MinMaxInt(pDX, m_12, 0, 60);
	DDX_Text(pDX, IDC_EDIT13, m_13);
	DDV_MinMaxInt(pDX, m_13, 0, 60);
	DDX_Text(pDX, IDC_EDIT14, m_14);
	DDV_MinMaxInt(pDX, m_14, 0, 60);
	DDX_Text(pDX, IDC_EDIT15, m_15);
	DDV_MinMaxInt(pDX, m_15, 0, 60);
	DDX_Text(pDX, IDC_EDIT2, m_2);
	DDV_MinMaxInt(pDX, m_2, 0, 60);
	DDX_Text(pDX, IDC_EDIT3, m_3);
	DDV_MinMaxInt(pDX, m_3, 0, 60);
	DDX_Text(pDX, IDC_EDIT4, m_4);
```

```
	DDV_MinMaxInt(pDX, m_4, 0, 60);
	DDX_Text(pDX, IDC_EDIT5, m_5);
	DDV_MinMaxInt(pDX, m_5, 0, 60);
	DDX_Text(pDX, IDC_EDIT6, m_6);
	DDV_MinMaxInt(pDX, m_6, 0, 60);
	DDX_Text(pDX, IDC_EDIT7, m_7);
	DDV_MinMaxInt(pDX, m_7, 0, 60);
	DDX_Text(pDX, IDC_EDIT8, m_8);
	DDV_MinMaxInt(pDX, m_8, 0, 60);
	DDX_Text(pDX, IDC_EDIT9, m_9);
	DDV_MinMaxInt(pDX, m_9, 0, 60);
	DDX_Text(pDX, IDC_EDIT16a, m_16a);
	//}}AFX_DATA_MAP
}
```

// Defining user-actions vis-a-vis system messages

```
BEGIN_MESSAGE_MAP(CMR1A1Dlg, CDialog)
	//{{AFX_MSG_MAP(CMR1A1Dlg)
	ON_BN_CLICKED(IDMR1A1P, OnMr1a1p)
	ON_BN_CLICKED(IDMR1A1N, OnMr1a1n)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

// Class method – Navigate to previous dialog

```
void CMR1A1Dlg::OnMr1a1p()
{
	EndDialog(IDCANCEL);
	CMR1ADlg a;
	a.DoModal();
}
```

// Class method – Compute validated mission time and navigate to next dialog

```
void CMR1A1Dlg::OnMr1a1n()
{
	UpdateData(TRUE);
	mt=0;
	if(m_1 > 0)
	mt += (m_1*3600);
	if(m_2 > 0)
	mt += (m_2*60);
	if(m_3 > 0)
	mt += (m_3);
	if(m_4 > 0)
	mt += (m_4*3600);
	if(m_5 > 0)
	mt += (m_5*60);
	if(m_6 > 0)
	mt += (m_6);
	if(m_7 > 0)
	mt += (m_7*3600);
	if(m_8 > 0)
	mt += (m_8*60);
	if(m_9 > 0)
	mt += (m_9);
	if(m_10 > 0)
	mt += (m_10*3600);
	if(m_11 > 0)
	mt += (m_11*60);
	if(m_12 > 0)
	mt += (m_12);
```

```
	if(m_13 > 0)
	mt += (m_13*3600);
	if(m_14 > 0)
	mt += (m_14*60);
	if(m_15 > 0)
	mt += (m_15);
	if(mt>0 && m_1<60 && m_2<60 && m_3<60 && m_4<60 && m_5<60 && m_6<60 &&
	m_7<60 && m_8<60 && m_9<60 && m_10<60 && m_11<60 && m_12<60 && m_13<60
	&& m_14<60 && m_15<60)
	{
		if(MessageBox("Please verify the entered values after auto-validation. Ensure all
		input boxes have value in [0,60]. Do you want proceed further?", "Verify the Entered
		Values" , MB_YESNO | MB_ICONQUESTION |MB_DEFBUTTON1 ) == IDYES)
		{
			m_16c.EnableWindow(TRUE);
			m_16a = mt;
			UpdateData(FALSE);
			m_16c.EnableWindow(FALSE);
			CString y;
			y.Format("%lf",mt);
			AfxMessageBox(" The computed mission time is "+y +"
			seconds");EndDialog(IDCANCEL);
			CUavdesignfcsDlg a;
			a.DoModal();
		}
	}
	else
	MessageBox("Please enter time in correct format","ZERO VALUE - EXCEED VALUE
	ERROR");
}
```

J.3 Payload class

The program code of 'Payload' class based on the acoustic-sensor and communication components of mission systems involves addition of selected mission systems weights to compute the payload weight as follows:

```
// File name: DPDlg.cpp
// Declaration of header files
#include "stdafx.h"
#include "VTUAV_Expert_System.h"
#include "DPDlg.h"
#include "Uavdesignfcs1Dlg.h"
#include "DPDlg1.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

// Declaration & initialisation of class variables
CDPDlg::CDPDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CDPDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CDPDlg)
	//}}AFX_DATA_INIT
}
```

```
// Limiting variables values
void CDPDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CDPDlg)
	//}}AFX_DATA_MAP
}

// Defining user-actions vis-a-vis system messages
BEGIN_MESSAGE_MAP(CDPDlg, CDialog)
	//{{AFX_MSG_MAP(CDPDlg)
	ON_BN_CLICKED(IDP1P, OnP1p)
	ON_BN_CLICKED(IDP1N, OnP1n)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

// Class method – Dialog initialisation
BOOL CDPDlg::OnInitDialog()
{
	CDialog::OnInitDialog();
	extern double pwt1;//si unit payload weight
	CString a;
	a.Format("%lf", pwt1);
	MessageBox("The following equipment and instrument capabilities should be installed on the
	UAV and/or be available to the supervising controller in order to comply with the
	requirements for safe flight under IFR procedures: Position Lights, Anti-Collision Lights,
	Transponder, Radios, Navigation Systems, UAV System and Altitude Displays, Flight and
	Voice Recorder, Built-in Test mechanisms. You may now select the payload as per your
	requirements. Your initially estimated payload weight was "+a+" kg", "SAFETY
	REQUIREMENTS - PAYLOAD DESIGN", MB_OK | MB_ICONINFORMATION
	|MB_DEFBUTTON1);

	return TRUE;  // return TRUE unless you set the focus to a control
		// EXCEPTION: OCX Property Pages should return FALSE
}

// Class method – Navigate to previous dialog
void CDPDlg::OnP1p()
{
	// TODO: Add your control notification handler code here
	EndDialog(IDCANCEL);
	CUavdesignfcs1Dlg a(this);
	a.DoModal();

}

// Class method – Aggregate selected mission systems weight and navigate to next dialog
void CDPDlg::OnP1n()
{
	extern int n;// no. of checkboxes
	n=0;
	extern double pa;//payload
	pa=0;

	//INITIALISE Array to Null
	extern char* payloadname_final[74];
	extern double payloadwt_final[74];

	for(int i=0;i<=73;i++)
```

```
	{
		payloadname_final[i]="";
		payloadwt_final[i]=0.0;
	}

	//obviously assign null to [0] value
	payloadname_final[0]="";payloadwt_final[0]=0.0;


	if(IsDlgButtonChecked(IDC_CHECKBOX1)){
	pa+=1.3;n+=1;payloadname_final[1]="SENSOR ARRAY";payloadwt_final[1]=1.3;}
	if(IsDlgButtonChecked(IDC_CHECKBOX2)){
	pa+=2.3;n+=1;payloadname_final[2]="TRANSPONDER";payloadwt_final[2]=2.3;}
	if(IsDlgButtonChecked(IDC_CHECKBOX3)){
	pa+=4.0;n+=1;payloadname_final[3]="TRANSPONDER";payloadwt_final[3]=4.0;}
	if(IsDlgButtonChecked(IDC_CHECKBOX4)){
	pa+=4.5;n+=1;payloadname_final[4]="TRANSPONDER";payloadwt_final[4]=4.5;}
	if(IsDlgButtonChecked(IDC_CHECKBOX5)){
	pa+=4.0;n+=1;payloadname_final[5]="TRANSPONDER";payloadwt_final[5]=4.0;}
	if(IsDlgButtonChecked(IDC_CHECKBOX6)){
	pa+=4.2;n+=1;payloadname_final[6]="TRANSPONDER";payloadwt_final[6]=4.2;}
	if(IsDlgButtonChecked(IDC_CHECKBOX7)){
	pa+=3.0;n+=1;payloadname_final[7]="TRANSPONDER";payloadwt_final[7]=3.0;}
	if(IsDlgButtonChecked(IDC_CHECKBOX8)){
	pa+=5.0;n+=1;payloadname_final[8]="TRANSMITTER";payloadwt_final[8]=5.0;}
	if(IsDlgButtonChecked(IDC_CHECKBOX9)){
	pa+=4.0;n+=1;payloadname_final[9]="TRANSMITTER";payloadwt_final[9]=4.0;}
	if(IsDlgButtonChecked(IDC_CHECKBOX10)){
	pa+=2.5;n+=1;payloadname_final[10]="TERMINAL";payloadwt_final[10]=2.5;}
	if(IsDlgButtonChecked(IDC_CHECKBOX11)){
	pa+=1.0;n+=1;payloadname_final[11]="TERMINAL";payloadwt_final[11]=1.0;}
	if(IsDlgButtonChecked(IDC_CHECKBOX12)){
	pa+=15.0;n+=1;payloadname_final[12]="RECORDER";payloadwt_final[12]=15.0;}
	if(IsDlgButtonChecked(IDC_CHECKBOX13)){
	pa+=4.0;n+=1;payloadname_final[13]="RECORDER";payloadwt_final[13]=4.0;}
	if(IsDlgButtonChecked(IDC_CHECKBOX14)){
	pa+=45.0;n+=1;payloadname_final[14]="INTERCEPTOR";payloadwt_final[14]=45.0;}

	CString a;
	a.Format("%lf",pa);
	CString num;
	num.Format("%d", n);
	if(MessageBox("Please ensure that you have double clicked the checkboxes. As of now you
	have selected "+num+" different payload checkboxes and their total weight is "+a+" kg. Do
	you want to proceed further?","Verify and proceed Ahead" , MB_YESNO |
	MB_ICONQUESTION  |MB_DEFBUTTON1 ) == IDYES)
	{
		EndDialog(IDCANCEL);
		CDPDlg1 a(this);
		a.DoModal();
	}
}
```

J.4 ‘Database - I’ class

The programme code of ‘Database-I’ class involves identification of VTUAVs based on the design requirements for estimation of power as follows:

```
// File name:  DatabaseSearchDLG.cpp : Implementation file
// Inclusion of header files
#include "stdafx.h"
#include "VTUAV_Expert_System.h"
#include "DatabaseSearchDLG.h"
#include "GSDlg.h"
#include "_recordset.h"
#include "7_1Dlg.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

// Declaration & initialisation of class variables
CDatabaseSearchDLG::CDatabaseSearchDLG(CWnd* pParent /*=NULL*/)
	: CDialog(CDatabaseSearchDLG::IDD, pParent)
{
	//{{AFX_DATA_INIT(CDatabaseSearchDLG)
	//}}AFX_DATA_INIT
}

// Limiting variables values
void CDatabaseSearchDLG::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CDatabaseSearchDLG)
	DDX_Control(pDX, IDC_BUTTONPFG1, m_pfg1);
	DDX_Control(pDX, IDC_BUTTONPFG, m_pfg);
	DDX_Control(pDX, IDC_EDIT9c, m_9c);
	DDX_Control(pDX, IDC_EDIT9b, m_9b);
	DDX_Control(pDX, IDC_EDIT9a, m_9a);
	DDX_Control(pDX, IDC_EDIT9, m_9);
	DDX_Control(pDX, IDC_EDIT8, m_8);
	DDX_Control(pDX, IDC_EDIT7, m_7);
	DDX_Control(pDX, IDC_EDIT6, m_6);
	DDX_Control(pDX, IDC_EDIT5, m_5);
	DDX_Control(pDX, IDC_EDIT4, m_4);
	DDX_Control(pDX, IDC_EDIT3, m_3);
	DDX_Control(pDX, IDC_ADOSEARCH, m_adosearch);
	DDX_Control(pDX, IDC_DATAGRIDAS, m_dgas);
	DDX_Control(pDX, IDC_DATAGRIDDS, m_dgds);
	DDX_Control(pDX, IDC_ADOSEARCH2, m_adosearch2);
	//}}AFX_DATA_MAP
}

// Defining user-actions vis-à-vis system messages
BEGIN_MESSAGE_MAP(CDatabaseSearchDLG, CDialog)
	//{{AFX_MSG_MAP(CDatabaseSearchDLG)
	ON_BN_CLICKED(ID6B, On6b)
	ON_BN_CLICKED(ID6S, On6s)
	ON_BN_CLICKED(IDC_BUTTONPFG, OnButtonpfg)
	ON_BN_CLICKED(IDC_BUTTONPFG1, OnButtonpfg1)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()
// Class method – Initialisation of dialog
BOOL CDatabaseSearchDLG::OnInitDialog()
{
	CDialog::OnInitDialog();
	extern double ewt;//Empty weight
```

```
		extern double pwt;//Payload weight
		extern double lwt;//MTOW or launch weight
		extern double sp;//Forward speed
		extern double en;//Endurance
		extern double ra;//Range
		extern double al;//Altitude
		extern double le;//Fuselage length
		extern double sn;//Fuselage span or width
		extern double di;//Rotor diameter
		extern double sr;//Safety regulations
		extern double aw;//Stub wing or additional wing
		extern double fr;//Fuel reserve weight
		extern double mt;//Mission time
		extern double is;//Integrated system weight

		CString ewts, pwts, lwts, sps, ens, ras, als, les, sns, dis, srs, aws, frs, mts, iss;
		ewts.Format("%lf",ewt);
		pwts.Format("%lf",pwt);
		lwts.Format("%lf",lwt);
		sps.Format("%lf",sp);
		ens.Format("%lf",en);
		ras.Format("%lf",ra);
		als.Format("%lf",al);
		les.Format("%lf",le);
		sns.Format("%lf",sn);
		dis.Format("%lf",di);
		srs.Format("%lf",sr);
		aws.Format("%lf",aw);
		frs.Format("%lf",fr);
		mts.Format("%lf",mt);
		iss.Format("%lf",is);

		m_3.SetWindowText(lwts);
		m_4.SetWindowText(sps);
		m_5.SetWindowText(ens);
		m_6.SetWindowText(ras);
		m_7.SetWindowText(als);
		m_8.SetWindowText(les);
		m_9.SetWindowText(sns);
		m_9a.SetWindowText(dis);
		m_9b.SetWindowText(mts);
		m_9c.SetWindowText(iss);

		return TRUE;  // return TRUE unless you set the focus to a control
		// EXCEPTION: OCX Property Pages should return FALSE
}

// Class method – Navigate to previous dialog
void CDatabaseSearchDLG::On6b()
{
		EndDialog(IDCANCEL);
		CGSDlg a;
		a.DoModal();
}


// Class method – Search method
void CDatabaseSearchDLG::On6s()
{
//Determine the parameters
```

```
CString lwts,sps,ens,ras,als;
extern double lwt;
extern double sp;
extern double en;
extern double ra;
extern double al;
double wt_parameter;
double sp_parameter;
double en_parameter;
double ra_parameter;
double al_parameter;

//Weight limit
if (lwt <= 100)  wt_parameter = 12.5;
else if((lwt>100) && (lwt<=800) ) wt_parameter = 32;
else if((lwt>800) && (lwt<=1800) ) wt_parameter = 75;
else if((lwt>1800) && (lwt<=4000) ) wt_parameter = 280;
else  wt_parameter = 750;

//Speed limit
if (sp <= 62)  sp_parameter = 4;
else if((sp>62) && (sp<=75) ) sp_parameter = 7;
else if((sp>75) && (sp<=161) ) sp_parameter = 3;
else if((sp>161) && (sp<=300) ) sp_parameter = 10;
else if((sp>300) && (sp<=627) ) sp_parameter = 22;
else  sp_parameter = 220;

//Endurance limit
if (en <= 20)  en_parameter = 1;
else  en_parameter = 3.5;

//Range limit
if (ra <= 310)  ra_parameter = 10;
else if((ra>310) && (ra<=410) ) ra_parameter = 12.5;
else if((ra>410) && (ra<=823) ) ra_parameter = 35;
else if((ra>823) && (ra<=1356) ) ra_parameter = 70;
else if((ra>1356) && (ra<=2050) ) ra_parameter = 130;
else  ra_parameter = 510;

//Altitude limit
if (al <= 5000)  al_parameter = 500;
else if((al>5000) && (al<=21000) ) al_parameter = 750;
else if((al>21000) && (al<=26500) ) al_parameter = 1000;
else if((al>26500) && (al<=72182) ) al_parameter = 25000;
else  al_parameter = 75000;

CString query, query1;

//Pre-first ascending sort query
query = "SELECT TOP 5 NAME, WEIGHT, SPEED, ENDURANCE, RANGE, ALTITUDE, ENGINE, MISSION, LENGTH, WINGSPAN, BODYDIA FROM Querytable WHERE (WEIGHT BETWEEN ";

lwts.Format("%lf",lwt);
query+=lwts;
query+=" AND ";
lwts.Format("%lf",(lwt + wt_parameter));
query+=lwts;
query+=") AND (SPEED BETWEEN ";
sps.Format("%lf",sp);
```

```
query+=sps;
query+=" AND ";
sps.Format("%lf",(sp + sp_parameter));
query+=sps;
query+=") AND (ENDURANCE BETWEEN ";
ens.Format("%lf",en);
query+=ens;
query+=" AND ";
ens.Format("%lf",(en + en_parameter));
query+=ens;
query+=") AND (RANGE BETWEEN ";
ras.Format("%lf",ra);
query+=ras;
query+=" AND ";
ras.Format("%lf",(ra + ra_parameter));
query+=ras;
query+=") AND (ALTITUDE BETWEEN ";
als.Format("%lf",al);
query+=als;
query+=" AND ";
als.Format("%lf",(al + al_parameter));
query+=als;
query+=") ORDER BY WEIGHT, SPEED, ENDURANCE, RANGE, ALTITUDE;";

//Pre-first descending sort query
query1 = "SELECT TOP 5 NAME, WEIGHT, SPEED, ENDURANCE, RANGE, ALTITUDE, ENGINE, MISSION, LENGTH, WINGSPAN, BODYDIA FROM Querytable WHERE (WEIGHT BETWEEN ";

lwts.Format("%lf",(lwt - wt_parameter));
query1+=lwts;
query1+=" AND ";
lwts.Format("%lf",lwt);
query1+=lwts;
query1+=") AND (SPEED BETWEEN ";
sps.Format("%lf",(sp - sp_parameter));
query1+=sps;
query1+=" AND ";
sps.Format("%lf",sp);
query1+=sps;
query1+=") AND (ENDURANCE BETWEEN ";
ens.Format("%lf",(en - en_parameter));
query1+=ens;
query1+=" AND ";
ens.Format("%lf",en);
query1+=ens;
query1+=") AND (RANGE BETWEEN ";
ras.Format("%lf",(ra - ra_parameter));
query1+=ras;
query1+=" AND ";
ras.Format("%lf",ra);
query1+=ras;
query1+=") AND (ALTITUDE BETWEEN ";
als.Format("%lf",(al - al_parameter));
query1+=als;
query1+=" AND ";
als.Format("%lf",al);
query1+=als;
query1+=") ORDER BY WEIGHT DESC, SPEED DESC, ENDURANCE DESC, RANGE DESC, ALTITUDE DESC";
```

```
//Set the record-source and update
m_adosearch.SetRecordSource(query);
m_adosearch.Refresh();
m_adosearch2.SetRecordSource(query1);
m_adosearch2.Refresh();
if(m_adosearch.GetRecordset().GetRecordCount() == 0 && m_adosearch2.GetRecordset().
GetRecordCount() == 0)
{
        //First ascending sort query involving /weight, speed, endurance, range
        query="";
        query1="";
        query = "SELECT TOP 5 NAME, WEIGHT, SPEED, ENDURANCE, RANGE, ALTITUDE,
        ENGINE, MISSION, LENGTH, WINGSPAN, BODYDIA FROM Querytable WHERE
        (WEIGHT BETWEEN ";

        lwts.Format("%lf",lwt);
        query+=lwts;
        query+=" AND ";
        lwts.Format("%lf",(lwt + wt_parameter));
        query+=lwts;
        query+=") AND (SPEED BETWEEN ";
        sps.Format("%lf",sp);
        query+=sps;
        query+=" AND ";
        sps.Format("%lf",(sp + sp_parameter));
        query+=sps;
        query+=") AND (ENDURANCE BETWEEN ";
        ens.Format("%lf",en);
        query+=ens;
        query+=" AND ";
        ens.Format("%lf",(en + en_parameter));
        query+=ens;
        query+=") AND (RANGE BETWEEN ";
        ras.Format("%lf",ra);
        query+=ras;
        query+=" AND ";
        ras.Format("%lf",(ra + ra_parameter));
        query+=ras;
        query+=") ORDER BY WEIGHT, SPEED, ENDURANCE, RANGE, ALTITUDE ";

        //First descending sort query
        query1 = "SELECT TOP 5 NAME, WEIGHT, SPEED, ENDURANCE, RANGE,
        ALTITUDE, ENGINE, MISSION, LENGTH, WINGSPAN, BODYDIA FROM Querytable
        WHERE (WEIGHT BETWEEN ";

        lwts.Format("%lf",(lwt - wt_parameter));
        query1+=lwts;
        query1+=" AND ";
        lwts.Format("%lf",lwt);
        query1+=lwts;
        query1+=") AND (SPEED BETWEEN ";
        sps.Format("%lf",(sp - sp_parameter));
        query1+=sps;
        query1+=" AND ";
        sps.Format("%lf",sp);
        query1+=sps;
        query1+=") AND (ENDURANCE BETWEEN ";
        ens.Format("%lf",(en - en_parameter));
        query1+=ens;
```

```
query1+=" AND ";
ens.Format("%lf",en);
query1+=ens;
query1+=") AND (RANGE BETWEEN ";
ras.Format("%lf",(ra - ra_parameter));
query1+=ras;
query1+=" AND ";
ras.Format("%lf",ra);
query1+=ras;
query1+=") ORDER BY WEIGHT DESC, SPEED DESC, ENDURANCE DESC, RANGE
DESC, ALTITUDE DESC";

//Set the record-source and update
m_adosearch.SetRecordSource(query);
m_adosearch.Refresh();
m_adosearch2.SetRecordSource(query1);
m_adosearch2.Refresh();

//Second query involving weight, speed and endurance
if(m_adosearch.GetRecordset().GetRecordCount() == 0 &&
m_adosearch2.GetRecordset().GetRecordCount() == 0)
{
        query="";
        query1="";

        //Second ascending sort query
        query = "SELECT TOP 5 NAME, WEIGHT, SPEED, ENDURANCE, RANGE,
        ALTITUDE, ENGINE, MISSION, LENGTH, WINGSPAN, BODYDIA FROM
        Querytable WHERE (WEIGHT BETWEEN ";

        lwts.Format("%lf",lwt);
        query+=lwts;
        query+=" AND ";
        lwts.Format("%lf",(lwt + wt_parameter));
        query+=lwts;
        query+=") AND (SPEED BETWEEN ";
        sps.Format("%lf",sp);
        query+=sps;
        query+=" AND ";
        sps.Format("%lf",(sp + sp_parameter));
        query+=sps;
        query+=") AND (ENDURANCE BETWEEN ";
        ens.Format("%lf",en);
        query+=ens;
        query+=" AND ";
        ens.Format("%lf",(en + en_parameter));
        query+=ens;
        query+=") ORDER BY WEIGHT, SPEED, ENDURANCE, RANGE, ALTITUDE ";

        //Second descending sort query
        query1 = "SELECT TOP 5 NAME, WEIGHT, SPEED, ENDURANCE, RANGE,
        ALTITUDE, ENGINE, MISSION, LENGTH, WINGSPAN, BODYDIA FROM
        Querytable WHERE (WEIGHT BETWEEN ";

        lwts.Format("%lf",(lwt - wt_parameter));
        query1+=lwts;
        query1+=" AND ";
        lwts.Format("%lf",lwt);
        query1+=lwts;
        query1+=") AND (SPEED BETWEEN ";
```

```
sps.Format("%lf",(sp - sp_parameter));
query1+=sps;
query1+=" AND ";
sps.Format("%lf",sp);
query1+=sps;
query1+=") AND (ENDURANCE BETWEEN ";
ens.Format("%lf",(en - en_parameter));
query1+=ens;
query1+=" AND ";
ens.Format("%lf",en);
query1+=ens;
query1+=") ORDER BY WEIGHT DESC, SPEED DESC, ENDURANCE DESC,
RANGE DESC, ALTITUDE DESC";

m_adosearch.SetRecordSource(query);
m_adosearch.Refresh();
m_adosearch2.SetRecordSource(query1);
m_adosearch2.Refresh();

//Weight and speed
if(m_adosearch.GetRecordset().GetRecordCount() == 0 &&
m_adosearch2.GetRecordset().GetRecordCount() == 0)
{
        query="";
        query1="";

        //Third ascending sort query
        query = "SELECT TOP 5 NAME, WEIGHT, SPEED, ENDURANCE,
        RANGE, ALTITUDE, ENGINE, MISSION, LENGTH, WINGSPAN,
        BODYDIA FROM Querytable WHERE (WEIGHT BETWEEN ";

        lwts.Format("%lf",lwt);
        query+=lwts;
        query+=" AND ";
        lwts.Format("%lf",(lwt + wt_parameter));
        query+=lwts;
        query+=") AND (SPEED BETWEEN ";
        sps.Format("%lf",sp);
        query+=sps;
        query+=" AND ";
        sps.Format("%lf",(sp + sp_parameter));
        query+=sps;
        query+=") ORDER BY WEIGHT, SPEED, ENDURANCE, RANGE,
        ALTITUDE ";


        //Third descending sort query
        query1 = "SELECT TOP 5 NAME, WEIGHT, SPEED, ENDURANCE,
        RANGE, ALTITUDE, ENGINE, MISSION, LENGTH, WINGSPAN,
        BODYDIA FROM Querytable WHERE (WEIGHT BETWEEN ";

        lwts.Format("%lf",(lwt - wt_parameter));
        query1+=lwts;
        query1+=" AND ";
        lwts.Format("%lf",lwt);
        query1+=lwts;
        query1+=") AND (SPEED BETWEEN ";
        sps.Format("%lf",(sp - sp_parameter));
        query1+=sps;
        query1+=" AND ";
```

```
                    sps.Format("%lf",sp);
                    query1+=sps;
                    query1+=") ORDER BY WEIGHT DESC, SPEED DESC, ENDURANCE
                    DESC, RANGE DESC, ALTITUDE DESC";

                    m_adosearch.SetRecordSource(query);
                    m_adosearch.Refresh();
                    m_adosearch2.SetRecordSource(query1);
                    m_adosearch2.Refresh();

                    //Only weight search
                    if(m_adosearch.GetRecordset().GetRecordCount() == 0 &&
                    m_adosearch2.GetRecordset().GetRecordCount() == 0)
                    {
                            query="";
                            query1="";

                            //Fourth ascending sort query
                            query = "SELECT TOP 5 NAME, WEIGHT, SPEED,
                            ENDURANCE, RANGE, ALTITUDE, ENGINE, MISSION,
                            LENGTH, WINGSPAN, BODYDIA FROM Querytable WHERE
                            (WEIGHT BETWEEN ";

                            lwts.Format("%lf",lwt);
                            query+=lwts;
                            query+=" AND ";
                            lwts.Format("%lf",(lwt + wt_parameter));
                            query+=lwts;
                            query+=") ORDER BY WEIGHT, SPEED, ENDURANCE,
                            RANGE, ALTITUDE ";

                            //Fourth descending sort query
                            query1 = "SELECT TOP 5 NAME, WEIGHT, SPEED,
                            ENDURANCE, RANGE, ALTITUDE, ENGINE, MISSION,
                            LENGTH, WINGSPAN, BODYDIA FROM Querytable WHERE
                            (WEIGHT BETWEEN ";

                            lwts.Format("%lf",(lwt - wt_parameter));
                            query1+=lwts;
                            query1+=" AND ";
                            lwts.Format("%lf",lwt);
                            query1+=lwts;
                            query1+=") ORDER BY WEIGHT DESC, SPEED DESC,
                            ENDURANCE DESC, RANGE DESC, ALTITUDE DESC";

                            m_adosearch.SetRecordSource(query);
                            m_adosearch.Refresh();
                            m_adosearch2.SetRecordSource(query1);
                            m_adosearch2.Refresh();

                    }

            }

    }

}

UpdateData(FALSE);
```

```
if(m_adosearch.GetRecordset().GetRecordCount() > 0 &&
m_adosearch2.GetRecordset().GetRecordCount() > 0)
{
	m_pfg.EnableWindow(TRUE);
	m_pfg.ShowWindow(SW_SHOW);
	m_pfg1.EnableWindow(TRUE);
	m_pfg1.ShowWindow(SW_SHOW);
	MessageBox("Results have been found on both data lists. The program pick is
	"+m_dgas.GetText() +" UAV. However, you may override the default selection by clicking on
	the name of the aircraft(column 1) that you want to select from the two data grids and then
	click the corresponding 'Pick From Grid' button", "State your preference");
}

//list 1 result chosen automatically
else if (m_adosearch.GetRecordset().GetRecordCount() > 0 )
{
	m_pfg.EnableWindow(TRUE);
	m_pfg.ShowWindow(SW_SHOW);
	MessageBox("Results have been found in first datalist. The program pick is
	"+m_dgas.GetText() +" UAV. Please click on the name of the aircraft(column 1)that you want
	to select from the first datagrid and then click the corresponding 'Pick From Grid' button",
	"State your preference");
}

//list 2 result chosen automatically
else if (m_adosearch2.GetRecordset().GetRecordCount() > 0 )
{
	m_pfg1.EnableWindow(TRUE);
	m_pfg1.ShowWindow(SW_SHOW);
	MessageBox("Results have been found in second datalist. The program pick is
	"+m_dgds.GetText() +" UAV. Please click on the name of the aircraft(column 1)that you want
	to select from the second datagrid and then click the corresponding 'Pick From Grid' button",
	"State your preference");
}

//no records matched
else
{
	MessageBox("No results found for the input parameters. Change the values all over again.",
	"Unprecedented Aircraft Parameters");
}

}
```

// Class method – Selection method (Algorithm 1)

```
void CDatabaseSearchDLG::OnButtonpfg()
{
	extern CString uavname_dbsearch;
	extern CString uavmission_dbsearch;
	extern CString uavengine_dbsearch;
	extern double uavweight_dbsearch;
	extern double uavpower_dbsearch;
	extern double uavspeed_dbsearch;
	extern double uavendurance_dbsearch;
	extern double uavrange_dbsearch;
	extern double uavaltitude_dbsearch;
	extern double uavlength_dbsearch;
	extern double uavwingspan_dbsearch;
	extern double uavrotorspan_dbsearch;
	extern double uavdiameter_dbsearch;
```

```
//Pointer to ending  number in a string
char  *stopstring;
uavname_dbsearch = m_dgas.GetText();
CString query_check;
query_check = "SELECT * FROM Querytable WHERE NAME ='";
query_check+=m_dgas.GetText();
query_check+="'";
m_adosearch.SetRecordSource(query_check);
  m_adosearch.Refresh();

if(m_adosearch.GetRecordset().GetRecordCount() > 0)
{
        query_check = "SELECT MISSION FROM Querytable WHERE NAME ='";
        query_check+=m_dgas.GetText();
        query_check+="'";
        m_adosearch2.SetRecordSource(query_check);
        m_adosearch2.Refresh();
        uavmission_dbsearch = m_dgds.GetText();

        /Results not displayed immediately but used later
        query_check = "SELECT ENGINE FROM Querytable WHERE NAME ='";
        query_check+=m_dgas.GetText();
        query_check+="'";
        m_adosearch2.SetRecordSource(query_check);
        m_adosearch2.Refresh();
        uavengine_dbsearch = m_dgds.GetText();

        query_check = "SELECT WEIGHT FROM Querytable WHERE NAME ='";
        query_check+=m_dgas.GetText();
        query_check+="'";
        m_adosearch2.SetRecordSource(query_check);
        m_adosearch2.Refresh();
        uavweight_dbsearch = strtod(m_dgds.GetText(),&stopstring);

        query_check = "SELECT POWER FROM Querytable WHERE NAME ='";
        query_check+=m_dgas.GetText();
        query_check+="'";
        m_adosearch2.SetRecordSource(query_check);
        m_adosearch2.Refresh();
        uavpower_dbsearch = strtod(m_dgds.GetText(),&stopstring);

        query_check = "SELECT SPEED FROM Querytable WHERE NAME ='";
        query_check+=m_dgas.GetText();
        query_check+="'";
        m_adosearch2.SetRecordSource(query_check);
        m_adosearch2.Refresh();
        uavspeed_dbsearch = strtod(m_dgds.GetText(),&stopstring);

        query_check = "SELECT ENDURANCE FROM Querytable WHERE NAME ='";
        query_check+=m_dgas.GetText();
        query_check+="'";
        m_adosearch2.SetRecordSource(query_check);
        m_adosearch2.Refresh();
        uavendurance_dbsearch = strtod(m_dgds.GetText(),&stopstring);

        query_check = "SELECT RANGE FROM Querytable WHERE NAME ='";
        query_check+=m_dgas.GetText();
        query_check+="'";
        m_adosearch2.SetRecordSource(query_check);
```

```
		m_adosearch2.Refresh();
		uavrange_dbsearch = strtod(m_dgds.GetText(),&stopstring);

		query_check = "SELECT ALTITUDE FROM Querytable WHERE NAME ='";
		query_check+=m_dgas.GetText();
		query_check+="'";
		m_adosearch2.SetRecordSource(query_check);
		m_adosearch2.Refresh();
		uavaltitude_dbsearch = strtod(m_dgds.GetText(),&stopstring);

		CString suavweight_dbsearch;
		CString suavpower_dbsearch;
		CString suavspeed_dbsearch;
		CString suavendurance_dbsearch;
		CString suavrange_dbsearch;
		CString suavaltitude_dbsearch;

		suavweight_dbsearch.Format("%lf",uavweight_dbsearch);
		suavpower_dbsearch.Format("%lf",uavpower_dbsearch);
		suavspeed_dbsearch.Format("%lf",uavspeed_dbsearch);
		suavendurance_dbsearch.Format("%lf",uavendurance_dbsearch);
		suavrange_dbsearch.Format("%lf",uavrange_dbsearch);
		suavaltitude_dbsearch.Format("%lf",uavaltitude_dbsearch);

		MessageBox("The selection is as follows. Name = '"+uavname_dbsearch +"',
		Mission='"+uavmission_dbsearch +"', Weight(lbs)='"+suavweight_dbsearch
		+"',Power(hp)='"+suavpower_dbsearch +"', Speed(miles/hr)='"+suavspeed_dbsearch
		+"', Range(miles)='"+suavrange_dbsearch +"',
		Endurance(hour)='"+suavendurance_dbsearch
		+"',Altitude(ft.)='"+suavaltitude_dbsearch +"'.","Retrieved Success");

		EndDialog(IDCANCEL);
		C7_1Dlg a;
		a.DoModal();
	}

	else
	{
		MessageBox("Please select the name field only. You may have to click the search
		button once more.","Warning");
	}

}

// Class method – Selection method (Algorithm 2)
void CDatabaseSearchDLG::OnButtonpfg1()
{
	extern CString uavname_dbsearch;
	extern CString uavmission_dbsearch;
	extern CString uavengine_dbsearch;
	extern double uavweight_dbsearch;
	extern double uavpower_dbsearch;
	extern double uavspeed_dbsearch;
	extern double uavendurance_dbsearch;
	extern double uavrange_dbsearch;
	extern double uavaltitude_dbsearch;
	extern double uavlength_dbsearch;
	extern double uavwingspan_dbsearch;
	extern double uavrotorspan_dbsearch;
	extern double uavdiameter_dbsearch;
```

```
//Pointer to ending number in a string
char  *stopstring;
CString suavweight_dbsearch;
CString suavpower_dbsearch;
CString suavspeed_dbsearch;
CString suavendurance_dbsearch;
CString suavrange_dbsearch;
CString suavaltitude_dbsearch;

uavname_dbsearch = m_dgds.GetText();
CString query_check;
query_check = "SELECT * FROM Querytable WHERE NAME ='";
query_check+=m_dgds.GetText();
query_check+="'";
m_adosearch2.SetRecordSource(query_check);
m_adosearch2.Refresh();

if(m_adosearch2.GetRecordset().GetRecordCount() > 0)
{
	query_check = "SELECT MISSION FROM Querytable WHERE NAME ='";
	query_check+=m_dgds.GetText();
	query_check+="'";
	m_adosearch.SetRecordSource(query_check);
	m_adosearch.Refresh();
	uavmission_dbsearch = m_dgas.GetText();

	//Results not displayed immediately but used later
	query_check = "SELECT ENGINE FROM Querytable WHERE NAME ='";
	query_check+=m_dgds.GetText();
	query_check+="'";
	m_adosearch.SetRecordSource(query_check);
	m_adosearch.Refresh();
	uavengine_dbsearch = m_dgas.GetText();

	query_check = "SELECT WEIGHT FROM Querytable WHERE NAME ='";
	query_check+=m_dgds.GetText();
	query_check+="'";
	m_adosearch.SetRecordSource(query_check);
	m_adosearch.Refresh();
	uavweight_dbsearch = strtod(m_dgas.GetText(),&stopstring);

	query_check = "SELECT POWER FROM Querytable WHERE NAME ='";
	query_check+=m_dgds.GetText();
	query_check+="'";
	m_adosearch.SetRecordSource(query_check);
	m_adosearch.Refresh();
	uavpower_dbsearch = strtod(m_dgas.GetText(),&stopstring);

	query_check = "SELECT SPEED FROM Querytable WHERE NAME ='";
	query_check+=m_dgds.GetText();
	query_check+="'";
	m_adosearch.SetRecordSource(query_check);
	m_adosearch.Refresh();
	uavspeed_dbsearch = strtod(m_dgas.GetText(),&stopstring);

	query_check = "SELECT ENDURANCE FROM Querytable WHERE NAME ='";
	query_check+=m_dgds.GetText();
	query_check+="'";
	m_adosearch.SetRecordSource(query_check);
```

```
                m_adosearch.Refresh();
                uavendurance_dbsearch = strtod(m_dgas.GetText(),&stopstring);

                query_check = "SELECT RANGE FROM Querytable WHERE NAME ='";
                query_check+=m_dgds.GetText();
                query_check+="'";
                m_adosearch.SetRecordSource(query_check);
                m_adosearch.Refresh();
                uavrange_dbsearch = strtod(m_dgas.GetText(),&stopstring);

                query_check = "SELECT ALTITUDE FROM Querytable WHERE NAME ='";
                query_check+=m_dgds.GetText();
                query_check+="'";
                m_adosearch.SetRecordSource(query_check);
                m_adosearch.Refresh();
                uavaltitude_dbsearch = strtod(m_dgas.GetText(),&stopstring);

                suavweight_dbsearch.Format("%lf",uavweight_dbsearch);
                suavpower_dbsearch.Format("%lf",uavpower_dbsearch);
                suavspeed_dbsearch.Format("%lf",uavspeed_dbsearch);
                suavendurance_dbsearch.Format("%lf",uavendurance_dbsearch);
                suavrange_dbsearch.Format("%lf",uavrange_dbsearch);
                suavaltitude_dbsearch.Format("%lf",uavaltitude_dbsearch);

                MessageBox("The selection is as follows. Name = '"+uavname_dbsearch +"',
                Mission='"+uavmission_dbsearch +"', Weight(lbs)='"+suavweight_dbsearch
                +"',Power(hp)='"+suavpower_dbsearch +"', Speed(miles/hr)='"+suavspeed_dbsearch
                +"', Range(miles)='"+suavrange_dbsearch +"',
                Endurance(hour)='"+suavendurance_dbsearch
                +"',Altitude(ft.)='"+suavaltitude_dbsearch +"'.","Retrieved Success");

                EndDialog(IDCANCEL);
                C7_1Dlg a;
                a.DoModal();
        }

        else
        {
                MessageBox("Please select the name field only. You may have to click the search
                button once more.","Warning");
        }

}
```

J.5 ‘Fuel’, ‘Structural weight’, ‘Vertical drag’, & ‘Rotor system’ classes

The fuel, structural weight, vertical drag and rotor system classes involve the evaluation of fuel, structural weight, vertical drag, and number of rotor blades based on the coaxial configuration. These classes are programmed in a single dialog to provide maximum information as follows:

```
// File name: 7_1Dlg.cpp
// Declaration of header files
```

```
#include "stdafx.h"
#include "VTUAV_Expert_System.h"
#include "7_1Dlg.h"
#include "MRDlg.h"
#include "7_2Dlg.h"
#include "InputDlg.h"
#include <cmath>   // equivalent of math.h
#include "DatabaseSearchDLG.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

// Declaration & initialisation of class variables
C7_1Dlg::C7_1Dlg(CWnd* pParent /*=NULL*/)
	: CDialog(C7_1Dlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(C7_1Dlg)
	m_1 = _T("");
	m_2 = _T("");
	m_3 = _T("");
	m_4 = _T("");
	m_5 = _T("");
	m_6 = _T("");
	m_10a = _T("");
	m_7 = _T("");
	m_8 = _T("");
	m_9 = _T("");
	m_6a = _T("");
	m_6a2 = _T("");
	m_6a3 = _T("");
	m_6a4 = _T("");
	m_11 = _T("");
	m_12 = _T("");
	m_13 = _T("");
	m_14 = _T("");
	m_15 = _T("");
	m_17 = _T("");
	m_18 = _T("");
	m_19 = _T("");
	m_20 = _T("");
	m_19a = _T("");
	m_21 = _T("");
	m_22 = _T("");
	m_23 = _T("");
	m_24 = _T("");
	m_25 = _T("");
	m_26 = _T("");
	m_27 = _T("");
	m_28 = _T("");
	m_6a5 = _T("");
	m_6a6 = _T("");
	m_6a7 = _T("");
	m_6a8 = _T("");
	//}}AFX_DATA_INIT
}

// Limiting variables values
void C7_1Dlg::DoDataExchange(CDataExchange* pDX)
```

```
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(C7_1Dlg)
	DDX_Text(pDX, IDC_EDIT1, m_1);
	DDX_Text(pDX, IDC_EDIT2, m_2);
	DDX_Text(pDX, IDC_EDIT3, m_3);
	DDX_Text(pDX, IDC_EDIT4, m_4);
	DDX_Text(pDX, IDC_EDIT5, m_5);
	DDX_Text(pDX, IDC_EDIT6, m_6);
	DDX_Text(pDX, IDC_EDIT10a, m_10a);
	DDX_Text(pDX, IDC_EDIT7, m_7);
	DDX_Text(pDX, IDC_EDIT8, m_8);
	DDX_Text(pDX, IDC_EDIT9, m_9);
	DDX_Control(pDX, IDC_LABELFE, m_labelfr);
	DDX_Text(pDX, IDC_EDIT6a, m_6a);
	DDX_Text(pDX, IDC_EDIT6a2, m_6a2);
	DDX_Text(pDX, IDC_EDIT6a3, m_6a3);
	DDX_Text(pDX, IDC_EDIT6a4, m_6a4);
	DDX_Text(pDX, IDC_EDIT11, m_11);
	DDX_Text(pDX, IDC_EDIT12, m_12);
	DDX_Text(pDX, IDC_EDIT13, m_13);
	DDX_Text(pDX, IDC_EDIT14, m_14);
	DDX_Text(pDX, IDC_EDIT15, m_15);
	DDX_Text(pDX, IDC_EDIT17, m_17);
	DDX_Text(pDX, IDC_EDIT18, m_18);
	DDX_Text(pDX, IDC_EDIT19, m_19);
	DDX_Text(pDX, IDC_EDIT20, m_20);
	DDX_Text(pDX, IDC_EDIT19a, m_19a);
	DDX_Text(pDX, IDC_EDIT21, m_21);
	DDX_Text(pDX, IDC_EDIT22, m_22);
	DDX_Text(pDX, IDC_EDIT23, m_23);
	DDX_Text(pDX, IDC_EDIT24, m_24);
	DDX_Text(pDX, IDC_EDIT25, m_25);
	DDX_Text(pDX, IDC_EDIT26, m_26);
	DDX_Text(pDX, IDC_EDIT27, m_27);
	DDX_Text(pDX, IDC_EDIT28, m_28);
	DDX_Text(pDX, IDC_EDIT6a5, m_6a5);
	DDX_Text(pDX, IDC_EDIT6a6, m_6a6);
	DDX_Text(pDX, IDC_EDIT6a7, m_6a7);
	DDX_Text(pDX, IDC_EDIT6a8, m_6a8);
	//}}AFX_DATA_MAP
}

// Defining user-actions vis-a-vis system messages
BEGIN_MESSAGE_MAP(C7_1Dlg, CDialog)
	//{{AFX_MSG_MAP(C7_1Dlg)
	ON_BN_CLICKED(ID7B, On7b)
	ON_BN_CLICKED(ID7H1, On7h1)
	ON_BN_CLICKED(ID7H2, On7h2)
	ON_BN_CLICKED(ID7H3, On7h3)
	ON_BN_CLICKED(ID7H4, On7h4)
	ON_BN_CLICKED(ID7H5, On7h5)
	ON_BN_CLICKED(ID7H6, On7h6)
	ON_BN_CLICKED(ID7H7, On7h7)
	ON_BN_CLICKED(ID7H8, On7h8)
	ON_BN_CLICKED(ID7H9, On7h9)
	ON_BN_CLICKED(ID7H10, On7h10)
	ON_BN_CLICKED(ID7H11, On7h11)
	ON_BN_CLICKED(ID7H12, On7h12)
	ON_BN_CLICKED(ID7H13, On7h13)
```

```
	ON_BN_CLICKED(ID7H14, On7h14)
	ON_BN_CLICKED(ID7H15, On7h15)
	ON_BN_CLICKED(ID7H16, On7h16)
	ON_BN_CLICKED(ID7H17, On7h17)
	ON_BN_CLICKED(ID7H19, On7h19)
	ON_BN_CLICKED(ID7H20, On7h20)
	ON_BN_CLICKED(ID7H21, On7h21)
	ON_BN_CLICKED(ID7H22, On7h22)
	ON_BN_CLICKED(ID7H23, On7h23)
	ON_BN_CLICKED(ID7H24, On7h24)
	ON_BN_CLICKED(ID7H6a, On7H6a)
	ON_BN_CLICKED(ID7N, On7n)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

// Class method – Initialisation of dialog
BOOL C7_1Dlg::OnInitDialog()
{
	CDialog::OnInitDialog();
	extern CString uavname_dbsearch;
	extern CString uavmission_dbsearch;
	extern CString uavengine_dbsearch;
	extern double uavweight_dbsearch;
	extern double uavpower_dbsearch;
	extern double uavspeed_dbsearch;
	extern double uavendurance_dbsearch;
	extern double uavrange_dbsearch;
	extern double uavaltitude_dbsearch;
	extern double uavlength_dbsearch;
	extern double uavwingspan_dbsearch;
	extern double uavrotorspan_dbsearch;
	extern double uavdiameter_dbsearch;

	extern double ewt;//Empty weight
	extern double pwt;//Payload weight
	extern double lwt;//MTOW or launch weight
	extern double mt;//Mission time
	extern double fr;//Fuel reserve
	extern double en;//Endurance
	extern double sp;//Forward Speed
	extern double al;//Altitude
	extern double uavprojectedarea;// Interim variable to store the value

	//To calculate power later
	extern double ARD_method;//Area of rotor disc
	extern double TS_method;//Tip Speed
	extern double RR_method;//Rotor radius
	extern double SR_method;//Solidity of rotor
	extern double fuel_system;//To record fuel weight
	extern double engine_weight;//To record engine weight

	int fv;
	if(fr==0)
	fv=2;
	else if(fr==1)
	fv=20;
	else //fr==2
	fv=30;

	CString a,b,c;
```

```
c.Format("%d",fv);
b="FUEL RESERVE, LB - ( ";
b+=c;
b+=" MINUTES)";

m_labelfr.SetWindowText(b);

//edit1 value
if((1500*ewt) < (8000*pwt))
{
	a.Format("%ld",int(1500*ewt));
	m_1=a;
	m_1+=" to ";
	a="";
	a.Format("%ld",int(8000*pwt));
	m_1+=a;
}
else
{
	a.Format("%ld",int(8000*pwt));
	m_1=a;
	m_1+=" to ";
	a="";
	a.Format("%ld",int(1500*ewt));
	m_1+=a;
}

//edit2 to be found
m_2.Format("%ld",int(120000 + (333.34*en*pwt)));

//edit3 value
m_3.Format("%g",(pwt+(0.5*uavpower_dbsearch)));
m_4.Format("%lf",uavpower_dbsearch);
m_5 = uavengine_dbsearch;
m_6.Format("%g",(0.45*uavpower_dbsearch*mt/3600));

//Check  fuel values feasibilit
if((0.45*uavpower_dbsearch*mt/3600) < 0.55*(lwt)) //.55 - fuel + .15 - engine
{
	m_6.Format("%g",(0.45*uavpower_dbsearch*mt/3600));
	m_6a.Format("%g",(0.45*uavpower_dbsearch*fv/60));
	m_6a2.Format("%g",((0.45*uavpower_dbsearch*mt/3600)+(0.45*uavpower_dbsearc
	h*fv/60)));
	m_6a3.Format("%g",0.04*((0.45*uavpower_dbsearch*mt/3600)+(0.45*uavpower_d
	bsearch*fv/60)));
	m_6a4.Format("%g",1.04*((0.45*uavpower_dbsearch*mt/3600)+(0.45*uavpower_d
	bsearch*fv/60)));
	fuel_system=1.04*((0.45*uavpower_dbsearch*mt/3600)+(0.45*uavpower_dbsearch
	*fv/60)) ;
}
else
{
	a.Format("%g",lwt);
	MessageBox("Fuel weight = sfc x power x mission-time.  Fuel weight cannot be
	more than 55% of gross weight. Current fuel weight: "+m_6+" lb and gross weight:
	"+a+" lb. You have to decrease the mission-time & endurance duration;  therefore are
	being redirected to input dialog.", "MISMATCHED VALUES");
	EndDialog(IDOK);
	CInputDlg a;
	a.DoModal();
```

```
}

//check if fuel is more than 37.8 litres
m_6a5.Format("%g", (((0.45*uavpower_dbsearch*mt/3600) +
(0.45*uavpower_dbsearch*fv/60))/2.2)*0.716);
if(((((0.45*uavpower_dbsearch*mt/3600)+(0.45*uavpower_dbsearch*fv/60))/2.2)*0.716) >
37.8)
m_6a6= "Safety regulations assert that fuel tank of 37.8 litres or greater capacity must have
internal baffles, or must have external support to resist surging. Besides, fuel tank must be
separated from engine compartment by a firewall";

else
m_6a6= "No need for internal baffles in fuel tank";
m_6a7.Format("%g",
(((0.45*uavpower_dbsearch*mt/3600)+(0.45*uavpower_dbsearch*fv/60))/2.2)*0.795);

if(((((0.45*uavpower_dbsearch*mt/3600)+(0.45*uavpower_dbsearch*fv/60))/2.2)*0.795) >
37.8)
m_6a8= "Safety regulations assert that fuel tank of 37.8 litres or greater capacity must have
internal baffles, or must have external support to resist surging. Besides, fuel tank must be
separated from engine compartment by a firewall";

else
m_6a8= "No need for internal baffles in fuel tank";
m_7.Format("%g",pwt);
m_8=m_6a4;
m_9.Format("%g",0.15*lwt);
engine_weight=0.15*lwt;
m_10a.Format("%g",(lwt-(pwt + 1.04*((0.45*uavpower_dbsearch*mt/3600) +
(0.45*uavpower_dbsearch*fv/60)) + (0.15*lwt))));
m_11.Format("%g",uavprojectedarea);
m_12.Format("%g",0.3);
m_13.Format("%g",4.0);
m_14.Format("%g",0.3*4.0*uavprojectedarea);
m_15= "COUNTER-ROTATING CO-AXIAL ROTOR SYSTEM";
//m_16 var not formed
m_17.Format("%g",(lwt+(0.3*4.0*uavprojectedarea))/4.0);
m_18.Format("%g",sqrt(((lwt+(0.3*4.0*uavprojectedarea))/4.0)/(3.141592653589793238462
6433832795)));//include cmath
m_19.Format("%g", 0.5);
m_19a.Format("%g",sp*0.681818182);
m_20.Format("%g",(sp*0.681818182)/(sqrt(((lwt+(0.3*4.0*uavprojectedarea))/4.0)/3.1416) *
0.5));
m_21.Format("%g",(sp*0.681818182)/0.5);
m_22.Format("%g",((al*0.3048)/1000));
double local= (((al*0.3048)/1000)*6356 / (6356+((al*0.3048)/1000)) );
m_23.Format("%g",local); // Z*E/Z+E 6356 is earth's radius
if(local < 44.3308)
//kg/m3 to lb/ft3 multiply by 0.0624
m_24.Format("%g",0.0624*(pow(((44.3308-local)/42.2665),4.2556)));
else
{
m_24="Out-of-range altitude";
}
m_25.Format("%g",fabs(pow(((44.3308-local)/4.94654),5.2556)));
m_26.Format("%ld",4);
m_27.Format("%g",0.23);
double solidity=fabs(((0.3*4.0*uavprojectedarea) + lwt)/(0.23*(pow((44.3308-
local)/42.2665,4.2556)) * ((lwt+(0.3*4.0*uavprojectedarea))/4.0) *
pow(((sp*0.681818182)/0.5),2)));
```

```
	m_28.Format("%g",solidity);
	UpdateData(FALSE);


	//Filling the values
	ARD_method=(lwt+(0.3*4.0*uavprojectedarea))/4.0;
	TS_method=(sp*0.681818182)/0.5;
	RR_method=sqrt(((lwt+(0.3*4.0*uavprojectedarea))/4.0)/(3.14159265358979323846264338327795));
	SR_method=solidity;
	return TRUE;  // return TRUE unless you set the focus to a control
		// EXCEPTION: OCX Property Pages should return FALSE
}

void C7_1Dlg::On7b()
{
	EndDialog(IDCANCEL);
	CDatabaseSearchDLG a;
	a.DoModal();
}

void C7_1Dlg::On7h1()
{
	MessageBox("Empty weight cost is a commonly used metric in the aviation industry because it tends to remain constant across a variety of aircraft types. That number today is roughly $1500 per pound. Also, historically it is presumed that it costs around $8000 per pound of payload capacity. The two different values are ranged to form max and min limit.","HOW?");
}

void C7_1Dlg::On7h2()
{
	MessageBox("A link between performance and cost in terms of dollars per pound-hour has been framed. The in-air cost of UAV is calculated as = US$ (120K + 334*(Payload*Endurance))","HOW?");
}

void C7_1Dlg::On7h3()
{
	MessageBox("Useful load (in pounds) = Payload+(0.5*Engine-Power(shp))","HOW?");
}

void C7_1Dlg::On7h4()
{
	MessageBox("Fuel=sfc(lb/hp/hr) * engine-power(hp) * Mission-time(hour); where sfc = 0.5 for piston engines and 0.4 for turbine engines","HOW?");
}

void C7_1Dlg::On7h5()
{
	MessageBox("Reserve-fuel=sfc(lb/hp/hr) * engine-power(hp) * Reserve-time(hour); where sfc = 0.5 for piston engines and 0.4 for turbine engines","HOW?");
}

void C7_1Dlg::On7h6()
{
	MessageBox("A guide to suitable petrol/oil ratios for engines is hard to  formulate but as a world-wide standard, a ratio of 50:1 is considered healthy. Usually, highly developed engines with maximum engine speeds in excess of 7500rpm have ratio of 20:1 and modified engines with maximum engine speed less than 7500 rpm have ratio closed at 35:1. Considering these propositions, the ratio of 24:1 seems fair.","HOW?");
```

```
}

void C7_1Dlg::On7h7()
{
	MessageBox("Initial guess at engine weight is from following equation: Engine-weight ~= 0.15*Gross-weight","HOW?");
}

void C7_1Dlg::On7h8()
{
	MessageBox("The projected area is evaluated from the plan view of the RWUAV. The plan view is divided into sectors and the areas of these sectors are evaluated geometrically. The aggregate of these areas is the projected area of the RWUAV, which will be presented in the profile diagram of the aircraft.","HOW?");
}

void C7_1Dlg::On7h9()
{
	MessageBox("A first estimate of downwash is obtained by assuming a drag coefficient of 0.3 for all RWUAV components in the remote wake.","HOW?");
}

void C7_1Dlg::On7h10()
{
	MessageBox("Disc Loading ranges from 4.0 to 12.0 (lb/ft*ft). As the constraint of the design process is to offer maximum hovering capabilities, disc loading is chosen to be at the lower end of the scale. Other advantages include low induced velocity and low autorotative rate of descent.","HOW?");
}

void C7_1Dlg::On7h11()
{
	MessageBox("Vertical Drag = Drag-Coefficient x Disk-Loading x Projected-Area","HOW?");
}

void C7_1Dlg::On7h12()
{
	MessageBox("The design chosen for RWUAV has a counter-coaxial rotor system. Both rotors will be of same shape and size, and will be rotating at same speed but in opposite direction. The advantages offered by this system are numerous. First, it is compact and saves space. Second, torque reaction is cancelled hence reducing the complexity in design calculations. Third, it allows for greater lifting capability as all the power is used by main rotor system. Fourth, it allows RWUAV to have a smaller fuselage and hence a narrower profile. Finally, lack of airframe structure necessary for the tail rotor system allows for greater payload capability with a low noise signature.","WHY?");
}

void C7_1Dlg::On7h13()
{
	MessageBox("DISC AREA = (GROSS WEIGHT + VERTICAL DRAG)/ DISC LOADING","HOW?");
}

void C7_1Dlg::On7h14()
{
	MessageBox("DISC AREA = Pi x R x R","HOW?");
}

void C7_1Dlg::On7h15()
{
```

```
	MessageBox("TIP SPEED of conventional helicopters range from 500 to 800 ft/sec and the
	'tip speed ratio' at max forward speed does not exceed 0.5","HOW?");
}

void C7_1Dlg::On7h16()
{
	MessageBox("ROTATIONAL SPEED = FORWARD SPEED/(TIP RATIO * ROTOR
	RADIUS)","HOW?");
}

void C7_1Dlg::On7h17()
{
	MessageBox("TIP SPEED = ROTATIONAL SPEED * ROTOR RADIUS","HOW?");
}

void C7_1Dlg::On7h19()
{
	MessageBox(" Z=(E*H)/(E-H), where E = 6356  km, the radius of the earth (for 1976 ISA), H
	= geopotential altitude in km, & Z = geometric altitude, km","HOW?");
}

void C7_1Dlg::On7h20()
{
	MessageBox("D = POW((44.3308-H)/42.2665, 4.2556) * 0.0624, where   H = geopotential
	altitude in km and D = air density, lb/ft3","HOW?");
}

void C7_1Dlg::On7h21()
{
	MessageBox("P = POW((44.3308-H)/4.94654, 5.2556), where   H = geopotential altitude in
	km and P = actual air pressure, Pascals","HOW?");
}

void C7_1Dlg::On7h22()
{
	MessageBox("The radius-to-chord (R/C) ratio governs the number of blades.  R/C ratios of
	less than eight results in high tip losses, affecting the performance.  R/C ratios of above 24
	provide structural flexibility being easy to bend and twist in flight.  At the preliminary design
	stage an aspect ratio of twelve is considered to evaluate the number of blades.  Alternatively,
	as number of blades is not a major design decision; four blades are generally accepted as
	conventional, for further design analysis.","HOW?");
}

void C7_1Dlg::On7h23()
{
	MessageBox("The number of blades is evaluated from the 'thrust coefficient'-to-'solidity ratio'.
	The ratio ranges from 0.17 at sea level and varies with altitude to a maximum of 0.23. Since,
	we are performing analysis at max altitude, therefore max value is assumed.","HOW?");
}

void C7_1Dlg::On7h24()
{
	MessageBox("SOLIDITY=(GROSS-WEIGHT + VERTICAL DRAG)/0.23 * AIR-DENSITY
	* ROTOR-DISC-AREA * TIP-SPEED * TIP-SPEED","HOW?");
}

void C7_1Dlg::On7H6a()
{
	MessageBox("Each fuel system must be constructed and arranged to ensure a flow of fuel at a
	rate and pressure established for proper engine functioning under any likely operating
```

```
        condition, including the maneuvers for which certification is requested. The component
        recommendation is as follow:  1.  Each fuel tank must be able to withstand, without failure,
        the vibration, inertia, fluid, and structural loads to which it may be subjected in operation. 2.
        The maximum exposed surface temperature of any component in the fuel tank must be less, by
        a safe margin, than the lowest expected auto-ignition temperature of the fuel or fuel vapor in
        the tank.  3.  At least one-half inch of clear airspace must be provided between the tank and
        the firewall. 4. Each fuel tank must be supported so that loads resulting from the weight of fuel
        are not concentrated on unsupported tank skin. 5. Each compartment containing a fuel tank
        must be ventilated and drained overboard. 6. Each fuel tank, if permanently installed, must
        have a drainable sump which is effective in all normal ground and flight attitudes and with a
        capacity of 0.10% of the tank capacity, or 120 ml, whichever is the greater. 7. Each fuel tank
        must be vented from the top of the tank. 8. There must be a fuel filter between the fuel tank
        outlet and the fuel pump inlet. 9. Each fuel line must be installed and supported to prevent
        excessive vibration and to withstand loads due to fuel pressure and accelerated flight
        conditions. 10. Each fuel drain must: (a) Be easily accessible; (b) Discharge clear of the
        aircraft; and (c) Have an automatic means for positive locking in the closed position.","FUEL
        SYSTEM DESIGN");
}

void C7_1Dlg::On7n()
{
        EndDialog(IDCANCEL);
        C7_2Dlg a;
        a.DoModal();

}
```

J.6 ‘Rotor system-II’ class

The ‘Rotor system-II’ class is programmed to involve the selection of blade airfoil as follows:

```
// 7_2Dlg.cpp : implementation file
// Declaration of header files
#include "stdafx.h"
#include "VTUAV_Expert_System.h"
#include "7_2Dlg.h"
#include "7_1Dlg.h"
#include "7_3Dlg.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

// Declaration & initialisation of class variables
C7_2Dlg::C7_2Dlg(CWnd* pParent /*=NULL*/)
        : CDialog(C7_2Dlg::IDD, pParent)
{
        //{{AFX_DATA_INIT(C7_2Dlg)
        m_edit1 = _T("");
        m_radio1 = 0;
        m_radio3 = 1;
        m_edit2 = _T("");
        //}}AFX_DATA_INIT
}
```

// Limiting variables values

```
void C7_2Dlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(C7_2Dlg)
	DDX_Control(pDX, IDC_EDIT2, m_edit2c);
	DDX_Control(pDX, IDC_EDIT1, m_editc);
	DDX_Control(pDX, IDC_SLIDER3, m_slider3);
	DDX_Control(pDX, IDC_SLIDER2, m_slider2);
	DDX_Control(pDX, IDC_SLIDER1, m_slider1);
	DDX_Text(pDX, IDC_EDIT1, m_edit1);
	DDX_Radio(pDX, IDC_RADIO1, m_radio1);
	DDX_Control(pDX, IDC_PICTURECLIP00, m_pic);
	DDX_Control(pDX, IDC_PICTURECLIP04, m_4);
	DDX_Control(pDX, IDC_PICTURECLIP06, m_6);
	DDX_Control(pDX, IDC_PICTURECLIP08, m_8);
	DDX_Control(pDX, IDC_PICTURECLIP20, m_20);
	DDX_Control(pDX, IDC_PICTURECLIP22, m_22);
	DDX_Control(pDX, IDC_PICTURECLIP24, m_24);
	DDX_Control(pDX, IDC_PICTURECLIP26, m_26);
	DDX_Control(pDX, IDC_PICTURECLIP28, m_28);
	DDX_Control(pDX, IDC_PICTURECLIP40, m_40);
	DDX_Control(pDX, IDC_PICTURECLIP42, m_42);
	DDX_Control(pDX, IDC_PICTURECLIP44, m_44);
	DDX_Control(pDX, IDC_PICTURECLIP46, m_46);
	DDX_Control(pDX, IDC_PICTURECLIP48, m_48);
	DDX_Control(pDX, IDC_PICTURECLIP60, m_60);
	DDX_Control(pDX, IDC_PICTURECLIP62, m_62);
	DDX_Control(pDX, IDC_PICTURECLIP64, m_64);
	DDX_Control(pDX, IDC_PICTURECLIP66, m_66);
	DDX_Control(pDX, IDC_PICTURECLIP68, m_68);
	DDX_Control(pDX, IDC_PICTURECLIP80, m_80);
	DDX_Control(pDX, IDC_PICTURECLIP82, m_82);
	DDX_Control(pDX, IDC_PICTURECLIP84, m_84);
	DDX_Control(pDX, IDC_PICTURECLIP86, m_86);
	DDX_Control(pDX, IDC_PICTURECLIP8812, m_88);
	DDX_Control(pDX, IDC_PICTURECLIP02, m_2);
	DDX_Radio(pDX, IDC_RADIO3, m_radio3);
	DDX_Text(pDX, IDC_EDIT2, m_edit2);
	//}}AFX_DATA_MAP
}
```

// Defining user-actions vis-a-vis system messages

```
BEGIN_MESSAGE_MAP(C7_2Dlg, CDialog)
	//{{AFX_MSG_MAP(C7_2Dlg)
	ON_NOTIFY(NM_RELEASEDCAPTURE, IDC_SLIDER1, OnReleasedcaptureSlider1)
	ON_NOTIFY(NM_RELEASEDCAPTURE, IDC_SLIDER2, OnReleasedcaptureSlider2)
	ON_NOTIFY(NM_RELEASEDCAPTURE, IDC_SLIDER3, OnReleasedcaptureSlider3)
	ON_BN_CLICKED(IDC_RADIO1, OnRadio1)
	ON_BN_CLICKED(IDC_RADIO2, OnRadio2)
	ON_BN_CLICKED(ID7M, On7m)
	ON_BN_CLICKED(ID7M2, On7m2)
	ON_BN_CLICKED(IDC_RADIO3, OnRadio3)
	ON_BN_CLICKED(IDC_RADIO4, OnRadio4)
	ON_BN_CLICKED(ID7_2B, On7_2b)
	ON_BN_CLICKED(ID7_2N, On2n)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

// Class method – Initialisation of dialog

```
BOOL C7_2Dlg::OnInitDialog()
{
	CDialog::OnInitDialog();
	// TODO: Add extra initialization here
	m_slider1.EnableWindow(TRUE);
	m_slider1.SetRange(0,8,FALSE);
	m_slider1.SetPos(0);
	m_slider1.SetTicFreq(2);
	m_slider2.EnableWindow(TRUE);
	m_slider2.SetRange(0,8,FALSE);
	m_slider2.SetPos(0);
	m_slider2.SetTicFreq(2);
	m_slider3.EnableWindow(TRUE);
	m_slider3.SetRange(0,98,FALSE);
	m_slider3.SetPos(12);
	m_slider3.SetTicFreq(2);
	UpdateData(TRUE);

	if(m_radio1==0)
	m_edit1= "NACA 0012 seems to be a good choice with camber value = 0, maximum camber
	position = 0 and thickness value = 12.";
	else
	m_edit1= "Assymmetrical NACA combinations can be explored/selected by moving the
	sliders. However the third/thickness slider is disabled as thickness value of 12 is highly
	recommended as it has been thoroughly investigated and is extensively used." ;

	if(m_radio3==0)
	m_edit2="SQUARE TIP - easy to design, easy to manufacture and is a much cheaper option.";
	else
	m_edit2="SWEPTBACK TIPS - produce dynamic twist that improves the unbalanced air
	loading distribution in forward flight; uploaded tip on retreating side increases the nose down
	twist on that side that is beneficial for hover performances; finally the swept-back blade tip
	configuration tends to offer VTUAV speed stability.";

	//Show pic
	m_pic.EnableWindow(TRUE);
	m_pic.ShowWindow(SW_SHOW);

	//Disable sliders
	m_slider1.EnableWindow(FALSE);
	m_slider2.EnableWindow(FALSE);
	m_slider3.EnableWindow(FALSE);

	UpdateData(FALSE);
	return TRUE;  // return TRUE unless you set the focus to a control
		// EXCEPTION: OCX Property Pages should return FALSE
}


void C7_2Dlg::OnReleasedcaptureSlider1(NMHDR* pNMHDR, LRESULT* pResult)
{
	*pResult = 0;
	m_pic.ShowWindow(SW_SHOW);

	if(m_slider1.GetPos()==1)
	m_slider1.SetPos(m_slider1.GetPos()+1);
	if(m_slider1.GetPos()==3)
	m_slider1.SetPos(m_slider1.GetPos()+1);
	if(m_slider1.GetPos()==5)
	m_slider1.SetPos(m_slider1.GetPos()+1);
```

```
if(m_slider1.GetPos()==7)
m_slider1.SetPos(m_slider1.GetPos()+1);

if(m_slider1.GetPos() == 0 && m_slider2.GetPos() == 2)
{
	m_pic.ShowWindow(SW_HIDE);
	m_2.ShowWindow(SW_SHOW);
	m_4.ShowWindow(SW_HIDE);
	m_6.ShowWindow(SW_HIDE);
	m_8.ShowWindow(SW_HIDE);
	m_20.ShowWindow(SW_HIDE);
	m_22.ShowWindow(SW_HIDE);
	m_24.ShowWindow(SW_HIDE);
	m_26.ShowWindow(SW_HIDE);
	m_28.ShowWindow(SW_HIDE);
	m_40.ShowWindow(SW_HIDE);
	m_42.ShowWindow(SW_HIDE);
	m_44.ShowWindow(SW_HIDE);
	m_46.ShowWindow(SW_HIDE);
	m_48.ShowWindow(SW_HIDE);
	m_60.ShowWindow(SW_HIDE);
	m_62.ShowWindow(SW_HIDE);
	m_64.ShowWindow(SW_HIDE);
	m_66.ShowWindow(SW_HIDE);
	m_68.ShowWindow(SW_HIDE);
	m_80.ShowWindow(SW_HIDE);
	m_82.ShowWindow(SW_HIDE);
	m_84.ShowWindow(SW_HIDE);
	m_86.ShowWindow(SW_HIDE);
	m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 0 && m_slider2.GetPos() == 4)
{
	m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
	m_4.ShowWindow(SW_SHOW); m_6.ShowWindow(SW_HIDE);
	m_8.ShowWindow(SW_HIDE);
	m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
	m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
	m_28.ShowWindow(SW_HIDE);
	m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
	m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
	m_48.ShowWindow(SW_HIDE);
	m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
	m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
	m_68.ShowWindow(SW_HIDE);
	m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
	m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
	m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 0 && m_slider2.GetPos() == 6)
{
	m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
	m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_SHOW);
	m_8.ShowWindow(SW_HIDE);
	m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
	m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
	m_28.ShowWindow(SW_HIDE);
```

```
		m_40.ShowWindow(SW_HIDE);	m_42.ShowWindow(SW_HIDE);
		m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
		m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
		m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
		m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_HIDE);
		m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
		m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 0 && m_slider2.GetPos() == 8)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_SHOW);
		m_20.ShowWindow(SW_HIDE);	m_22.ShowWindow(SW_HIDE);
		m_24.ShowWindow(SW_HIDE);	m_26.ShowWindow(SW_HIDE);
		m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);	m_42.ShowWindow(SW_HIDE);
		m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
		m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
		m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
		m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_HIDE);
		m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
		m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 2 && m_slider2.GetPos() == 0)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_SHOW); m_22.ShowWindow(SW_HIDE);
		m_24.ShowWindow(SW_HIDE);	m_26.ShowWindow(SW_HIDE);
		m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);	m_42.ShowWindow(SW_HIDE);
		m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
		m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
		m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
		m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_HIDE);
		m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
		m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 2 && m_slider2.GetPos() == 2)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);	m_22.ShowWindow(SW_SHOW);
		m_24.ShowWindow(SW_HIDE);	m_26.ShowWindow(SW_HIDE);
		m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);	m_42.ShowWindow(SW_HIDE);
		m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
		m_48.ShowWindow(SW_HIDE);
```

```
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
		m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
		m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_HIDE);
		m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
		m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 2 && m_slider2.GetPos() == 4)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);	m_22.ShowWindow(SW_HIDE);
		m_24.ShowWindow(SW_SHOW); m_26.ShowWindow(SW_HIDE);
		m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);	m_42.ShowWindow(SW_HIDE);
		m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
		m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
		m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
		m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_HIDE);
		m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
		m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 2 && m_slider2.GetPos() == 6)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);	m_22.ShowWindow(SW_HIDE);
		m_24.ShowWindow(SW_HIDE);	m_26.ShowWindow(SW_SHOW);
		m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);	m_42.ShowWindow(SW_HIDE);
		m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
		m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
		m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
		m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_HIDE);
		m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
		m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 2 && m_slider2.GetPos() == 8)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);	m_22.ShowWindow(SW_HIDE);
		m_24.ShowWindow(SW_HIDE);	m_26.ShowWindow(SW_HIDE);
		m_28.ShowWindow(SW_SHOW);
		m_40.ShowWindow(SW_HIDE);	m_42.ShowWindow(SW_HIDE);
		m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
		m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
		m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
		m_68.ShowWindow(SW_HIDE);
```

```
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_HIDE);
		m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
		m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 4 && m_slider2.GetPos() == 0)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);	m_22.ShowWindow(SW_HIDE);
		m_24.ShowWindow(SW_HIDE);	m_26.ShowWindow(SW_HIDE);
		m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_SHOW); m_42.ShowWindow(SW_HIDE);
		m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
		m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
		m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
		m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_HIDE);
		m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
		m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 4 && m_slider2.GetPos() == 2)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);	m_22.ShowWindow(SW_HIDE);
		m_24.ShowWindow(SW_HIDE);	m_26.ShowWindow(SW_HIDE);
		m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);	m_42.ShowWindow(SW_SHOW);
		m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
		m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
		m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
		m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_HIDE);
		m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
		m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 4 && m_slider2.GetPos() == 4)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);	m_22.ShowWindow(SW_HIDE);
		m_24.ShowWindow(SW_HIDE);	m_26.ShowWindow(SW_HIDE);
		m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);	m_42.ShowWindow(SW_HIDE);
		m_44.ShowWindow(SW_SHOW);		m_46.ShowWindow(SW_HIDE);
		m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
		m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
		m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_HIDE);
		m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
		m_88.ShowWindow(SW_HIDE);
```

```
}

if(m_slider1.GetPos() == 4 && m_slider2.GetPos() == 6)
{
	m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
	m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
	m_8.ShowWindow(SW_HIDE);
	m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
	m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
	m_28.ShowWindow(SW_HIDE);
	m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
	m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_SHOW);
	m_48.ShowWindow(SW_HIDE);
	m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
	m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
	m_68.ShowWindow(SW_HIDE);
	m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
	m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
	m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 4 && m_slider2.GetPos() == 8)
{
	m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
	m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
	m_8.ShowWindow(SW_HIDE);
	m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
	m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
	m_28.ShowWindow(SW_HIDE);
	m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
	m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
	m_48.ShowWindow(SW_SHOW);
	m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
	m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
	m_68.ShowWindow(SW_HIDE);
	m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
	m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
	m_88.ShowWindow(SW_HIDE);
}


if(m_slider1.GetPos() == 6 && m_slider2.GetPos() == 0)
{
	m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
	m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
	m_8.ShowWindow(SW_HIDE);
	m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
	m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
	m_28.ShowWindow(SW_HIDE);
	m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
	m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
	m_48.ShowWindow(SW_HIDE);
	m_60.ShowWindow(SW_SHOW); m_62.ShowWindow(SW_HIDE);
	m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
	m_68.ShowWindow(SW_HIDE);
	m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
	m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
	m_88.ShowWindow(SW_HIDE);
}
```

```
if(m_slider1.GetPos() == 6 && m_slider2.GetPos() == 2)
{
	m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
	m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
	m_8.ShowWindow(SW_HIDE);
	m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
	m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
	m_28.ShowWindow(SW_HIDE);
	m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
	m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
	m_48.ShowWindow(SW_HIDE);
	m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_SHOW);
	m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
	m_68.ShowWindow(SW_HIDE);
	m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
	m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
	m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 6 && m_slider2.GetPos() == 4)
{
	m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
	m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
	m_8.ShowWindow(SW_HIDE);
	m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
	m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
	m_28.ShowWindow(SW_HIDE);
	m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
	m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
	m_48.ShowWindow(SW_HIDE);
	m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
	m_64.ShowWindow(SW_SHOW); m_66.ShowWindow(SW_HIDE);
	m_68.ShowWindow(SW_HIDE);
	m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
	m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
	m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 6 && m_slider2.GetPos() == 6)
{
	m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
	m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
	m_8.ShowWindow(SW_HIDE);
	m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
	m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
	m_28.ShowWindow(SW_HIDE);
	m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
	m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
	m_48.ShowWindow(SW_HIDE);
	m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
	m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_SHOW);
	m_68.ShowWindow(SW_HIDE);
	m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
	m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
	m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 6 && m_slider2.GetPos() == 8)
{
```

```
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
		m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
		m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
		m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
		m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
		m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
		m_68.ShowWindow(SW_SHOW);
		m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
		m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
		m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 8 && m_slider2.GetPos() == 0)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
		m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
		m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
		m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
		m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
		m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
		m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_SHOW); m_82.ShowWindow(SW_HIDE);
		m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
		m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 8 && m_slider2.GetPos() == 2)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
		m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
		m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
		m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
		m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
		m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
		m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_SHOW);
		m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
		m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 8 && m_slider2.GetPos() == 4)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
```

```
		m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
		m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
		m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
		m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
		m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
		m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
		m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
		m_84.ShowWindow(SW_SHOW); m_86.ShowWindow(SW_HIDE);
		m_88.ShowWindow(SW_HIDE);
	}

	if(m_slider1.GetPos() == 8 && m_slider2.GetPos() == 6)
	{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
		m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
		m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
		m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
		m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
		m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
		m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
		m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_SHOW);
		m_88.ShowWindow(SW_HIDE);
	}

	if(m_slider1.GetPos() == 8 && m_slider2.GetPos() == 8)
	{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
		m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
		m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
		m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
		m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
		m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
		m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
		m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
		m_88.ShowWindow(SW_SHOW);
	}

}


void C7_2Dlg::OnReleasedcaptureSlider2(NMHDR* pNMHDR, LRESULT* pResult)
{
	*pResult = 0;
	m_pic.ShowWindow(SW_SHOW);
```

```
if(m_slider2.GetPos()==1)
m_slider2.SetPos(m_slider2.GetPos()+1);
if(m_slider2.GetPos()==3)
m_slider2.SetPos(m_slider2.GetPos()+1);
if(m_slider2.GetPos()==5)
m_slider2.SetPos(m_slider2.GetPos()+1);
if(m_slider2.GetPos()==7)
m_slider2.SetPos(m_slider2.GetPos()+1);

if(m_slider1.GetPos() == 0 && m_slider2.GetPos() == 2)
{
	m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_SHOW);
	m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
	m_8.ShowWindow(SW_HIDE);
	m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
	m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
	m_28.ShowWindow(SW_HIDE);
	m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
	m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
	m_48.ShowWindow(SW_HIDE);
	m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
	m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
	m_68.ShowWindow(SW_HIDE);
	m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
	m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
	m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 0 && m_slider2.GetPos() == 4)
{
	m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
	m_4.ShowWindow(SW_SHOW); m_6.ShowWindow(SW_HIDE);
	m_8.ShowWindow(SW_HIDE);
	m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
	m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
	m_28.ShowWindow(SW_HIDE);
	m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
	m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
	m_48.ShowWindow(SW_HIDE);
	m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
	m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
	m_68.ShowWindow(SW_HIDE);
	m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
	m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
	m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 0 && m_slider2.GetPos() == 6)
{
	m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
	m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_SHOW);
	m_8.ShowWindow(SW_HIDE);
	m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
	m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
	m_28.ShowWindow(SW_HIDE);
	m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
	m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
	m_48.ShowWindow(SW_HIDE);
```

```
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
		m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
		m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_HIDE);
		m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
		m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 0 && m_slider2.GetPos() == 8)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_SHOW);
		m_20.ShowWindow(SW_HIDE);	m_22.ShowWindow(SW_HIDE);
		m_24.ShowWindow(SW_HIDE);	m_26.ShowWindow(SW_HIDE);
		m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);	m_42.ShowWindow(SW_HIDE);
		m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
		m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
		m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
		m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_HIDE);
		m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
		m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 2 && m_slider2.GetPos() == 0)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_SHOW); m_22.ShowWindow(SW_HIDE);
		m_24.ShowWindow(SW_HIDE);	m_26.ShowWindow(SW_HIDE);
		m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);	m_42.ShowWindow(SW_HIDE);
		m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
		m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
		m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
		m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_HIDE);
		m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
		m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 2 && m_slider2.GetPos() == 2)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE); m_22.ShowWindow(SW_SHOW);
		m_24.ShowWindow(SW_HIDE); m_26.ShowWindow(SW_HIDE);
		m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);	m_42.ShowWindow(SW_HIDE);
		m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
		m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
		m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
		m_68.ShowWindow(SW_HIDE);
```

```
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_HIDE);
		m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
		m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 2 && m_slider2.GetPos() == 4)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);	m_22.ShowWindow(SW_HIDE);
		m_24.ShowWindow(SW_SHOW); m_26.ShowWindow(SW_HIDE);
		m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);	m_42.ShowWindow(SW_HIDE);
		m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
		m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
		m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
		m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_HIDE);
		m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
		m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 2 && m_slider2.GetPos() == 6)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);	m_22.ShowWindow(SW_HIDE);
		m_24.ShowWindow(SW_HIDE);	m_26.ShowWindow(SW_SHOW);
		m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);	m_42.ShowWindow(SW_HIDE);
		m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
		m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
		m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
		m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_HIDE);
		m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
		m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 2 && m_slider2.GetPos() == 8)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);	m_22.ShowWindow(SW_HIDE);
		m_24.ShowWindow(SW_HIDE);	m_26.ShowWindow(SW_HIDE);
		m_28.ShowWindow(SW_SHOW);
		m_40.ShowWindow(SW_HIDE);	m_42.ShowWindow(SW_HIDE);
		m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
		m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
		m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
		m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_HIDE);
		m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
		m_88.ShowWindow(SW_HIDE);
```

```
}

if(m_slider1.GetPos() == 4 && m_slider2.GetPos() == 0)
{
	m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
	m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
	m_8.ShowWindow(SW_HIDE);
	m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
	m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
	m_28.ShowWindow(SW_HIDE);
	m_40.ShowWindow(SW_SHOW); m_42.ShowWindow(SW_HIDE);
	m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
	m_48.ShowWindow(SW_HIDE);
	m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
	m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
	m_68.ShowWindow(SW_HIDE);
	m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
	m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
	m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 4 && m_slider2.GetPos() == 2)
{
	m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
	m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
	m_8.ShowWindow(SW_HIDE);
	m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
	m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
	m_28.ShowWindow(SW_HIDE);
	m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_SHOW);
	m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
	m_48.ShowWindow(SW_HIDE);
	m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
	m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
	m_68.ShowWindow(SW_HIDE);
	m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
	m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
	m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 4 && m_slider2.GetPos() == 4)
{
	m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
	m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
	m_8.ShowWindow(SW_HIDE);
	m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
	m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
	m_28.ShowWindow(SW_HIDE);
	m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
	m_44.ShowWindow(SW_SHOW);          m_46.ShowWindow(SW_HIDE);
	m_48.ShowWindow(SW_HIDE);
	m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
	m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
	m_68.ShowWindow(SW_HIDE);
	m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
	m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
	m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 4 && m_slider2.GetPos() == 6)
```

```
{
	m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
	m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
	m_8.ShowWindow(SW_HIDE);
	m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
	m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
	m_28.ShowWindow(SW_HIDE);
	m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
	m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_SHOW);
	m_48.ShowWindow(SW_HIDE);
	m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
	m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
	m_68.ShowWindow(SW_HIDE);
	m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
	m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
	m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 4 && m_slider2.GetPos() == 8)
{
	m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
	m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
	m_8.ShowWindow(SW_HIDE);
	m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
	m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
	m_28.ShowWindow(SW_HIDE);
	m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
	m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
	m_48.ShowWindow(SW_SHOW);
	m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
	m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
	m_68.ShowWindow(SW_HIDE);
	m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
	m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
	m_88.ShowWindow(SW_HIDE);
}


if(m_slider1.GetPos() == 6 && m_slider2.GetPos() == 0)
{
	m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
	m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
	m_8.ShowWindow(SW_HIDE);
	m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
	m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
	m_28.ShowWindow(SW_HIDE);
	m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
	m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
	m_48.ShowWindow(SW_HIDE);
	m_60.ShowWindow(SW_SHOW); m_62.ShowWindow(SW_HIDE);
	m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
	m_68.ShowWindow(SW_HIDE);
	m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
	m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
	m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 6 && m_slider2.GetPos() == 2)
{
```

```
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
		m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
		m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
		m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
		m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_SHOW);
		m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
		m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
		m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
		m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 6 && m_slider2.GetPos() == 4)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
		m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
		m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
		m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
		m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
		m_64.ShowWindow(SW_SHOW); m_66.ShowWindow(SW_HIDE);
		m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
		m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
		m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 6 && m_slider2.GetPos() == 6)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
		m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
		m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
		m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
		m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
		m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_SHOW);
		m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
		m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
		m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 6 && m_slider2.GetPos() == 8)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
```

```
		m_20.ShowWindow(SW_HIDE);	m_22.ShowWindow(SW_HIDE);
		m_24.ShowWindow(SW_HIDE);	m_26.ShowWindow(SW_HIDE);
		m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);	m_42.ShowWindow(SW_HIDE);
		m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
		m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
		m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
		m_68.ShowWindow(SW_SHOW);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_HIDE);
		m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
		m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 8 && m_slider2.GetPos() == 0)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);	m_22.ShowWindow(SW_HIDE);
		m_24.ShowWindow(SW_HIDE);	m_26.ShowWindow(SW_HIDE);
		m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);	m_42.ShowWindow(SW_HIDE);
		m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
		m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
		m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
		m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_SHOW); m_82.ShowWindow(SW_HIDE);
		m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
		m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 8 && m_slider2.GetPos() == 2)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);	m_22.ShowWindow(SW_HIDE);
		m_24.ShowWindow(SW_HIDE);	m_26.ShowWindow(SW_HIDE);
		m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);	m_42.ShowWindow(SW_HIDE);
		m_44.ShowWindow(SW_HIDE);	m_46.ShowWindow(SW_HIDE);
		m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);	m_62.ShowWindow(SW_HIDE);
		m_64.ShowWindow(SW_HIDE);	m_66.ShowWindow(SW_HIDE);
		m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);	m_82.ShowWindow(SW_SHOW);
		m_84.ShowWindow(SW_HIDE);	m_86.ShowWindow(SW_HIDE);
		m_88.ShowWindow(SW_HIDE);
}

if(m_slider1.GetPos() == 8 && m_slider2.GetPos() == 4)
{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);	m_22.ShowWindow(SW_HIDE);
		m_24.ShowWindow(SW_HIDE);	m_26.ShowWindow(SW_HIDE);
		m_28.ShowWindow(SW_HIDE);
```

```
		m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
		m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
		m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
		m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
		m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
		m_84.ShowWindow(SW_SHOW); m_86.ShowWindow(SW_HIDE);
		m_88.ShowWindow(SW_HIDE);
	}

	if(m_slider1.GetPos() == 8 && m_slider2.GetPos() == 6)
	{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
		m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
		m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
		m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
		m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
		m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
		m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
		m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_SHOW);
		m_88.ShowWindow(SW_HIDE);
	}

	if(m_slider1.GetPos() == 8 && m_slider2.GetPos() == 8)
	{
		m_pic.ShowWindow(SW_HIDE); m_2.ShowWindow(SW_HIDE);
		m_4.ShowWindow(SW_HIDE); m_6.ShowWindow(SW_HIDE);
		m_8.ShowWindow(SW_HIDE);
		m_20.ShowWindow(SW_HIDE);   m_22.ShowWindow(SW_HIDE);
		m_24.ShowWindow(SW_HIDE);   m_26.ShowWindow(SW_HIDE);
		m_28.ShowWindow(SW_HIDE);
		m_40.ShowWindow(SW_HIDE);   m_42.ShowWindow(SW_HIDE);
		m_44.ShowWindow(SW_HIDE);   m_46.ShowWindow(SW_HIDE);
		m_48.ShowWindow(SW_HIDE);
		m_60.ShowWindow(SW_HIDE);   m_62.ShowWindow(SW_HIDE);
		m_64.ShowWindow(SW_HIDE);   m_66.ShowWindow(SW_HIDE);
		m_68.ShowWindow(SW_HIDE);
		m_80.ShowWindow(SW_HIDE);   m_82.ShowWindow(SW_HIDE);
		m_84.ShowWindow(SW_HIDE);   m_86.ShowWindow(SW_HIDE);
		m_88.ShowWindow(SW_SHOW);
	}

}



void C7_2Dlg::OnReleasedcaptureSlider3(NMHDR* pNMHDR, LRESULT* pResult)
{
	*pResult = 0;
	CString a;
	a.Format("%d",m_slider3.GetPos());
	MessageBox(a);
}
```

```
void C7_2Dlg::OnRadio1()
{
	m_pic.ShowWindow(SW_SHOW);
	m_slider1.SetPos(0);
	m_slider2.SetPos(0);
	m_slider1.EnableWindow(FALSE);
	m_slider2.EnableWindow(FALSE);
	m_editc.SetWindowText("NACA 0012 seems to be a good choice with camber value = 0,
	maximum camber position = 0 and thickness value = 12.");
}

void C7_2Dlg::OnRadio2()
{
	m_slider1.EnableWindow(TRUE);
	m_slider2.EnableWindow(TRUE);
	m_editc.SetWindowText("Asymmetrical NACA combinations can be explored/selected by
	moving the sliders. However the third/thickness slider is disabled as thickness value of 12 is
	highly recommended as it has been thoroughly investigated and is extensively used.");
}

void C7_2Dlg::On7m()
{
	MessageBox("Airfoil sections are of two basic types, symmetrical and nonsymmetrical. (A)
	Symmetrical airfoils have identical upper and lower surfaces. They are suited to rotary-wing
	applications because they have almost no center of pressure travel. Travel remains relatively
	constant under varying angles of attack, affording the best lift-drag ratios for the full range of
	velocities from rotor blade root to tip. However, the symmetrical airfoil produces less lift than
	a nonsymmetrical airfoil and also has relatively undesirable stall characteristics. The rotorcraft
	blade must adapt to a wide range of airspeeds and angles of attack during each revolution of
	the rotor. The symmetrical airfoil delivers acceptable performance under those alternating
	conditions. Other benefits are lower cost and ease of construction as compared to the
	nonsymmetrical airfoil. (B) Nonsymmetrical (cambered) airfoils may have a wide variety of
	upper and lower surface designs. Advantages of the nonsymmetrical airfoil are increased lift-
	drag ratios and more desirable stall characteristics.","AIRFOIL SECTION TYPES");
}

void C7_2Dlg::On7m2()
{
	MessageBox("Advanced airfoil design methodology including Navier-Stokes CFD codes to
	design multi-element airfoils with leading edge slots. The unique rotorcraft challenge is to
	achieve the high lift of a slotted airfoil to delay stall on the helicopter rotor retreating blade
	without incurring a significant drag penalty at the low angles of attack of the advancing blade
	in high-speed forward flight. Another high-lift airfoil approach is to introduce a variable
	geometry, such as a deployable leading edge slat or variable leading edge contour. Variable
	leading edge camber or droop would be especially beneficial since no drag penalty would be
	incurred for azimuthal locations operating at low lift, although such concepts pose demanding
	actuator and structural requirements. Finally, new approaches to active flow control based on
	low-mass flow, oscillatory blowing concepts have emerged.","Advanced Airfoil Design");
}

void C7_2Dlg::OnRadio3()
{
	m_edit2c.SetWindowText("SQUARE TIP - easy to design, easy to manufacture and is a much
	cheaper option.");
}

void C7_2Dlg::OnRadio4()
{
```

```
	m_edit2c.SetWindowText("SWEPTBACK TIPS - produce dynamic twist that improves the
	unbalanced air loading distribution in forward flight; uploaded tip on retreating side increases
	the nose down twist on that side that is beneficial for hover performances; finally the swept-
	back blade tip configuration tends to offer VTUAV speed stability.");
}

void C7_2Dlg::On7_2b()
{
	EndDialog(IDCANCEL);
	C7_1Dlg a;
	a.DoModal();
}

void C7_2Dlg::On2n()
{
	EndDialog(IDCANCEL);
	C7_3Dlg a;
	a.DoModal();
}
```

J.7 'Drag' class

The 'Drag' class is programmed to evaluate coefficients of parasite and profile drag as follows:

```
// 7_4Dlg.cpp : implementation file
// Declaration of header files
#include "stdafx.h"
#include "VTUAV_Expert_System.h"
#include "7_4Dlg.h"
#include "7_3Dlg.h"
#include "7_5Dlg.h"
#include <cmath>  // equivalent of math.h
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

// Declaration & initialisation of class variables
C7_4Dlg::C7_4Dlg(CWnd* pParent /*=NULL*/)
	: CDialog(C7_4Dlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(C7_4Dlg)
	m_14 = _T("");
	m_15 = _T("");
	m_1 = _T("");
	m_2 = _T("");
	m_3 = _T("");
	m_4 = _T("");
	m_5 = _T("");
	m_6 = _T("");
	m_7 = _T("");
	m_8 = _T("");
	m_9a = _T("");
	m_10a = _T("");
	m_11 = _T("");
```

```
	m_12 = _T("");
	m_13 = _T("");
	m_16 = _T("");
	m_17 = _T("");
	m_18 = _T("");
	m_19 = _T("");
	m_20 = _T("");
	m_21 = _T("");
	m_22 = _T("");
	m_23 = _T("");
	m_24 = _T("");
	m_25 = _T("");
	//}}AFX_DATA_INIT
}

// Limiting variables values
void C7_4Dlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(C7_4Dlg)
	DDX_Text(pDX, IDC_EDIT14, m_14);
	DDX_Text(pDX, IDC_EDIT15, m_15);
	DDX_Text(pDX, IDC_EDIT1, m_1);
	DDX_Text(pDX, IDC_EDIT2, m_2);
	DDX_Text(pDX, IDC_EDIT3, m_3);
	DDX_Text(pDX, IDC_EDIT4, m_4);
	DDX_Text(pDX, IDC_EDIT5, m_5);
	DDX_Text(pDX, IDC_EDIT6, m_6);
	DDX_Text(pDX, IDC_EDIT7, m_7);
	DDX_Text(pDX, IDC_EDIT8, m_8);
	DDX_Text(pDX, IDC_EDIT9a, m_9a);
	DDX_Text(pDX, IDC_EDIT10a, m_10a);
	DDX_Text(pDX, IDC_EDIT11, m_11);
	DDX_Text(pDX, IDC_EDIT12, m_12);
	DDX_Text(pDX, IDC_EDIT13, m_13);
	DDX_Text(pDX, IDC_EDIT16, m_16);
	DDX_Text(pDX, IDC_EDIT17, m_17);
	DDX_Text(pDX, IDC_EDIT18, m_18);
	DDX_Text(pDX, IDC_EDIT19, m_19);
	DDX_Text(pDX, IDC_EDIT20, m_20);
	DDX_Text(pDX, IDC_EDIT21, m_21);
	DDX_Text(pDX, IDC_EDIT22, m_22);
	DDX_Text(pDX, IDC_EDIT23, m_23);
	DDX_Text(pDX, IDC_EDIT24, m_24);
	DDX_Text(pDX, IDC_EDIT25, m_25);
	//}}AFX_DATA_MAP
}

// Defining user-actions vis-a-vis system messages
BEGIN_MESSAGE_MAP(C7_4Dlg, CDialog)
	//{{AFX_MSG_MAP(C7_4Dlg)
	ON_BN_CLICKED(ID7_3B, On3b)
	ON_BN_CLICKED(ID7H1, On7h1)
	ON_BN_CLICKED(ID7H11, On7h11)
	ON_BN_CLICKED(ID7H2, On7h2)
	ON_BN_CLICKED(ID7H3, On7h3)
	ON_BN_CLICKED(ID7H4a, On7H4a)
	ON_BN_CLICKED(ID7H5, On7h5)
	ON_BN_CLICKED(ID7H6, On7h6)
	ON_BN_CLICKED(ID7H7, On7h7)
```

```
	ON_BN_CLICKED(ID7H8, On7h8)
	ON_BN_CLICKED(ID7H9, On7h9)
	ON_BN_CLICKED(ID7H10, On7h10)
	ON_BN_CLICKED(ID7H12, On7h12)
	ON_BN_CLICKED(ID7H13, On7h13)
	ON_BN_CLICKED(ID7H14, On7h14)
	ON_BN_CLICKED(ID7H15, On7h15)
	ON_BN_CLICKED(ID7H16, On7h16)
	ON_BN_CLICKED(ID7H17, On7h17)
	ON_BN_CLICKED(ID7H18, On7h18)
	ON_BN_CLICKED(ID7H19, On7h19)
	ON_BN_CLICKED(ID7H20, On7h20)
	ON_BN_CLICKED(ID7H21, On7h21)
	ON_BN_CLICKED(ID7_3N, On3n)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

// Class method – Initialisation of dialog
BOOL C7_4Dlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	extern CString uavname_dbsearch;
	extern CString uavmission_dbsearch;
	extern CString uavengine_dbsearch;
	extern double uavweight_dbsearch;
	extern double uavpower_dbsearch;
	extern double uavspeed_dbsearch;
	extern double uavendurance_dbsearch;
	extern double uavrange_dbsearch;
	extern double uavaltitude_dbsearch;
	extern double uavlength_dbsearch;
	extern double uavwingspan_dbsearch;
	extern double uavrotorspan_dbsearch;
	extern double uavdiameter_dbsearch;
	extern double ewt;//empty weight
	extern double pwt;//payload weight
	extern double lwt;//launch weight
	extern double mt;//mission time
	extern double fr;//fuel reserve
	extern double en;//endurance
	extern double sp;//speed
	extern double al;//altitude
	extern double le;//length
	extern double sn;//span, infact width
	extern double uavprojectedarea;// store the value

	//Variables for calculating power
	extern double DEN_method;//Density
	extern double PRD_method;//profile drag
	extern double PAD_method;//parasite drag

	CString a;

	m_1.Format("%g",al);
	m_2.Format("%g",sp*1.46666667);//change mph to fps and 1 mile / hour = 1.46666667 foot /
	second
	m_3.Format("%g",le);
```

```
//Calculating the air variables
//Local variable declaration
double TEMPSL=518.67;
double RHOSL=0.00237689;
double PRESSSL=2116.22;
double saTheta=1.0;
double saSigma=1.0;
double saDelta=1.0;

double h=al*1.46666667;//convert mph to fps
double v=sp;
double rl=le;

//program for calculation
if(h<232940)
{
        saTheta=1.434843-(h/337634);
        saSigma=pow(0.79899-(h/606330),11.20114);
        saDelta=pow(0.838263-(h/577922),12.20114);
}

if(h<167323)
{
        saTheta=0.939268;
        saSigma=0.00116533*exp((h-154200)/-25992);
        saDelta=0.00109456*exp((h-154200)/-25992);
}

if(h<154199)
{
        saTheta=0.482561+(h/337634);
        saSigma=pow(0.857003+(h/190115),-13.20114);
        saDelta=pow(0.898309+(h/181373),-12.20114);
}

if(h<104987)
{
        saTheta=0.682457+(h/945374);
        saSigma=pow(0.978261+(h/659515),-35.16319);
        saDelta=pow(0.988626+(h/652600),-34.16319);
}

if(h<65617)
{
        saTheta=0.751865;
        saSigma=0.297076*exp((36089-h)/20806);
        saDelta=0.223361*exp((36089-h)/20806);
}

if(h<36089)
{
        saTheta=1.0-(h/145442);
        saSigma=pow(1.0-(h/145442),4.255876);
        saDelta=pow(1.0-(h/145442),5.255876);
}

double tempVal=TEMPSL*saTheta;
double rhoVal=RHOSL*saSigma;
double pVal=PRESSSL*saDelta;
double soundVal=sqrt(1.4*1716.56*(tempVal));
```

```
	double machVal=v/soundVal;
	double qVal=0.7*pVal*machVal*machVal;
	double viscVal=0.0226968*pow(tempVal,1.5)/((tempVal)+198.72)/1000000.0;
	double reynolds=v*rl*rhoVal/viscVal;
	double cfturb=0.455/pow((log(reynolds)/log(10)),2.58);
	double cflam=1.328/sqrt(reynolds);

	m_4.Format("%g",tempVal);
	m_5.Format("%g",rhoVal);
	m_6.Format("%g",pVal);
	m_7.Format("%g",qVal);
	m_8.Format("%g",soundVal);
	m_9a.Format("%g",viscVal);
	m_10a.Format("%g",machVal);
	m_11.Format("%g",reynolds);
	m_12.Format("%g",cflam);
	m_13.Format("%g",cfturb);
	m_14="0.00818";
	m_15.Format("%g",0.00818*(pow((lwt-pwt),(2/3))));
	m_16.Format("%g",1 + (1.5*pow(sn/le,1.5)) + (7*pow(sn/le,3)));

	double f_value=0, PDE;
	if(reynolds < 100000)
	{
		m_17.Format("%g",1.3*pow(reynolds,-0.05));
		f_value=1.3*pow(reynolds,-0.05);
	}
	else
	{
		m_17.Format("%g",0.45*pow(log(reynolds),-2.50));
		f_value=0.45*pow(log(reynolds),-2.50);
	}

	PDE=1.15 * (1 + (1.5*pow(sn/le,1.5)) + (7*pow(sn/le,3))) * (f_value);
	m_18.Format("%g", PDE);
	m_19.Format("%g",(0.00818*(pow((lwt-pwt),(2/3))))+PDE);
	m_20.Format("%g",0.23);
	m_21.Format("%g",5.9);
	m_22.Format("%g",6*0.23/5.9);
	m_23.Format("%g",0.0087 + (0.0126*(6*0.23/5.9)) + (0.400*(6*0.23/5.9)*(6*0.23/5.9)) );
	m_24.Format("%g",0.010 + (0.3*(6*0.23/5.9)*(6*0.23/5.9)));
	m_25.Format("%g",((0.0087 + (0.0126*(6*0.23/5.9)) +
	(0.400*(6*0.23/5.9)*(6*0.23/5.9)))+(0.010 + (0.3*(6*0.23/5.9)*(6*0.23/5.9))))/2);
	UpdateData(FALSE);


	DEN_method=rhoVal;//Density
	PRD_method=((0.0087 + (0.0126*(6*0.23/5.9)) + (0.400*(6*0.23/5.9)*(6*0.23/5.9)))+(0.010
	+ (0.3*(6*0.23/5.9)*(6*0.23/5.9))))/2 ;//profile drag
	PAD_method=(0.00818*(pow((lwt-pwt),(2/3))))+PDE ;//parasite drag
	return TRUE;  // return TRUE unless you set the focus to a control
		// EXCEPTION: OCX Property Pages should return FALSE
}

// Class method – Navigate to previous dialog
void C7_4Dlg::On3b()
{
	EndDialog(IDCANCEL);
	C7_3Dlg a;
	a.DoModal();
```

```
}

void C7_4Dlg::On7h1()
{
	MessageBox("Temp=518.67*st;where st is computed from flying altitude. Also, Celsius=
	((Rankine-491.67)*0.5555)+273.15 ","HOW?");
}

void C7_4Dlg::On7h11()
{
	MessageBox("COEFFICIENT-OF-PARASITE-DRAG-OF-CLEAN-VTUAV-WITH-
	INTERNAL-USEFUL-LOAD = k*pow((W-P),0.667); where 'k' is parasite_drag-
	gross_weight constant, 'W' is Gross weight and 'P' is mission system weight external
	","HOW?");
}

void C7_4Dlg::On7h2()
{
	MessageBox("Density=0.00237689*ss; where ss is computed from flying altitude","HOW?");
}

void C7_4Dlg::On7h3()
{
	MessageBox("Pressure=2116.22*sd; where sd is computed from flying altitude","HOW?");
}

void C7_4Dlg::On7H4a()
{
	MessageBox("Dynamic Pressure = 0.7 * Pressure * MACH_no * MACH_no","HOW?");
}

void C7_4Dlg::On7h5()
{
	MessageBox("Sound Speed=sqrt(1.4*1716.56*(Temp))","HOW?");
}

void C7_4Dlg::On7h6()
{
	MessageBox("Viscosity=(0.0226968*pow(tempVal,1.5))/(((tempVal)+198.72)/1000000.0)","
	HOW?");
}

void C7_4Dlg::On7h7()
{
	MessageBox("Mach Num=Speed/Sound_Speed","HOW?");
}

void C7_4Dlg::On7h8()
{
	MessageBox("Reynolds Num=Speed*Reference_Length *Air_Density/Air_Viscosity",
	"HOW?");
}

void C7_4Dlg::On7h9()
{
	MessageBox("Laminar Cf = 1.328/sqrt(Reynolds_num)","HOW?");
}

void C7_4Dlg::On7h10()
{
```

```
	MessageBox("Turbulent Cf = 0.455/pow((log(Reynolds_num)/log(10)),2.58)","HOW?");
}

void C7_4Dlg::On7h12()
{
	MessageBox("FORM FACTOR = 1 +
	(1.5*pow(diameter_external_load/length_external_load,1.5)) +
	(7*pow(diameter_external_load/length_external_load,3))","HOW?");
}

void C7_4Dlg::On7h13()
{
	MessageBox("IF(Reynolds_num < 100000) then Coefficient_Friction =
	1.3*pow(Reynolds_num,-0.05) ELSE Coefficient_Friction= 0.45* pow (log (Reynolds_num),
	-2.50)","HOW?");
}

void C7_4Dlg::On7h14()
{
	MessageBox("COEFFICIENT OF PARASITE DRAG EXTERNAL = 1.15 * FORM-
	FACTOR * COEFFICIENT-OF-FRICTION","HOW?");
}

void C7_4Dlg::On7h15()
{
	MessageBox(" COEFFICIENT-OF-PARASITE-DRAG = COEFFICIENT-OF-PARASITE-
	DRAG-OF-CLEAN-VTUAV-WITH-INTERNAL-USEFUL-LOAD + COEFFICIENT-OF-
	PARASITE-DRAG-EXTERNAL ","HOW?");
}

void C7_4Dlg::On7h16()
{
	MessageBox("The number of blades is evaluated from the 'thrust coefficient'-to-'solidity ratio'.
	The ratio ranges from 0.17 at sea level and varies with altitude to a maximum of 0.23. Since,
	we are performing analysis at max altitude, therefore max value is assumed.","HOW?");
}

void C7_4Dlg::On7h17()
{
	MessageBox("For conventional airfoils at low Mach number the slope of the lift curve is '5.73'
	per radian. It increases slightly with Mach number and may be assumed to be '5.9' for
	preliminary design","HOW?");
}

void C7_4Dlg::On7h18()
{
	MessageBox("AVERAGE ANGLE OF ATTACK = 6 * THRUST_COEFFICIENT-TO-
	SOLIDITY_RATIO / SLOPE_OF_LIFT_CURVE","HOW?");
}

void C7_4Dlg::On7h19()
{
	MessageBox("COEFFICIENT_OF_PROFILE_DRAG- METHOD 1 = 0.0087 +
	0.0126*(AVERAGE_ANGLE_OF_ATTACK) +
	0.400*(AVERAGE_ANGLE_OF_ATTACK)^2","HOW?");
}

void C7_4Dlg::On7h20()
{
```

```
	MessageBox("COEFFICIENT_OF_PROFILE_DRAG- METHOD 2 = 0.01 +
	0.3*(AVERAGE_ANGLE_OF_ATTACK)^2","HOW?");
}

void C7_4Dlg::On7h21()
{
	MessageBox("","HOW?");
}

void C7_4Dlg::On3n()
{
	EndDialog(IDCANCEL);
	C7_5Dlg a;
	a.DoModal();
}
```

J.8 'Power', 'Database-II', and 'Airframe-section weight' classes

The 'Power', 'Database-II', and 'Airframe-section weight' classes are programmed in a single dialog for evaluation of the power required at maximum speed to sustain forward flight; reselection of engine; and computation of structural weight of various airframe sections of a VTUAV as follows:

```
// 7_5Dlg.cpp : implementation file
// Declaration of header files

#include "stdafx.h"
#include "VTUAV_Expert_System.h"
#include "7_5Dlg.h"
#include "7_4Dlg.h"
#include "7_1Dlg.h"
#include "7_6Dlg.h"
#include <cmath> //equivalent of math.h
#include "_recordset.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

double check=0;
extern double check_final;
extern double uavpower_dbsearch;//reset the power value
extern double engine_weight;//recording the selected engine weight
extern double final_power;//record computed engine power
extern double extreme_result;

//variables for calculating power
extern double ARD_method;//Area of rotor disc
extern double DEN_method;//Density
extern double TS_method;//Tip speed
extern double RR_method;//Rotor radius
extern double SR_method;//Solidity of rotor
```

```
extern double PRD_method;//Profile drag
extern double PAD_method;//Parasite drag

extern double pwt;//Payload weight
extern double lwt;//Launch weight
extern double sp;//Forward speed
extern double le;// Fuselage length
extern double uavprojectedarea;// store the value
extern double fuel_system;//recording fuel weight

// Declaration & initialisation of class variables
C7_5Dlg::C7_5Dlg(CWnd* pParent /*=NULL*/)
	: CDialog(C7_5Dlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(C7_5Dlg)
	m_1 = _T("");
	m_2 = _T("");
	m_3 = _T("");
	m_4 = _T("");
	m_5 = _T("");
	m_6 = _T("");
	m_7 = _T("");
	m_8 = _T("");
	m_9 = _T("");
	m_11 = _T("");
	m_10 = _T("");
	m_12 = _T("");
	m_13 = _T("");
	m_14 = _T("");
	m_15 = _T("");
	m_16 = _T("");
	m_17 = _T("");
	m_18 = _T("");
	m_19 = _T("");
	m_20 = _T("");
	m_21 = _T("");
	m_22 = _T("");
	m_23 = _T("");
	m_24 = _T("");
	m_25 = _T("");
	//}}AFX_DATA_INIT
}

// Limiting variables values
void C7_5Dlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(C7_5Dlg)
	DDX_Control(pDX, ID7_3N, m_next);
	DDX_Control(pDX, IDC_EDIT25, m_25c);
	DDX_Control(pDX, IDC_EDIT24a, m_24c);
	DDX_Control(pDX, IDC_EDIT23, m_23c);
	DDX_Control(pDX, IDC_EDIT22a, m_22c);
	DDX_Control(pDX, IDC_EDIT21, m_21c);
	DDX_Control(pDX, IDC_EDIT20, m_20c);
	DDX_Control(pDX, IDC_EDIT19a, m_19c);
	DDX_Control(pDX, IDC_EDIT18, m_18c);
	DDX_Control(pDX, IDC_EDIT17, m_17c);
	DDX_Control(pDX, IDC_EDIT16, m_16c);
	DDX_Control(pDX, IDC_EDIT15, m_15c);
```

```
	DDX_Control(pDX, IDC_EDIT14, m_14c);
	DDX_Control(pDX, IDC_BUTTON2, m_but2);
	DDX_Control(pDX, IDC_BUTTON1, m_but1);
	DDX_Text(pDX, IDC_EDIT1, m_1);
	DDX_Text(pDX, IDC_EDIT2, m_2);
	DDX_Text(pDX, IDC_EDIT3, m_3);
	DDX_Text(pDX, IDC_EDIT4a, m_4);
	DDX_Text(pDX, IDC_EDIT5a, m_5);
	DDX_Text(pDX, IDC_EDIT6a, m_6);
	DDX_Text(pDX, IDC_EDIT7a, m_7);
	DDX_Text(pDX, IDC_EDIT8a, m_8);
	DDX_Text(pDX, IDC_EDIT9a, m_9);
	DDX_Text(pDX, IDC_EDIT11a, m_11);
	DDX_Text(pDX, IDC_EDIT10a, m_10);
	DDX_Text(pDX, IDC_EDIT12, m_12);
	DDX_Text(pDX, IDC_EDIT13, m_13);
	DDX_Text(pDX, IDC_EDIT14, m_14);
	DDX_Text(pDX, IDC_EDIT15, m_15);
	DDX_Text(pDX, IDC_EDIT16, m_16);
	DDX_Text(pDX, IDC_EDIT17, m_17);
	DDX_Text(pDX, IDC_EDIT18, m_18);
	DDX_Text(pDX, IDC_EDIT19a, m_19);
	DDX_Text(pDX, IDC_EDIT20, m_20);
	DDX_Text(pDX, IDC_EDIT21, m_21);
	DDX_Text(pDX, IDC_EDIT22a, m_22);
	DDX_Text(pDX, IDC_EDIT23, m_23);
	DDX_Text(pDX, IDC_EDIT24a, m_24);
	DDX_Text(pDX, IDC_EDIT25, m_25);
	DDX_Control(pDX, IDC_LABEL1a, m_l1);
	DDX_Control(pDX, IDC_LABEL1b, m_l2);
	DDX_Control(pDX, IDC_LABEL1c, m_l3);
	DDX_Control(pDX, IDC_LABEL1d, m_l4);
	DDX_Control(pDX, IDC_LABEL1e, m_l5);
	DDX_Control(pDX, IDC_LABEL1f, m_l6);
	DDX_Control(pDX, IDC_LABEL1g, m_l7);
	DDX_Control(pDX, IDC_LABEL1h, m_l8);
	DDX_Control(pDX, IDC_LABEL1i, m_l9);
	DDX_Control(pDX, IDC_LABEL1j, m_l10);
	DDX_Control(pDX, IDC_LABEL1k, m_l11);
	DDX_Control(pDX, IDC_LABEL1l, m_l12);
	DDX_Control(pDX, IDC_ADOSEARCH, m_ado);
	DDX_Control(pDX, IDC_ADOSEARCH1, m_ado1);
	DDX_Control(pDX, IDC_DATAGRIDAS, m_dgas);
	DDX_Control(pDX, IDC_DATAGRIDDS1, m_dgds);
	DDX_Control(pDX, IDC_ADOSEARCH3, m_ado2);
	DDX_Control(pDX, IDC_DATALISTID1, m_ls1);
	DDX_Control(pDX, IDC_DATALISTID2, m_ls2);
	DDX_Control(pDX, IDC_ADOSEARCH4, m_ado3);
	//}}AFX_DATA_MAP
}

// Defining user-actions vis-a-vis system messages
BEGIN_MESSAGE_MAP(C7_5Dlg, CDialog)
	//{{AFX_MSG_MAP(C7_5Dlg)
	ON_BN_CLICKED(ID7H1, On7h1)
	ON_BN_CLICKED(ID7_3B, On3b)
	ON_BN_CLICKED(IDC_BUTTON1, OnButton1)
	ON_BN_CLICKED(IDC_BUTTON2, OnButton2)
	ON_BN_CLICKED(IDC_BUTTONengine, OnBUTTONengine)
	ON_BN_CLICKED(IDC_BUTTONengine1, OnBUTTONengine1)
```

```
	ON_BN_CLICKED(ID7_3N, On3n)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

// Class method – Initialisation of dialog
BOOL C7_5Dlg::OnInitDialog()
{
	CDialog::OnInitDialog();
	//Local variables converted into SI units
	double weight=lwt/2.2;//weight in kg
	double disc_area=ARD_method*0.09290304;//m^2 and 1(foot^2)/sec =
	0.09290304(meter^2)/sec
	double density=DEN_method*536.523;// in kg/m^3 and 1 sl/ft3 = 536.523 kg/m3
	double tip_speed=TS_method*0.3048;// in m/sec and  1 foot / second = 0.3048 meter / second
	double forward_velocity=sp*0.44704;//in mps and 1 mile / hour = 0.44704 meter / second
	double rotor_radius=RR_method*0.3048;//in m and 1 foot = 0.3048 meters
	double solidity=SR_method;
	double profile_drag=PRD_method;
	double parasite_drag=PAD_method;

	m_1.Format("%g",weight);
	m_2.Format("%g",disc_area);
	m_3.Format("%g",density);
	m_4.Format("%g",tip_speed);
	m_5.Format("%g",forward_velocity);
	m_6.Format("%g",rotor_radius);
	m_7.Format("%g",solidity);
	m_8.Format("%g",profile_drag);
	m_9.Format("%g",parasite_drag);

	//Enter loop first time only
	if(extreme_result==0)
	{
	check=(weight*weight)/(2*disc_area*density*forward_velocity);//in KW
	check+=(solidity*profile_drag*density*disc_area*pow(tip_speed,2)*(1+(4.65*0.5*0.5)))/8;
	check=check*1.15;//miscellaneous power is 15% extra
	extreme_result=extreme_result + 1;
	}

	else
	{
	check=check_final;
	}

	CString Neetu, Neetu1;
	Neetu.Format("%g", check);
	//Value check-used in extreme cases
	if(check > 2.0*uavpower_dbsearch)
	{
		check=1.80*uavpower_dbsearch;
		Neetu1.Format("%g", check);
		MessageBox("Extreme case max algorithm followed. The program generates
		maximum power value of "+Neetu+" kw and in range value of "+Neetu1+" kw. The
		in-range value is being automatically selected.","BOUNDARY VALUES");
		check_final=check;
	}

	if(check < 0.5*uavpower_dbsearch)
	{
		check=0.60*uavpower_dbsearch;
```

```
		Neetu1.Format("%g", check);
		MessageBox("Extreme case min algorithm followed. The program generates
		minimum power value of "+Neetu+" kw and in range value of "+Neetu1+" kw. The
		in-range value is being automatically selected.","BOUNDARY VALUES");
		check_final=check;
}

m_10.Format("%g",check);
check=check*1.341;//in hp and Power (HP) = Power (kW) x 1.341
m_11.Format("%g",check);
m_12.Format("%g",uavpower_dbsearch);
m_13.Format("%g",fabs((check)-uavpower_dbsearch));
final_power=check;


//make them invisible
m_14c.ShowWindow(SW_HIDE);
m_15c.ShowWindow(SW_HIDE);
m_16c.ShowWindow(SW_HIDE);
m_17c.ShowWindow(SW_HIDE);
m_18c.ShowWindow(SW_HIDE);
m_19c.ShowWindow(SW_HIDE);
m_20c.ShowWindow(SW_HIDE);
m_21c.ShowWindow(SW_HIDE);
m_22c.ShowWindow(SW_HIDE);
m_23c.ShowWindow(SW_HIDE);
m_24c.ShowWindow(SW_HIDE);
m_25c.ShowWindow(SW_HIDE);

//Weight distribution
double balance;
if((0.75 * 6.9 * pow((lwt/1000),0.49) * pow(le,0.61) * pow(uavprojectedarea,0.25))<lwt)
{
		m_14c.ShowWindow(SW_SHOW);
		m_14.Format("%g",0.75 * 6.9 * pow((lwt/1000),0.49) * pow(le,0.61) *
		pow(uavprojectedarea,0.25) );
		balance=lwt-((0.75 * 6.9 * pow((lwt/1000),0.49) * pow(le,0.61) *
		pow(uavprojectedarea,0.25)));
}

if((0.75 * 2 * 0.026 * pow(2*RR_method,0.66) * 0.4265 * pow(RR_method,1.3) *
pow(TS_method,0.67)) < balance)
{
		m_15c.ShowWindow(SW_SHOW);
		m_15.Format("%g", 0.75 * 2 * 0.026 * pow(2*RR_method,0.66) * 0.4265 *
		pow(RR_method,1.3) * pow(TS_method,0.67));//chord length is assumed at 0.13m
		or 3.2808399 *.13 = 0.426509187 ft. Also rotor span is twice the radius
		balance-=0.75 * 2 * 0.026 * pow(2*RR_method,0.66) * 0.4265 *
		pow(RR_method,1.3) * pow(TS_method,0.67) ;
}

if(((0.75 * 0.041 * pow(engine_weight,1.1)) + (0.33*pow(uavprojectedarea,1.3)))<balance)
{
		m_16c.ShowWindow(SW_SHOW);
		m_16.Format("%g",(0.75 * 0.041 * pow(engine_weight,1.1)) +
		(0.33*pow(uavprojectedarea,1.3)));
		balance-=(0.75 * 0.041 * pow(engine_weight,1.1)) +
		(0.33*pow(uavprojectedarea,1.3));
}
```

```
if(engine_weight < balance)
{
	m_17c.ShowWindow(SW_SHOW);
	m_17.Format("%g",engine_weight);
	balance-=engine_weight;
}

if((0.5*13.6*pow((check),0.82) * pow(5.5,0.37) / pow((TS_method/RR_method),0.64))
<balance)
{
	m_18c.ShowWindow(SW_SHOW);
	m_18.Format("%g", 0.5*13.6*pow((check),0.82) * pow(5.5,0.37) /
	pow((TS_method/RR_method),0.64));//rpm assumed at 5500
	balance-=0.5*13.6*pow((check),0.82) * pow(5.5,0.37) /
	pow((TS_method/RR_method),0.64);
}

if((0.75*2*pow(engine_weight,0.59))<balance)
{
	m_19c.ShowWindow(SW_SHOW);
	m_19.Format("%g",0.75*2*pow(engine_weight,0.59));
	balance -=0.75*2*pow(engine_weight,0.59);
}

if(fuel_system<balance)
{
	m_20c.ShowWindow(SW_SHOW);
	m_20.Format("%g",fuel_system);
	balance-=fuel_system;
}

if((3.2*pow((check),0.65) / pow((lwt/1000),0.4))<balance)// formula is 9.6*pow((check),0.65)
/ pow((lwt/1000),0.4) but i have changed it
{
	m_21c.ShowWindow(SW_SHOW);
	m_21.Format("%g",3.2*pow((check),0.65) / pow((lwt/1000),0.4));
	balance-=3.2*pow((check),0.65) / pow((lwt/1000),0.4);
}

if(pwt<balance)
{
	m_22c.ShowWindow(SW_SHOW);
	m_22.Format("%g",pwt);
	balance-=pwt;
}

UpdateData(FALSE);

if(((check)-uavpower_dbsearch)!=0)
{
	m_but1.ShowWindow(SW_SHOW);
	m_but2.ShowWindow(SW_HIDE);
}

else
{
	m_but1.ShowWindow(SW_HIDE);
	m_but2.ShowWindow(SW_SHOW);
}
```

```
	return TRUE;
}

void C7_5Dlg::On7h1()
{
	MessageBox("Power = (weight*weight)/(2*disc_area*density*forward_velocity) +
	((solidity*profile_drag*density*disc_area*pow(tip_speed,2)*(1+(4.65*0.5*0.5)))/8) +
	((density*parasite_drag*disc_area*pow(forward_velocity,3))/2)","How?");
}

void C7_5Dlg::On3b()
{
	EndDialog(IDCANCEL);
	C7_4Dlg a;
	a.DoModal();
}

void C7_5Dlg::OnButton1()
{
	MessageBox("The assumed power value and the computed power value are different. The
	methodology needs to be iterated. You will be automatically redirected to first methodology
	dialog. Please observe new values", "Iteration Required");
	uavpower_dbsearch = check;
	EndDialog(IDCANCEL);
	C7_1Dlg a;
	a.DoModal();
}

void C7_5Dlg::OnButton2()
{
	double val=check;
	CString query,query1,lwts;

	query = "SELECT * FROM UAVENGINELIST WHERE (POWER > ";
	lwts.Format("%g",val);
	query+=lwts;
	query+=") ORDER BY POWER, WEIGHT";

	query1= "SELECT * FROM UAVENGINELIST WHERE (POWER < ";
	query1+=lwts;
	query1+=") ORDER BY POWER DESC, WEIGHT";

	m_ado.SetRecordSource(query);
	m_ado.Refresh();
	m_ado1.SetRecordSource(query1);
	m_ado1.Refresh();
}

void C7_5Dlg::OnBUTTONengine()
{
	char  *stopstring;
	m_23c.ShowWindow(SW_SHOW);
	m_24c.ShowWindow(SW_SHOW);
	m_25c.ShowWindow(SW_SHOW);

	CString query_check;
	query_check = "SELECT * FROM UAVENGINELIST WHERE MODEL ='";
	query_check+=m_dgas.GetText();
	query_check+="'";
	m_ado1.SetRecordSource(query_check);
```

```
m_ado1.Refresh();

if(m_ado1.GetRecordset().GetRecordCount() > 0)
{
	m_23=m_dgas.GetText();
	query_check = "SELECT WEIGHT FROM UAVENGINELIST WHERE MODEL
	='";
	query_check+=m_dgas.GetText();
	query_check+="'";
	m_ado1.SetRecordSource(query_check);
	m_ado1.Refresh();
	m_24=m_dgds.GetText();
	engine_weight= strtod(m_dgds.GetText(),&stopstring);

	query_check = "SELECT POWER FROM UAVENGINELIST WHERE MODEL
	='";
	query_check+=m_dgas.GetText();
	query_check+="'";
	m_ado1.SetRecordSource(query_check);
	m_ado1.Refresh();
	m_25=m_dgds.GetText();

	//Name of closest uav
	query_check = "SELECT TOP 1 NAME FROM Querytable WHERE POWER >= ";
	query_check+= m_dgds.GetText();
	query_check+= " ORDER BY POWER";
	m_ado2.SetRecordSource(query_check);
	m_ado2.Refresh();

	query_check = "SELECT TOP 1 NAME FROM Querytable WHERE POWER < ";
	query_check+= m_dgds.GetText();
	query_check+= " ORDER BY POWER DESC";
	m_ado3.SetRecordSource(query_check);
	m_ado3.Refresh();

	//Resetting the other datagrid
	query_check="SELECT POWER FROM UAVENGINELIST WHERE MODEL =
	'Ankush'";
	m_ado1.SetRecordSource(query_check);
	m_ado1.Refresh();
}

else
{
	MessageBox("Please select the name field only. You may have to click the search
	button once more.","Warning");
}


//2nd part - recalculate the weight values
//weight distribution
double balance;

if((0.75 * 6.9 * pow((lwt/1000),0.49) * pow(le,0.61) * pow(uavprojectedarea,0.25))<lwt)
{
//m_l1.ShowWindow(true);
	m_14c.ShowWindow(SW_SHOW);
	m_14.Format("%g",0.75 * 6.9 * pow((lwt/1000),0.49) * pow(le,0.61) *
	pow(uavprojectedarea,0.25) );
```

```
		balance=lwt-((0.75 * 6.9 * pow((lwt/1000),0.49) * pow(le,0.61) *
		pow(uavprojectedarea,0.25)));
}

if((0.75 * 2 * 0.026 * pow(2*RR_method,0.66) * 0.4265 * pow(RR_method,1.3) *
pow(TS_method,0.67)) < balance)
{
		//m_l2.ShowWindow(true);
		m_15c.ShowWindow(SW_SHOW);
		m_15.Format("%g", 0.75 * 2 * 0.026 * pow(2*RR_method,0.66) * 0.4265 *
		pow(RR_method,1.3) * pow(TS_method,0.67));//chord length is assumed at 0.13m
		or 3.2808399 *.13 = 0.426509187 ft. Also rotor span is twice the radius
		balance-=0.75 * 2 * 0.026 * pow(2*RR_method,0.66) * 0.4265 *
		pow(RR_method,1.3) * pow(TS_method,0.67) ;
}

if(((0.75 * 0.041 * pow(engine_weight,1.1)) + (0.33*pow(uavprojectedarea,1.3)))<balance)
{
		//m_l3.ShowWindow(true);
		m_16c.ShowWindow(SW_SHOW);
		m_16.Format("%g",(0.75 * 0.041 * pow(engine_weight,1.1)) +
		(0.33*pow(uavprojectedarea,1.3)));
		balance-=(0.75 * 0.041 * pow(engine_weight,1.1)) +
		(0.33*pow(uavprojectedarea,1.3));
}

if(engine_weight < balance)
{
		//m_l4.ShowWindow(true);
		m_17c.ShowWindow(SW_SHOW);
		m_17.Format("%g",engine_weight);
		balance-=engine_weight;
}

if((0.5*13.6*pow((check),0.82) * pow(5.5,0.37) /
pow((TS_method/RR_method),0.64))<balance)
{
		//m_l5.ShowWindow(true);
		m_18c.ShowWindow(SW_SHOW);
		m_18.Format("%g", 0.5*13.6*pow((check),0.82) * pow(5.5,0.37) /
		pow((TS_method/RR_method),0.64));//rpm assumed at 5500
		balance-=0.5*13.6*pow((check),0.82) * pow(5.5,0.37) /
		pow((TS_method/RR_method),0.64);
}

if((0.75*2*pow(engine_weight,0.59))<balance)
{
		//m_l6.ShowWindow(true);
		m_19c.ShowWindow(SW_SHOW);
		m_19.Format("%g",0.75*2*pow(engine_weight,0.59));
		balance-=0.75*2*pow(engine_weight,0.59);
}

if(fuel_system<balance)
{
		//m_l7.ShowWindow(true);
		m_20c.ShowWindow(SW_SHOW);
		m_20.Format("%g",fuel_system);
		balance-=fuel_system;
}
```

```
	if((3.2*pow((check),0.65) / pow((lwt/1000),0.4))<balance)
	{
		//m_l8.ShowWindow(true);
		m_21c.ShowWindow(SW_SHOW);
		m_21.Format("%g",3.2*pow((check),0.65) / pow((lwt/1000),0.4));
		balance-=3.2*pow((check),0.65) / pow((lwt/1000),0.4);
	}

	if(pwt<balance)
	{
		//m_l9.ShowWindow(true);
		m_22c.ShowWindow(SW_SHOW);
		m_22.Format("%g",pwt);
		balance-=pwt;
	}

	//proceed further
	m_next.ShowWindow(SW_SHOW);
	MessageBox("Please Click on Next Button to determine C.G.","Proceed Further");
	UpdateData(FALSE);
}

void C7_5Dlg::OnBUTTONengine1()
{
	char  *stopstring;//pointer to end of number in a string
	m_23c.ShowWindow(SW_SHOW);
	m_24c.ShowWindow(SW_SHOW);
	m_25c.ShowWindow(SW_SHOW);

	CString query_check;
	query_check = "SELECT * FROM UAVENGINELIST WHERE MODEL ='";
	query_check+=m_dgds.GetText();
	query_check+="'";
	m_ado.SetRecordSource(query_check);
	 m_ado.Refresh();

	if(m_ado.GetRecordset().GetRecordCount() > 0)
	{
		m_23=m_dgds.GetText();
		query_check = "SELECT WEIGHT FROM UAVENGINELIST WHERE MODEL
		='";
		query_check+=m_dgds.GetText();
		query_check+="'";
		m_ado.SetRecordSource(query_check);
		m_ado.Refresh();
		m_24=m_dgas.GetText();
		engine_weight= strtod(m_dgas.GetText(),&stopstring);

		query_check = "SELECT POWER FROM UAVENGINELIST WHERE MODEL
		='";
		query_check+=m_dgds.GetText();
		query_check+="'";
		m_ado.SetRecordSource(query_check);
		m_ado.Refresh();
		m_25=m_dgas.GetText();

		//name of closest uav
		query_check = "SELECT TOP 1 NAME FROM Querytable WHERE POWER >= ";
```

```
		query_check+= m_dgas.GetText();
		query_check+= " ORDER BY POWER";
		m_ado2.SetRecordSource(query_check);
		m_ado2.Refresh();

		query_check = "SELECT TOP 1 NAME FROM Querytable WHERE POWER < ";
		query_check+= m_dgas.GetText();
		query_check+= " ORDER BY POWER DESC";
		m_ado3.SetRecordSource(query_check);
		m_ado3.Refresh();

		//Resetting the other datagrid
		query_check="SELECT POWER FROM UAVENGINELIST WHERE MODEL =
		'Ankush'";
		m_ado.SetRecordSource(query_check);
		m_ado.Refresh();
}

else
{
		MessageBox("Please select the name field only. You may have to click the search
		button once more.","Warning");
}


//2nd part - recalculate the weight values
//weight distribution
double balance;

if((0.75 * 6.9 * pow((lwt/1000),0.49) * pow(le,0.61) * pow(uavprojectedarea,0.25))<lwt)
{
		//m_l1.ShowWindow(true);
		m_14c.ShowWindow(SW_SHOW);
		m_14.Format("%g",0.75 * 6.9 * pow((lwt/1000),0.49) * pow(le,0.61) *
		pow(uavprojectedarea,0.25) );
		balance=lwt-((0.75 * 6.9 * pow((lwt/1000),0.49) * pow(le,0.61) *
		pow(uavprojectedarea,0.25)));
}

if((0.75 * 2 * 0.026 * pow(2*RR_method,0.66) * 0.4265 * pow(RR_method,1.3) *
pow(TS_method,0.67)) < balance)
{
		//m_l2.ShowWindow(true);
		m_15c.ShowWindow(SW_SHOW);
		m_15.Format("%g", 0.75 * 2 * 0.026 * pow(2*RR_method,0.66) * 0.4265 *
		pow(RR_method,1.3) * pow(TS_method,0.67));//chord length is assumed at 0.13m
		or 3.2808399 *.13 = 0.426509187 ft. Also rotor span is twice the radius
		balance-=0.75 * 2 * 0.026 * pow(2*RR_method,0.66) * 0.4265 *
		pow(RR_method,1.3) * pow(TS_method,0.67) ;
}

if(((0.75 * 0.041 * pow(engine_weight,1.1)) + (0.33*pow(uavprojectedarea,1.3)))<balance)
{
		//m_l3.ShowWindow(true);
		m_16c.ShowWindow(SW_SHOW);
		m_16.Format("%g",(0.75 * 0.041 * pow(engine_weight,1.1)) +
		(0.33*pow(uavprojectedarea,1.3)));
		balance-=(0.75 * 0.041 * pow(engine_weight,1.1)) +
		(0.33*pow(uavprojectedarea,1.3));
}
```

```
	if(engine_weight < balance)
	{
		//m_l4.ShowWindow(true);
		m_17c.ShowWindow(SW_SHOW);
		m_17.Format("%g",engine_weight);
		balance-=engine_weight;
	}

	if((0.5*13.6*pow((check),0.82) * pow(5.5,0.37) /
	pow((TS_method/RR_method),0.64))<balance)
	{
		//m_l5.ShowWindow(true);
		m_18c.ShowWindow(SW_SHOW);
		m_18.Format("%g", 0.5*13.6*pow((check),0.82) * pow(5.5,0.37) /
		pow((TS_method/RR_method),0.64));//rpm assumed at 5500
		balance-=0.5*13.6*pow((check),0.82) * pow(5.5,0.37) /
		pow((TS_method/RR_method),0.64);
	}

	if((0.75*2*pow(engine_weight,0.59))<balance)
	{
		m_19c.ShowWindow(SW_SHOW);
		m_19.Format("%g",0.75*2*pow(engine_weight,0.59));
		balance-=0.75*2*pow(engine_weight,0.59);
	}

	if(fuel_system<balance)
	{
		m_20c.ShowWindow(SW_SHOW);
		m_20.Format("%g",fuel_system);
		balance-=fuel_system;
	}

	if((3.2*pow((check),0.65) / pow((lwt/1000),0.4))<balance
	{
		m_21c.ShowWindow(SW_SHOW);
		m_21.Format("%g",3.2*pow((check),0.65) / pow((lwt/1000),0.4));
		balance-=3.2*pow((check),0.65) / pow((lwt/1000),0.4);
	}

	if(pwt<balance)
	{
		m_22c.ShowWindow(SW_SHOW);
		m_22.Format("%g",pwt);
		balance-=pwt;
	}

	//Proceed further
	m_next.ShowWindow(SW_SHOW);
	MessageBox("Please Click on Next Button to determine C.G.","Proceed Further");
	UpdateData(FALSE);
}

void C7_5Dlg::On3n()
{
	EndDialog(IDCANCEL);
	C7_6Dlg a;
	a.DoModal();
}
```

I.9 ‘Balancing’ and ‘Design-output’ classes

The ‘Balancing’, and ‘Design-output’ classes are programmed together and involve ‘Weight and balance’ exercise to refine the inboard and external profile of the VTUAV to position the cg within the acceptable limits as follows:

```
// 7_6Dlg.cpp : implementation file
// Declaration of header files
#include "stdafx.h"
#include "VTUAV_Expert_System.h"
#include "7_6Dlg.h"
#include "7_5Dlg.h"
#include "7_7Dlg.h"
#include "cmath"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

// Declaration & initialisation of class variables
extern double uavpower_dbsearch;//reset the power value
extern double engine_weight;//recording engine weight
extern double uavprojectedarea;// store the value
extern double fuel_system;//recording fuel weight

//Variables for calculating power
extern double ARD_method;//Area of rotor disc
extern double DEN_method;//Density
extern double TS_method;//Tip Speed
extern double RR_method;//Rotor radius
extern double SR_method;//Solidity of rotor
extern double PRD_method;//profile drag
extern double PAD_method;//parasite drag
extern double pwt;//payload weight
extern double lwt;//launch weight
extern double sp;//speed
extern double le;//length
extern double final_power;//record computed engine power
extern double wsn;//wing span
extern double SENSORARRAY[1];// record of individual weight
extern int i1;
extern double w1;
extern double TRANSPONDER[6];
extern int i2;
extern double w2;
extern double TRANSMITTER[2];
extern int i3;
extern double w3;
extern double TERMINAL[2];
extern int i4;
extern double w4;
extern double RECORDER[2];
extern int i5;
extern double w5;
```

```
extern double INTERCEPTOR[1];
extern int i6;
extern double w6;
extern double IMAGINGPROCESSOR[12];
extern int i7;
extern double w7;
extern double OBSERVATIONDISPLAYUNIT[5];
extern int i8;
extern double w8;
extern double CAMERAMODULARACCESSORIESSYSTEM[15];
extern int i9;
extern double w9;
extern double MODULARPAYLOAD[4];
extern int i10;
extern double w10;
extern double RWRESMSIGNITSYSTEM[5];
extern int i11;
extern double w11;
extern double RADARDECOYJAMMERS[2];
extern int i12;
extern double w12;
extern double FLIRS[7];
extern int i13;
extern double w13;
extern double ONBOARDRADAR[9];
extern int i14;
extern double w14;
double total_moment;
double total_weight;
double c;//c.g value

// Limiting variables values
C7_6Dlg::C7_6Dlg(CWnd* pParent /*=NULL*/)
	: CDialog(C7_6Dlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(C7_6Dlg)
	m_Ba = _T("");
	m_Ra = _T("");
	m_Da = _T("");
	m_Ma = _T("");
	m_Pa = _T("");
	m_Na = _T("");
	m_Ea = _T("");
	m_Fa = _T("");
	m_Bb = _T("");
	m_Bc = _T("");
	m_Db = _T("");
	m_Dc = _T("");
	m_Eb = _T("");
	m_Ec = _T("");
	m_Fb = _T("");
	m_Fc = _T("");
	m_Mb = _T("");
	m_Mc = _T("");
	m_Nb = _T("");
	m_Nc = _T("");
	m_Pb = _T("");
	m_Pc = _T("");
	m_Rb = _T("");
	m_Rc = _T("");
```

```
m_M1 = _T("");
m_M2 = _T("");
m_W1 = _T("");
m_W2 = _T("");
m_CG1 = _T("");
m_CG2 = _T("");
m_V1 = _T("");
m_V2 = _T("");
m_san = _T("");
m_saw = _T("");
m_sas = _T("");
m_sam = _T("");
m_trm = _T("");
m_trn = _T("");
m_trs = _T("");
m_trw = _T("");
m_trnm = _T("");
m_trnn = _T("");
m_trns = _T("");
m_trnw = _T("");
m_tem = _T("");
m_ten = _T("");
m_tes = _T("");
m_tew = _T("");
m_rem = _T("");
m_ren = _T("");
m_res = _T("");
m_rew = _T("");
m_inm = _T("");
m_inn = _T("");
m_ins = _T("");
m_inw = _T("");
m_ipm = _T("");
m_ipn = _T("");
m_ips = _T("");
m_ipw = _T("");
m_oum = _T("");
m_oun = _T("");
m_ous = _T("");
m_ouw = _T("");
m_csm = _T("");
m_csn = _T("");
m_css = _T("");
m_csw = _T("");
m_mpn = _T("");
m_mpm = _T("");
m_mps = _T("");
m_mpw = _T("");
m_rsm = _T("");
m_rsn = _T("");
m_rss = _T("");
m_rsw = _T("");
m_rjm = _T("");
m_rjn = _T("");
m_rjs = _T("");
m_rjw = _T("");
m_flm = _T("");
m_fln = _T("");
m_fls = _T("");
m_flw = _T("");
```

```
	m_ram = _T("");
	m_ran = _T("");
	m_ras = _T("");
	m_raw = _T("");
	m_M3 = _T("");
	m_W3 = _T("");
	m_radio = 0;
	//}}AFX_DATA_INIT
}

void C7_6Dlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(C7_6Dlg)
	DDX_Text(pDX, IDC_EDITBa, m_Ba);
	DDX_Text(pDX, IDC_EDITRa, m_Ra);
	DDX_Text(pDX, IDC_EDITDa, m_Da);
	DDX_Text(pDX, IDC_EDITMa, m_Ma);
	DDX_Text(pDX, IDC_EDITPa, m_Pa);
	DDX_Text(pDX, IDC_EDITNa, m_Na);
	DDX_Text(pDX, IDC_EDITEa, m_Ea);
	DDX_Text(pDX, IDC_EDITFa, m_Fa);
	DDX_Text(pDX, IDC_EDITBb, m_Bb);
	DDX_Text(pDX, IDC_EDITBc, m_Bc);
	DDX_Text(pDX, IDC_EDITDb, m_Db);
	DDX_Text(pDX, IDC_EDITDc, m_Dc);
	DDX_Text(pDX, IDC_EDITEb, m_Eb);
	DDX_Text(pDX, IDC_EDITEc, m_Ec);
	DDX_Text(pDX, IDC_EDITFb, m_Fb);
	DDX_Text(pDX, IDC_EDITFc, m_Fc);
	DDX_Text(pDX, IDC_EDITMb, m_Mb);
	DDX_Text(pDX, IDC_EDITMc, m_Mc);
	DDX_Text(pDX, IDC_EDITNb, m_Nb);
	DDX_Text(pDX, IDC_EDITNc, m_Nc);
	DDX_Text(pDX, IDC_EDITPb, m_Pb);
	DDX_Text(pDX, IDC_EDITPc, m_Pc);
	DDX_Text(pDX, IDC_EDITRb, m_Rb);
	DDX_Text(pDX, IDC_EDITRc, m_Rc);
	DDX_Text(pDX, IDC_EDITM1, m_M1);
	DDX_Text(pDX, IDC_EDITM2, m_M2);
	DDX_Text(pDX, IDC_EDITW1, m_W1);
	DDX_Text(pDX, IDC_EDITW2, m_W2);
	DDX_Text(pDX, IDC_EDITCG1, m_CG1);
	DDX_Text(pDX, IDC_EDITCG2, m_CG2);
	DDX_Text(pDX, IDC_EDITV1, m_V1);
	DDX_Text(pDX, IDC_EDITV2, m_V2);
	DDX_Text(pDX, IDC_EDITsan, m_san);
	DDX_Text(pDX, IDC_EDITsaw, m_saw);
	DDX_Text(pDX, IDC_EDITsas, m_sas);
	DDX_Text(pDX, IDC_EDITsam, m_sam);
	DDX_Text(pDX, IDC_EDITtrm, m_trm);
	DDX_Text(pDX, IDC_EDITtrn, m_trn);
	DDX_Text(pDX, IDC_EDITtrs, m_trs);
	DDX_Text(pDX, IDC_EDITtrw, m_trw);
	DDX_Text(pDX, IDC_EDITtrnm, m_trnm);
	DDX_Text(pDX, IDC_EDITtrnn, m_trnn);
	DDX_Text(pDX, IDC_EDITtrns, m_trns);
	DDX_Text(pDX, IDC_EDITtrnw, m_trnw);
	DDX_Text(pDX, IDC_EDITtem, m_tem);
	DDX_Text(pDX, IDC_EDITten, m_ten);
```

```
	DDX_Text(pDX, IDC_EDITtes, m_tes);
	DDX_Text(pDX, IDC_EDITtew, m_tew);
	DDX_Text(pDX, IDC_EDITrem, m_rem);
	DDX_Text(pDX, IDC_EDITren, m_ren);
	DDX_Text(pDX, IDC_EDITres, m_res);
	DDX_Text(pDX, IDC_EDITrew, m_rew);
	DDX_Text(pDX, IDC_EDITinm, m_inm);
	DDX_Text(pDX, IDC_EDITinn, m_inn);
	DDX_Text(pDX, IDC_EDITins, m_ins);
	DDX_Text(pDX, IDC_EDITinw, m_inw);
	DDX_Text(pDX, IDC_EDITipm, m_ipm);
	DDX_Text(pDX, IDC_EDITipn, m_ipn);
	DDX_Text(pDX, IDC_EDITips, m_ips);
	DDX_Text(pDX, IDC_EDITipw, m_ipw);
	DDX_Text(pDX, IDC_EDITоum, m_oum);
	DDX_Text(pDX, IDC_EDITоun, m_oun);
	DDX_Text(pDX, IDC_EDITоus, m_ous);
	DDX_Text(pDX, IDC_EDITоuw, m_ouw);
	DDX_Text(pDX, IDC_EDITcsm, m_csm);
	DDX_Text(pDX, IDC_EDITcsn, m_csn);
	DDX_Text(pDX, IDC_EDITcss, m_css);
	DDX_Text(pDX, IDC_EDITcsw, m_csw);
	DDX_Text(pDX, IDC_EDITmpn, m_mpn);
	DDX_Text(pDX, IDC_EDITmpm, m_mpm);
	DDX_Text(pDX, IDC_EDITmps, m_mps);
	DDX_Text(pDX, IDC_EDITmpw, m_mpw);
	DDX_Text(pDX, IDC_EDITrsm, m_rsm);
	DDX_Text(pDX, IDC_EDITrsn, m_rsn);
	DDX_Text(pDX, IDC_EDITrss, m_rss);
	DDX_Text(pDX, IDC_EDITrsw, m_rsw);
	DDX_Text(pDX, IDC_EDITrjm, m_rjm);
	DDX_Text(pDX, IDC_EDITrjn, m_rjn);
	DDX_Text(pDX, IDC_EDITrjs, m_rjs);
	DDX_Text(pDX, IDC_EDITrjw, m_rjw);
	DDX_Text(pDX, IDC_EDITflm, m_flm);
	DDX_Text(pDX, IDC_EDITfln, m_fln);
	DDX_Text(pDX, IDC_EDITfls, m_fls);
	DDX_Text(pDX, IDC_EDITflw, m_flw);
	DDX_Text(pDX, IDC_EDITram, m_ram);
	DDX_Text(pDX, IDC_EDITran, m_ran);
	DDX_Text(pDX, IDC_EDITras, m_ras);
	DDX_Text(pDX, IDC_EDITraw, m_raw);
	DDX_Text(pDX, IDC_EDITM3, m_M3);
	DDX_Text(pDX, IDC_EDITW3, m_W3);
	DDX_Radio(pDX, IDC_RADIO1, m_radio);
	DDX_Control(pDX, IDC_PICTURECLIP1, m_pic1);
	DDX_Control(pDX, IDC_PICTURECLIP1a, m_pic2);
	//}}AFX_DATA_MAP
}

// Defining user-actions vis-a-vis system messages
BEGIN_MESSAGE_MAP(C7_6Dlg, CDialog)
	//{{AFX_MSG_MAP(C7_6Dlg)
	ON_BN_CLICKED(ID7H2, On7h2)
	ON_BN_CLICKED(ID7P, On7p)
	ON_BN_CLICKED(ID7_3B, On3b)
	ON_BN_CLICKED(IDC_RADIO2, OnRadio2)
	ON_BN_CLICKED(IDC_RADIO3, OnRadio3)
	ON_BN_CLICKED(IDC_RADIO1, OnRadio1)
	ON_BN_CLICKED(ID7F, On7f)
```

```
	ON_BN_CLICKED(IDC_RADIO4, OnRadio4)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()


void C7_6Dlg::On7h2()
{
	MessageBox("Payload Positioning Aims: 1) To distribute the load symmetrically around the
	center of gravity of the aircraft. 2)To mount the payload such that there are no large moment
	arms. 3)Payload to be placed on the underside of the fuselage, so the infra-red camera can be
	pointed down to view the target area. 4)Mounted such that vibrations absorbed by the Infra-
	red camera are minimized.","AIM");
}

void C7_6Dlg::On7p()
{
	MessageBox("Longitudinal Position of the center of gravity of the empty VTUAV is
	calculated from the sum of the static moments about some arbitrary point contributed by each
	group that makes up the empty weight divided by that weight. The arbitrary point should be
	selected ahead of and below the nose so that all parts of the aircraft will have positive
	locations. A variation under 7% is an acceptable result.","PROCEDURE");
}

// Class method – Initialisation of dialog
BOOL C7_6Dlg::OnInitDialog()
{
	CDialog::OnInitDialog();
	//C.G without payload - recalculate the weight values
	//weight distribution

	double balance, check;
	check=final_power;

	//weight var
	double body_weight=0, rotor_weight=0, nacelle_weight=0;
	double eng_weight=0, drysystem_weight=0, propulsionsubsys_weight=0;
	double fuel_weight=0, electricalsys_weight=0;

	//distance var
	double body_station=0, rotor_station=0, nacelle_station=0;
	double eng_station=0, drysystem_station=0, propulsionsubsys_station=0;
	double fuel_station=0, electricalsys_station=0;


	if((0.75 * 6.9 * pow((lwt/1000),0.49) * pow(le,0.61) * pow(uavprojectedarea,0.25))<lwt)
	{
		body_weight=0.75 * 6.9 * pow((lwt/1000),0.49) * pow(le,0.61) *
		pow(uavprojectedarea,0.25);
		balance=lwt-((0.75 * 6.9 * pow((lwt/1000),0.49) * pow(le,0.61) *
		pow(uavprojectedarea,0.25)));
	}

	if((0.75 * 2 * 0.026 * pow(2*RR_method,0.66) * 0.4265 * pow(RR_method,1.3) *
	pow(TS_method,0.67)) < balance)
	{
		rotor_weight = 0.75 * 2 * 0.026 * pow(2*RR_method,0.66) * 0.4265 *
		pow(RR_method,1.3) * pow(TS_method,0.67);//chord length is assumed at 0.13m or
		3.2808399 *.13 = 0.426509187 ft. Also rotor span is twice the radius
		balance-=0.75 * 2 * 0.026 * pow(2*RR_method,0.66) * 0.4265 *
		pow(RR_method,1.3) * pow(TS_method,0.67) ;
```

```
}

if(((0.75 * 0.041 * pow(engine_weight,1.1)) + (0.33*pow(uavprojectedarea,1.3)))<balance)
{
	nacelle_weight = (0.75 * 0.041 * pow(engine_weight,1.1)) +
	(0.33*pow(uavprojectedarea,1.3));
	balance-=(0.75 * 0.041 * pow(engine_weight,1.1)) +
	(0.33*pow(uavprojectedarea,1.3));
}

if(engine_weight < balance)
{
	eng_weight = engine_weight;
	balance-=engine_weight;
}

if((0.5*13.6*pow((check),0.82) * pow(5.5,0.37) /
pow((TS_method/RR_method),0.64))<balance)
{
	drysystem_weight = 0.5*13.6*pow((check),0.82) * pow(5.5,0.37) /
	pow((TS_method/RR_method),0.64);//rpm assumed at 5500
	balance-=0.5*13.6*pow((check),0.82) * pow(5.5,0.37) /
	pow((TS_method/RR_method),0.64);
}

if((0.75*2*pow(engine_weight,0.59))<balance)
{
	propulsionsubsys_weight = 0.75*2*pow(engine_weight,0.59);
	balance-=0.75*2*pow(engine_weight,0.59);
}

if(fuel_system<balance)
{
	fuel_weight = fuel_system;
	balance-=fuel_system;
}

if((3.2*pow((check),0.65) / pow((lwt/1000),0.4))<balance)
{
	electricalsys_weight = 3.2*pow((check),0.65) / pow((lwt/1000),0.4);
	balance-=3.2*pow((check),0.65) / pow((lwt/1000),0.4);
}

//check if the leftover weight can be adjusted
if(balance>0)
{

	if(body_weight == 0)
	body_weight = balance;

	if(rotor_weight == 0)
	rotor_weight = balance;

	if(nacelle_weight == 0)
	nacelle_weight = balance;

	if(eng_weight == 0)
	eng_weight = balance;

	if(drysystem_weight == 0)
```

```
            drysystem_weight = balance;

            if(propulsionsubsys_weight == 0)
            propulsionsubsys_weight = balance;

            if(fuel_weight == 0)
            fuel_weight = balance;

            if(electricalsys_weight == 0)
             electricalsys_weight = balance;
}

m_Ba.Format("%g", body_weight);
m_Ra.Format("%g", rotor_weight);
m_Da.Format("%g", drysystem_weight);
m_Ma.Format("%g", eng_weight);
m_Pa.Format("%g", propulsionsubsys_weight);
m_Na.Format("%g", nacelle_weight);
m_Fa.Format("%g", fuel_weight);
m_Ea.Format("%g", electricalsys_weight);

//determine the type of aircraft and find the cg of the body
//D'Gopher VTUAV
if(wsn>0)
{
            c= 2*le/3;
            body_station = c;
            rotor_station = 0.95*c;
            eng_station = 0.96*c;
            propulsionsubsys_station = 0.97*c;
            nacelle_station = 0.98*c;
            fuel_station = 0.88*c;
            drysystem_station = 1.20*c;
            electricalsys_station =1.15*c;
}

//SEAMOS VTUAV
else
{
            c= le/2;
            body_station = c;
            rotor_station = 0.95*c;
            eng_station = 0.96*c;
            propulsionsubsys_station = 0.97*c;
            nacelle_station = 0.98*c;
            fuel_station = 0.88*c;
            drysystem_station = 1.20*c;
            electricalsys_station =1.15*c;
}

m_Bb.Format("%g", body_station);
m_Rb.Format("%g", rotor_station);
m_Db.Format("%g", drysystem_station);
m_Mb.Format("%g", eng_station);
m_Pb.Format("%g", propulsionsubsys_station);
m_Nb.Format("%g", nacelle_station);
m_Fb.Format("%g", fuel_station);
m_Eb.Format("%g", electricalsys_station);

//calculate moment
```

```
m_Bc.Format("%g", body_weight * body_station);
m_Rc.Format("%g", rotor_weight * rotor_station);
m_Dc.Format("%g", drysystem_weight * drysystem_station);
m_Mc.Format("%g", eng_weight * eng_station);
m_Pc.Format("%g", propulsionsubsys_weight * propulsionsubsys_station);
m_Nc.Format("%g", nacelle_weight * nacelle_station);
m_Fc.Format("%g", fuel_weight * fuel_station);
m_Ec.Format("%g", electricalsys_weight * electricalsys_station);


total_moment=(body_weight * body_station) + (rotor_weight * rotor_station) +
(drysystem_weight * drysystem_station) + (eng_weight * eng_station) +
(propulsionsubsys_weight * propulsionsubsys_station) + (nacelle_weight * nacelle_station) +
(fuel_weight * fuel_station) + (electricalsys_weight * electricalsys_station);
total_weight=(body_weight) + (rotor_weight) + (drysystem_weight) + (eng_weight) +
(propulsionsubsys_weight) + (nacelle_weight) + (fuel_weight) + (electricalsys_weight);

m_M1.Format("%g",total_moment);
m_W1.Format("%g",total_weight);
m_CG1.Format("%g",total_moment/total_weight);
m_V1.Format("%g",fabs(  ((total_moment/total_weight)-(c)) * 100 / c ));

//C.G with payload
m_san.Format("%d",i1);
m_saw.Format("%g",w1);
m_sas.Format("%g",1.10*c);
m_sam.Format("%g", w1 * (1.10*c));
m_trn.Format("%d",i2);
m_trw.Format("%g",w2);
m_trs.Format("%g",1.30*c);
m_trm.Format("%g", w2 * (1.30*c));
m_trnn.Format("%d",i3);
m_trnw.Format("%g",w3);
m_trns.Format("%g",1.25*c);
m_trnm.Format("%g", w3 * (1.25*c));
m_ten.Format("%d",i4);
m_tew.Format("%g",w4);
m_tes.Format("%g",1.20*c);
m_tem.Format("%g", w4 * (1.20*c));
m_ren.Format("%d",i5);
m_rew.Format("%g",w5);
m_res.Format("%g",1.15*c);
m_rem.Format("%g", w5 * (1.15*c));
m_inn.Format("%d",i6);
m_inw.Format("%g",w6);
m_ins.Format("%g",1.03*c);
m_inm.Format("%g", w6 * (1.03*c));
m_ipn.Format("%d",i7);
m_ipw.Format("%g",w7);
m_ips.Format("%g",0.99*c);
m_ipm.Format("%g", w7 * (0.99*c));
m_oun.Format("%d",i8);
m_ouw.Format("%g",w8);
m_ous.Format("%g",0.94*c);
m_oum.Format("%g", w8 * (0.94*c));
m_csn.Format("%d",i9);
m_csw.Format("%g",w9);
m_css.Format("%g",0.88*c);
m_csm.Format("%g", w9 * (0.88*c));
m_mpn.Format("%d",i10);
```

```
		m_mpw.Format("%g",w10);
		m_mps.Format("%g",1.22*c);
		m_mpm.Format("%g", w10 * (1.22*c));
		m_rsn.Format("%d",i11);
		m_rsw.Format("%g",w11);
		m_rss.Format("%g",0.85*c);
		m_rsm.Format("%g", w11 * (0.85*c));
		m_rjn.Format("%d",i12);
		m_rjw.Format("%g",w12);
		m_rjs.Format("%g",c);
		m_rjm.Format("%g", w12 * (c));
		m_fln.Format("%d",i13);
		m_flw.Format("%g",w13);
		m_fls.Format("%g",1.05*c);
		m_flm.Format("%g", w13 * (1.05*c));
		m_ran.Format("%d",i14);
		m_raw.Format("%g",w14);
		m_ras.Format("%g",c);
		m_ram.Format("%g", w14 * (c));

		//finally
		double payload_moment, payload_weight, final_moment, final_weight;
		payload_moment= (w1 * (1.10*c)) + (w2 * (1.30*c)) + (w3 * (1.25*c)) + (w4 * (1.20*c)) +
		(w5 * (1.15*c)) + (w6 * (1.03*c)) + (w7 * (0.99*c)) + (w8 * (0.94*c)) + (w9 * (0.88*c))
		+(w10 * (1.22*c)) +(w11 * (0.85*c)) +(w12 * (c)) +(w13 * (1.05*c)) +(w14 * (c))  ;
		payload_weight= w1+w2+w3+w4+w5+w6+w7+w8+w9+w10+w11+w12+w13+w14;
		final_moment=payload_moment + total_moment ;
		final_weight=payload_weight + total_weight ;
		m_M2.Format("%g",payload_moment);
		m_W2.Format("%g",payload_weight);
		m_M3.Format("%g",final_moment);
		m_W3.Format("%g",final_weight);
		m_CG2.Format("%g",final_moment/final_weight);
		m_V2.Format("%g",fabs(  ((final_moment/final_weight)-(c)) * 100 / c ));

		//show the graph
		m_pic1.ShowWindow(SW_SHOW);
		UpdateData(FALSE);
		return TRUE;
}

void C7_6Dlg::On3b()
{
		EndDialog(IDCANCEL);
		C7_5Dlg a;
		a.DoModal();
}

void C7_6Dlg::OnRadio1()
{
		m_pic1.ShowWindow(SW_SHOW);
		m_pic2.ShowWindow(SW_HIDE);
}


void C7_6Dlg::OnRadio2()
{
		m_pic2.ShowWindow(SW_SHOW);
		m_pic1.ShowWindow(SW_HIDE);
}
```

```
void C7_6Dlg::OnRadio3()
{
	m_pic2.ShowWindow(SW_HIDE);
	m_pic1.ShowWindow(SW_HIDE);

	//Generate the profile
	//Method 2 -  Dont have to capture and release DC - Device Context
	CClientDC dlgDC(this);
	CRect rc;
	GetClientRect(&rc);
	CPen penBlue(PS_DASHDOTDOT, 1, RGB(0,0,255));
	CPen *pOldPen = NULL;
	pOldPen = dlgDC.SelectObject(&penBlue);
	CBrush brGrey(RGB(128,128,128));
	CBrush *pOldBrush=NULL;
	pOldBrush=dlgDC.SelectObject(&brGrey);

	if(wsn>0)
	{
		dlgDC.MoveTo(410+170,570);
		dlgDC.LineTo(655+170,670);
		dlgDC.LineTo(755+170,670);
		dlgDC.LineTo(755+170,470);
		dlgDC.LineTo(655+170,470);
		dlgDC.LineTo(410+170,570);
		dlgDC.LineTo(430+170,530);
		dlgDC.LineTo(675+170,633);
		dlgDC.LineTo(655+170,670);
		dlgDC.LineTo(675+170,633);
		dlgDC.LineTo(775+170,633);
		dlgDC.LineTo(755+170,670);
		dlgDC.LineTo(775+170,633);
		dlgDC.LineTo(775+170,433);
		dlgDC.LineTo(755+170,470);
		dlgDC.LineTo(775+170,433);
		dlgDC.LineTo(675+170,433);
		dlgDC.LineTo(655+170,470);
		dlgDC.LineTo(675+170,433);
		dlgDC.LineTo(675+170,633);
		dlgDC.LineTo(675+170,433);
		dlgDC.LineTo(430+170,530);
		dlgDC.Chord(402+170,390,753+170,683,382+170,575,687+170,720);
		dlgDC.AngleArc(609+170,633,206,74,80);
		dlgDC.TextOut(660+170,530,"cg");
		dlgDC.TextOut(560,690,"LAYOUT OF COUNTER ROTATING VTUAV W/O
		ROTOR SYSTEM");

		dlgDC.Ellipse(410+85-20,485-40,410+85+30-20,515-40);
		dlgDC.MoveTo(410+85-20,570-40);
		CRect a;
		a.TopLeft().x=410+85-20;
		a.TopLeft().y=600-40;
		a.BottomRight().x=410+85+30-20;
		a.BottomRight().y=500-40;
		dlgDC.Draw3dRect(a,0x00FF1122,0x00FFFFFF);
		//first rotor
		CPoint x1,y1,z1;
		x1.x=510-20;
		x1.y=445;
```

```
y1.x=615-20;
y1.y=385;
z1.x=595-20;
z1.y=365;

CPoint point[]={x1,y1,z1};
dlgDC.Polygon(point,3);
x1.x=510-20;
x1.y=445;
y1.x=435-20;
y1.y=525;
z1.x=415-20;
z1.y=505;
CPoint point1[]={x1,y1,z1};
dlgDC.Polygon(point1,3);
x1.x=510-20;
x1.y=445;
y1.x=435-20;
y1.y=360;
z1.x=415-20;
z1.y=380;
CPoint point2[]={x1,y1,z1};
dlgDC.Polygon(point2,3);
x1.x=510-20;
x1.y=445;
y1.x=615-20;
y1.y=505;
z1.x=595-20;
z1.y=525;
CPoint point3[]={x1,y1,z1};
dlgDC.Polygon(point3,3);

//2nd rotor
x1.x=510-20;
x1.y=445+30;
y1.x=615-20;
y1.y=385+30;
z1.x=595-20;
z1.y=365+30;

CPoint point4[]={x1,y1,z1};
dlgDC.Polygon(point4,3);
x1.x=510-20;
x1.y=445+30;
y1.x=435-20;
y1.y=525+30;
z1.x=415-20;
z1.y=505+30;

CPoint point5[]={x1,y1,z1};
dlgDC.Polygon(point5,3);
x1.x=510-20;
x1.y=445+30;
y1.x=435-20;
y1.y=360+30;
z1.x=415-20;
z1.y=380+30;

CPoint point6[]={x1,y1,z1};
dlgDC.Polygon(point6,3);
```

```
		x1.x=510-20;
		x1.y=445+30;
		y1.x=615-20;
		y1.y=505+30;
		z1.x=595-20;
		z1.y=525+30;

		CPoint point7[]={x1,y1,z1};
		dlgDC.Polygon(point7,3);
		dlgDC.TextOut(405,590,"COUNTER ROTATING");
		dlgDC.TextOut(415,610,"ROTOR SYSTEM");
}

else
{
		dlgDC.MoveTo(695,570);
		dlgDC.AngleArc(695,570,80,90,180);
		dlgDC.LineTo(695,490);
		dlgDC.LineTo(855,490);
		dlgDC.LineTo(855,650);
		dlgDC.LineTo(695,650);
		dlgDC.MoveTo(855,570);
		dlgDC.AngleArc(855,570,80,270,180);

		dlgDC.MoveTo(695+10,570-25);
		dlgDC.AngleArc(695+10,570-25,80,90,180);
		dlgDC.LineTo(695+10,490-25);
		dlgDC.LineTo(855+10,490-25);
		dlgDC.LineTo(855+10,650-25);
		dlgDC.LineTo(695+10,650-25);
		dlgDC.MoveTo(855+10,570-25);
		dlgDC.AngleArc(855+10,570-25,80,270,180);

		dlgDC.MoveTo(695,490);
		dlgDC.LineTo(695+10,490-25);
		dlgDC.MoveTo(855,490);
		dlgDC.LineTo(855+10,490-25);
		dlgDC.MoveTo(855,650);
		dlgDC.LineTo(855+10,650-25);
		dlgDC.MoveTo(695,650);
		dlgDC.LineTo(695+10,650-25);

		dlgDC.TextOut(560,690,"LAYOUT OF COUNTER ROTATING VTUAV W/O
		ROTOR SYSTEM");

		dlgDC.Ellipse(410+85-20,485-40,410+85+30-20,515-40);
		dlgDC.MoveTo(410+85-20,570-40);
		CRect a;
		a.TopLeft().x=410+85-20;
		a.TopLeft().y=600-40;
		a.BottomRight().x=410+85+30-20;
		a.BottomRight().y=500-40;

		dlgDC.Draw3dRect(a,0x00FF1122,0x00FFFFFF);

		//first rotor
		CPoint x1,y1,z1;
		x1.x=510-20;
		x1.y=445;
		y1.x=615-20;
```

```
y1.y=385;
z1.x=595-20;
z1.y=365;

CPoint point[]={x1,y1,z1};
dlgDC.Polygon(point,3);

x1.x=510-20;
x1.y=445;
y1.x=435-20;
y1.y=525;
z1.x=415-20;
z1.y=505;

CPoint point1[]={x1,y1,z1};
dlgDC.Polygon(point1,3);
x1.x=510-20;
x1.y=445;
y1.x=435-20;
y1.y=360;
z1.x=415-20;
z1.y=380;

CPoint point2[]={x1,y1,z1};
dlgDC.Polygon(point2,3);
x1.x=510-20;
x1.y=445;
y1.x=615-20;
y1.y=505;
z1.x=595-20;
z1.y=525;

CPoint point3[]={x1,y1,z1};
dlgDC.Polygon(point3,3);

//2nd rotor
x1.x=510-20;
x1.y=445+30;
y1.x=615-20;
y1.y=385+30;
z1.x=595-20;
z1.y=365+30;

CPoint point4[]={x1,y1,z1};
dlgDC.Polygon(point4,3);

x1.x=510-20;
x1.y=445+30;
y1.x=435-20;
y1.y=525+30;
z1.x=415-20;
z1.y=505+30;

CPoint point5[]={x1,y1,z1};
dlgDC.Polygon(point5,3);

x1.x=510-20;
x1.y=445+30;
y1.x=435-20;
y1.y=360+30;
```

```
		z1.x=415-20;
		z1.y=380+30;

		CPoint point6[]={x1,y1,z1};
		dlgDC.Polygon(point6,3);
		x1.x=510-20;
		x1.y=445+30;
		y1.x=615-20;
		y1.y=505+30;
		z1.x=595-20;
		z1.y=525+30;

		CPoint point7[]={x1,y1,z1};
		dlgDC.Polygon(point7,3);

		dlgDC.TextOut(405,590,"COUNTER ROTATING");
		dlgDC.TextOut(415,610,"ROTOR SYSTEM");
	}
}

void C7_6Dlg::On7f()
{
	UpdateData(TRUE);

	char  *stopstring;//pointer to end of number in a string
	m_sam.Format("%g", w1*strtod(m_sas,&stopstring));
	m_trm.Format("%g", w2*strtod(m_trs,&stopstring));
	m_trnm.Format("%g", w3*strtod(m_trns,&stopstring));
	m_tem.Format("%g", w4*strtod(m_tes,&stopstring));
	m_rem.Format("%g", w5*strtod(m_res,&stopstring));
	m_inm.Format("%g", w6*strtod(m_ins,&stopstring));
	m_ipm.Format("%g", w7*strtod(m_ips,&stopstring));
	m_oum.Format("%g", w8*strtod(m_ous,&stopstring));
	m_csm.Format("%g", w9*strtod(m_css,&stopstring));
	m_mpm.Format("%g", w10*strtod(m_mps,&stopstring));
	m_rsm.Format("%g", w11*strtod(m_rss,&stopstring));
	m_rjm.Format("%g", w12*strtod(m_rjs,&stopstring));
	m_flm.Format("%g", w13*strtod(m_fls,&stopstring));
	m_ram.Format("%g", w14*strtod(m_ras,&stopstring));

	// finally
	double payload_moment, payload_weight, final_moment, final_weight;
	payload_moment = (w1 * strtod(m_sas,&stopstring)) + (w2 * strtod(m_trs,&stopstring))
		+ (w3 * strtod(m_trns,&stopstring)) + (w4 * strtod(m_tes,&stopstring))
		+ (w5 * strtod(m_res,&stopstring)) + (w6 * strtod(m_ins,&stopstring))
		+ (w7 * strtod(m_ips,&stopstring)) + (w8 * strtod(m_ous,&stopstring))
		+ (w9 * strtod(m_css,&stopstring)) + (w10 * strtod(m_mps,&stopstring))
		+ (w11 * strtod(m_rss,&stopstring)) + (w12 * strtod(m_rjs,&stopstring))
		+(w13 * strtod(m_fls,&stopstring)) + (w14 * strtod(m_ras,&stopstring));

	payload_weight= w1+w2+w3+w4+w5+w6+w7+w8+w9+w10+w11+w12+w13+w14;
	final_moment=payload_moment + strtod(m_M1,&stopstring) ;
	final_weight=payload_weight + strtod(m_W1,&stopstring) ;
	m_M2.Format("%g",payload_moment);
	m_W2.Format("%g",payload_weight);
	m_M3.Format("%g",final_moment);
	m_W3.Format("%g",final_weight);
	m_CG2.Format("%g",final_moment/final_weight);
	m_V2.Format("%g",fabs(  ((final_moment/final_weight)-(c)) * 100 / c ));
	UpdateData(FALSE);
```

```
	if(fabs(((final_moment/final_weight)-(c)) * 100 / c ) > 7)
	MessageBox("Since the variation is >7%, hence the solution may not be acceptable.");
}

void C7_6Dlg::OnRadio4()
{
	C7_7Dlg a;
	a.DoModal();
}
```

J.10 'Non-editable database' and 'Editable database' classes

The 'Non-editable database' and 'Editable database' classes provide read-only/update facilities of the design parameters of various VTUAVs in-development/in-service and their power-plants as follows:

```
// BviDlg.cpp : implementation file
// Declaration of header files
#include "stdafx.h"
#include "VTUAV_Expert_System.h"
#include "BviDlg.h"
#include "column.h"
#include "columns.h"
#include "datagrid.h"
#include "_recordset.h"
#include "FindBVIDlg.h"
#include "Find1BVIDlg.h"
#include "Find2BVIDlg.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

int return1;
int return2;

// Declaration & initialisation of class variables
CBviDlg::CBviDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CBviDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CBviDlg)
	m_editva = _T("");
	m_editva1 = _T("");
	m_e5str = _T("");
	m_editva2 = _T("");
	m_e4str = _T("");
	//}}AFX_DATA_INIT
}

// Limiting variables values
void CBviDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CBviDlg)
```

```
	DDX_Control(pDX, IDC_EDIT3, m_3c);
	DDX_Control(pDX, IDC_EDIT5, m_e5c);
	DDX_Control(pDX, IDC_BUTTON1, m_ltmt);
	DDX_Control(pDX, IDC_EDIT4, m_c);
	DDX_Control(pDX, IDC_BUTTONFIND, m_find);
	DDX_Control(pDX, IDC_EDIT1, m_editco);
	DDX_Control(pDX, IDC_ADOVTUAVMAIN, m_adovtmain);
	DDX_Control(pDX, IDC_DATAGRID1, m_dg);
	DDX_Text(pDX, IDC_EDIT1, m_editva);
	DDX_Text(pDX, IDC_EDIT2, m_editva1);
	DDX_Text(pDX, IDC_EDIT5, m_e5str);
	DDX_Text(pDX, IDC_EDIT3, m_editva2);
	DDX_Text(pDX, IDC_EDIT4, m_e4str);
	//}}AFX_DATA_MAP
}

// Defining user-actions vis-a-vis system messages
BEGIN_MESSAGE_MAP(CBviDlg, CDialog)
	//{{AFX_MSG_MAP(CBviDlg)
	ON_BN_CLICKED(IDC_BUTTONFIND, OnButtonfind)
	ON_EN_CHANGE(IDC_EDIT4, OnChangeEdit4)
	ON_BN_CLICKED(IDC_BUTTON1, OnButton1)
	ON_EN_CHANGE(IDC_EDIT5, OnChangeEdit5)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

BEGIN_EVENTSINK_MAP(CBviDlg, CDialog)
	//{{AFX_EVENTSINK_MAP(CBviDlg)
	ON_EVENT(CBviDlg, IDC_DATAGRID1, 218 /* RowColChange */,
	OnRowColChangeDatagrid1, VTS_PVARIANT VTS_I2)
	//}}AFX_EVENTSINK_MAP
END_EVENTSINK_MAP()

// Class method – Initialisation of dialog
BOOL CBviDlg::OnInitDialog()
{
	CDialog::OnInitDialog();
	m_adovtmain.GetRecordset().MoveLast();
	m_adovtmain.GetRecordset().MoveFirst();
	m_find.SetWindowText("Find");

	CEdit* ed = (CEdit*)GetDlgItem(IDC_EDIT1);
	CEdit*ed1 = (CEdit*)GetDlgItem(IDC_EDIT2);
	long num1 =  m_dg.GetColumns().GetCount();

	if(num1<10)
	{
		ed->SetWindowText(CString(num1+48));
	}

	else
	{
		int quo = num1/10;
		int rem = num1%10;
		ed->SetWindowText(CString(quo+48) + CString(rem+48));
	}
		long num2 = m_adovtmain.GetRecordset().GetRecordCount();
	if (num2<10)
	{
		ed1->SetWindowText(CString(num2+48));
```

```
}


else if(num2<100)
{
	int quo = num2/10;
	int rem = num2%10;
	ed1->SetWindowText(CString(quo+48) + CString(rem+48));
}

else if(num2<1000)
{
	int quo = num2/10;
	int quo1 = quo/10;
	int quo2 = quo%10;
	int rem = num2%10;
	ed1->SetWindowText(CString(quo1+48) + CString(quo2+48) + CString(rem+48));
}

else
{
	int quo = num2/100;
	int quo1 = quo/10;
	int quo2 = quo%10;
	int quoa = num2%100;
	int quo3 = quoa/10;
	int rem = num2%10;
	ed1->SetWindowText(CString(quo1+48) + CString(quo2+48) + CString(quo3+48) +
	CString(rem+48));
}

CEdit*ed2 = (CEdit*)GetDlgItem(IDC_EDIT3);
long num3 = m_adovtmain.GetRecordset().GetAbsolutePosition();

if (num3<10)
{
	ed2->SetWindowText(CString(num3+48));
}
else if(num3<100)
{
	int quo = num3/10;
	int rem = num3%10;
	ed2->SetWindowText(CString(quo+48) + CString(rem+48));
}
else if(num3<1000)
{
	int quo = num3/10;
	int quo1 = quo/10;
	int quo2 = quo%10;
	int rem = num3%10;
	ed2->SetWindowText(CString(quo1+48) + CString(quo2+48) + CString(rem+48));
}
else
{
	int quo = num3/100;
	int quo1 = quo/10;
	int quo2 = quo%10;
	int quoa = num3%100;
	int quo3 = quoa/10;
	int rem = num3%10;
```

```
                ed2->SetWindowText(CString(quo1+48) + CString(quo2+48) + CString(quo3+48) +
                CString(rem+48));

        }

        //set the properties
        m_dg.SetAllowUpdate(FALSE);
        m_dg.SetAllowAddNew(FALSE);
        m_dg.SetAllowDelete(FALSE);

        extern int modify_check;
        //add new
        if(modify_check == 1)
        {
                m_dg.SetAllowUpdate(TRUE); //this is essential to set the add new property
                m_dg.SetAllowAddNew(TRUE);
                m_adovtmain.GetRecordset().SetAbsolutePosition(m_adovtmain.GetRecordset().Get
                RecordCount());
                UpdateData(FALSE);
        }

        //update
        if(modify_check == 2)
        {
                m_dg.SetAllowUpdate(TRUE); //only update the existing record values
        }

        //delete
        if(modify_check == 3)
        {
                m_dg.SetAllowDelete(TRUE); //delete the record
        }
                return TRUE;  // return TRUE unless you set the focus to a control

}

void CBviDlg::OnRowColChangeDatagrid1(VARIANT FAR* LastRow, short LastCol)
{
        CEdit*ed2 = (CEdit*)GetDlgItem(IDC_EDIT3);//ed2 should have been named as edrow
        long num3 = m_adovtmain.GetRecordset().GetAbsolutePosition();
        if (num3<10)
        {
                ed2->SetWindowText(CString(num3+48));
        }

        else if(num3<100)
        {
                int quo = num3/10;
                int rem = num3%10;
                ed2->SetWindowText(CString(quo+48) + CString(rem+48));
        }

        else if(num3<1000)
        {
                int quo = num3/10;
                int quo1 = quo/10;
                int quo2 = quo%10;
                int rem = num3%10;
                ed2->SetWindowText(CString(quo1+48) + CString(quo2+48) + CString(rem+48));
        }
```

```
	else
	{
		int quo = num3/100;
		int quo1 = quo/10;
		int quo2 = quo%10;
		int quoa = num3%100;
		int quo3 = quoa/10;
		int rem = num3%10;
		ed2->SetWindowText(CString(quo1+48) + CString(quo2+48) + CString(quo3+48) +
		CString(rem+48));
	}

}
```

// Class method – Search function

```
void CBviDlg::OnButtonfind()
{
	m_e5c.ShowWindow(SW_HIDE);
	m_c.ShowWindow(SW_HIDE);
	m_ltmt.ShowWindow(SW_SHOW);
	m_find.EnableWindow(FALSE);
	CFindBVIDlg a;
	a.DoModal();

	if(a.result_findbvidlg  ==  1)
	{
		m_e5c.ShowWindow(SW_SHOW);
		Find1BVIDlg a;
		if (a.DoModal() == IDOK)
		return1 = 1;
		else return1 = 0;
	}

	if(a.result_findbvidlg == 2)
	{
		m_c.ShowWindow(SW_SHOW);
		CFind2BVIDlg a;
		if (a.DoModal() == IDOK){
		return2 = 1;
	 }

	else
	{
		return2 = 0;
	}
}

void CBviDlg::OnChangeEdit4()
{
	m_find.EnableWindow(FALSE);
	UpdateData(TRUE);
	CString x ;
	CString y;
	y = m_e4str;

	if(return2 == 1)
	{
		x = "SELECT TOP 18 * FROM All_UAV WHERE MISSION Like '";
		x+=y;
```

```
		x+="%'";
	}


	else
	{
		x = "SELECT TOP 18 * FROM All_UAV WHERE MISSION = '";
		x+=y;
		x+="'";
	}

	m_adovtmain.SetRecordSource(x);
	m_adovtmain.Refresh();
}

void CBviDlg::OnButton1()
{
	m_find.EnableWindow(TRUE);
	m_find.SetWindowText("Find");
	m_e5c.ShowWindow(SW_HIDE);
	m_c.ShowWindow(SW_HIDE);
	CString x = "SELECT * FROM All_UAV";
	m_adovtmain.SetRecordSource(x);
	m_adovtmain.Refresh();
	m_c.ShowWindow(SW_HIDE);
 }


void CBviDlg::OnChangeEdit5()
{
	m_find.EnableWindow(FALSE);
	UpdateData(TRUE);
	CString str = m_e5str;
	CString x = " SELECT TOP 18 * FROM All_UAV WHERE NAME ";
	CFind2BVIDlg b;

	if (return1 == 1)
	{
		x+= "Like '";
		x+= str;
		x+= "%'";
	}

	else
	{
		x+= "='";
		x+=str;
		x+= "'";
	}

	m_adovtmain.SetRecordSource(x);
	m_adovtmain.Refresh();
	m_editva.Format("%ld",m_dg.GetColumns().GetCount());
	m_editva1.Format("%ld",m_adovtmain.GetRecordset().GetRecordCount());

	if(m_adovtmain.GetRecordset().GetAbsolutePosition()!= -1)
	m_editva2.Format("%ld",m_adovtmain.GetRecordset().GetAbsolutePosition());

	else
	m_editva2.Format("%ld",m_adovtmain.GetRecordset().GetAbsolutePosition()+1);
```

```
	UpdateData(FALSE);
}
void CBviDlg::OnOK()
{
	int pnum=1;
	CAdodc* ad =(CAdodc*)GetDlgItem(IDC_ADOVTUAVMAIN);
	VARIANT a;
	a = ad->GetRecordset().GetBookmark();
	a.intVal = ad->GetRecordset().GetAbsolutePosition();

	CDataList* dl1 =(CDataList*)GetDlgItem(IDC_DATALISTID1);
	CDataList* dl2 =(CDataList*)GetDlgItem(IDC_DATALISTID2);
	CDataList* dl3 =(CDataList*)GetDlgItem(IDC_DATALISTID3);
	CDataList* dl4 =(CDataList*)GetDlgItem(IDC_DATALISTID4);
	CDataList* dl5 =(CDataList*)GetDlgItem(IDC_DATALISTID5);
	CDataList* dl6 =(CDataList*)GetDlgItem(IDC_DATALISTID6);
	CDataList* dl7 =(CDataList*)GetDlgItem(IDC_DATALISTID7);
	CDataList* dl8 =(CDataList*)GetDlgItem(IDC_DATALISTID8);
	CDataList* dl9 =(CDataList*)GetDlgItem(IDC_DATALISTID9);
	CDataList* dl10 =(CDataList*)GetDlgItem(IDC_DATALISTID10);
	CDataList* dl11 =(CDataList*)GetDlgItem(IDC_DATALISTID11);

	//1. Construct a CPrintDialog object
	CPrintDialog dlgPrint(FALSE,PD_NOSELECTION,this);
	//PD_ALLPAGES flag was used by me at first instance
	MessageBox("Please select landscape orientation and A4 paper size. Elsewise, print will not
	work correctly","Compulsory Printer Settings");

	//dlgPrint.m_pd.Flags
	//2. The printdialog has to be initiated and displayed
	if(dlgPrint.DoModal()==IDOK)
	{
		//3. Attach the printer DC from the dialog to a CDC object
		CDC dcPrint;
		dcPrint.Attach(dlgPrint.GetPrinterDC());

		//4. Create and Fill a DOCINFO structure
		DOCINFO myPrintJob;
		myPrintJob.cbSize = sizeof(myPrintJob);
		myPrintJob.lpszDocName = "UAV_Design_PrintJob";
		myPrintJob.lpszOutput = NULL;
		myPrintJob.lpszDatatype = NULL;
		myPrintJob.fwType = NULL;

		 //5. Start the printing document. StartDoc() function tells window to start spooling
		and EndDoc tells it that the spooling is done
		if(dcPrint.StartDoc(&myPrintJob)>=0)
		//Document is checking if their is sufficient memory/diskspace to proceed or not.
		{
			//5.1 Start the page
			dcPrint.StartPage();

			//5.2 Start the drawing
			dcPrint.TextOut(3000,30,"UAV DESIGN ASSIST DATASHEET");
			dcPrint.TextOut(0,100,"ID");//dl1->GetBoundColumn());
			dcPrint.TextOut(300,100,"Name");//dl1->GetBoundColumn());
			dcPrint.TextOut(2100,100,"Length"); //dl2->GetBoundColumn());
			dcPrint.TextOut(2600,100,"Wing");//dl3->GetBoundColumn());
			dcPrint.TextOut(3000,100,"Rotor");//dl4->GetBoundColumn());
			dcPrint.TextOut(3500,100,"Body");//dl5->GetBoundColumn());
```

```
dcPrint.TextOut(3900,100,"Wt.");//dl6->GetBoundColumn());
dcPrint.TextOut(4300,100,"Power.");//dl7->GetBoundColumn());
dcPrint.TextOut(4900,100,"Endur.");//dl8->GetBoundColumn());
dcPrint.TextOut(5400,100,"Range");//dl9->GetBoundColumn());
dcPrint.TextOut(5900,100,"Altitude");//dl10->GetBoundColumn());
dcPrint.TextOut(0,200,"");//dl1->GetBoundColumn());
dcPrint.TextOut(300,200,"");//dl1->GetBoundColumn());
dcPrint.TextOut(2100,200,"(ft)"); //dl2->GetBoundColumn());
dcPrint.TextOut(2600,200,"(ft)");//dl3->GetBoundColumn());
dcPrint.TextOut(3000,200,"(ft)");//dl4->GetBoundColumn());
dcPrint.TextOut(3500,200,"(ft)");//dl5->GetBoundColumn());
dcPrint.TextOut(3900,200,"(lb)");//dl6->GetBoundColumn());
dcPrint.TextOut(4300,200,"(hp)");//dl7->GetBoundColumn());
dcPrint.TextOut(4900,200,"(hr)");//dl8->GetBoundColumn());
dcPrint.TextOut(5400,200,"(mi)");//dl9->GetBoundColumn());
dcPrint.TextOut(5900,200,"(ft)");//dl10->GetBoundColumn());

//use a bookmark
ad->GetRecordset().MoveFirst();

int y=400;
for(int i=0;i<ad->GetRecordset().GetRecordCount(); i++)
{
        CString str;
        str.Format("%ld",ad->GetRecordset().GetAbsolutePosition());
        dcPrint.TextOut(0,y,str);
        dcPrint.TextOut(300,y,dl1->GetBoundText());
        dcPrint.TextOut(2100,y,dl2->GetText());// >GetBoundText());
        dcPrint.TextOut(2600,y,dl3->GetText());// >GetBoundText());
        dcPrint.TextOut(3000,y,dl4->GetText());// >GetBoundText());
        dcPrint.TextOut(3500,y,dl5->GetText());// >GetBoundText());
        dcPrint.TextOut(3900,y,dl6->GetText());// >GetBoundText());
        dcPrint.TextOut(4300,y,dl7->GetText());// >GetBoundText());
        dcPrint.TextOut(4900,y,dl8->GetText());// >GetBoundText());
        dcPrint.TextOut(5400,y,dl9->GetText());// >GetBoundText());
        dcPrint.TextOut(5900,y,dl10->GetText());// >GetBoundText())

        ad->GetRecordset().MoveNext();
        y+=175;

        if((y/4300)>0)
        {
                CString pagem,page;
                page="Page ";
                pagem.Format("%d",pnum++);
                page+=pagem;
                dcPrint.TextOut(3000,4400,page);
                dcPrint.EndPage();
                dcPrint.StartPage();
                dcPrint.TextOut(0,100,"ID");
                dcPrint.TextOut(300,100,"Name");Column());
                dcPrint.TextOut(2100,100,"Length");
                dcPrint.TextOut(2600,100,"Wing");
                dcPrint.TextOut(3000,100,"Rotor");
                dcPrint.TextOut(3500,100,"Body");
                dcPrint.TextOut(3900,100,"Wt.");
                dcPrint.TextOut(4300,100,"Power");
                dcPrint.TextOut(4900,100,"Endur.");
                dcPrint.TextOut(5400,100,"Range");
                dcPrint.TextOut(5900,100,"Altitude");
```

```
                    dcPrint.TextOut(0,200,"");
                    dcPrint.TextOut(300,200,"");
                    dcPrint.TextOut(2100,200,"(ft)");
                    dcPrint.TextOut(2600,200,"(ft)");
                    dcPrint.TextOut(3000,200,"(ft)");
                    dcPrint.TextOut(3500,200,"(ft)");
                    dcPrint.TextOut(3900,200,"(lb)");
                    dcPrint.TextOut(4300,200,"(hp)");
                    dcPrint.TextOut(4900,200,"(hr)");
                    dcPrint.TextOut(5400,200,"(mi)");
                    dcPrint.TextOut(5900,200,"(ft)");
                    y=400;
            }

}
ad->GetRecordset().SetAbsolutePosition(a.lVal);
 //Using Bookmarks.can also use (a.intval) no warnings will be generated
//ad->GetRecordset().MoveFirst();

if((y/4300) >0)
// This loop is for checking the space to print my warning below
{

            CString pagem,page;
            page="Page";
            pagem.Format("%d",++pnum);
            page+=pagem;
            dcPrint.TextOut(3500,4400, page);
            dcPrint.EndPage();
            dcPrint.StartPage();
            y=4450;
}

dcPrint.TextOut(10,y+80,"This Text has been Printed from the VTUAV
DESIGN ASSIST SOFTWARE's DATABASE COLLECTION. Software");
dcPrint.TextOut(10,y+160,"Manufacturer's and Distributors do not attest to
the accuracy of this information and thereby are not");
dcPrint.TextOut(10,y+240,"liable for any resulting damages/charges. ");

SYSTEMTIME a;
GetSystemTime(&a);
CString afinal,a1,a2,a3;
a3=".";
afinal="The Document has been printed on ";

if(a.wDayOfWeek==0)
            a1="Sunday";
if(a.wDayOfWeek==1)
            a1="Monday";
if(a.wDayOfWeek==2)
            a1="Tuesday";
if(a.wDayOfWeek==3)
            a1="Wednesday";
if(a.wDayOfWeek==4)
            a1="Thursday";
if(a.wDayOfWeek==5)
{a1="Friday"; a3=" Have A Great WeekEnd";}
if(a.wDayOfWeek==6)
a1="Saturday";
```

```
                        afinal+=a1;
                        afinal+=",The ";

                        a1="";
                        a1.Format("%d",a.wDay);//Word is a 16bit integer/equivalent to short int
                        if((a.wDay) >= 4 && (a.wDay)<= 20) a1+="th";
                        if((a.wDay) >= 24 && (a.wDay)<= 30)   a1+="th";
                        if(a.wDay == 1 || a.wDay == 21 || a.wDay == 31) a1+="st";
                        if(a.wDay == 2 || a.wDay == 22) a1+="nd";
                        if(a.wDay == 3 || a.wDay == 23) a1+="rd";

                        a1+="of";

                        if(a.wMonth == 1) a2="January";
                        if(a.wMonth == 2) a2="February";
                        if(a.wMonth == 3) a2="March";
                        if(a.wMonth == 4) a2="April";
                        if(a.wMonth == 5) a2="May";
                        if(a.wMonth == 6) a2="June";
                        if(a.wMonth == 7) a2="July";
                        if(a.wMonth == 8) a2="August";
                        if(a.wMonth == 9) a2="September";
                        if(a.wMonth == 10) a2="October";
                        if(a.wMonth == 11) a2="November";
                        if(a.wMonth == 12) a2="December";

                        //a2.Format("%d", a.wMonth);
                        a1+=a2;
                        a1+=",";
                        a2="";
                        a2.Format("%d", a.wYear);
                        a1+=a2;
                        a1+=a3;
                        afinal+=a1;
                        dcPrint.TextOut(10,y+320,afinal);

                        //5.3 Throw the page
                        dcPrint.EndPage();

                        //5.4 Close the document.
                        dcPrint.EndDoc();
                }

                        //6. Donot forget to delete printer devide context
                        dcPrint.DeleteDC();
        }

}
```

J.11 Print class

The print class involves print of database and design configuration as follows:

```
BOOL CVTUAV_Expert_SystemView::OnPreparePrinting(CPrintInfo* pInfo)
{
        // Default preparation
        pInfo->SetMinPage(1);
```

```
	pInfo->SetMaxPage(2);
	pInfo->m_pPD->m_pd.nCopies=1; // Default number of copies is 1

	 if(!pInfo->m_bPreview)
	AfxMessageBox("The Program is initiating your default Printer. For your information the
	default number of copies of each page is 1. The maximum number of pages are set to 2. Please
	manually override to change the settings. Also, you cannot print more than 5 copies of each
	page. This setting is non-negotiable.");


	do
	{
	 if(DoPreparePrinting(pInfo) == FALSE)
	 return FALSE;

	 //Warn user if too many copies are specified
	 if(pInfo->m_pPD->GetCopies() > 5)
	 MessageBox("Please choose less than 6 copies","Warning",MB_OK);

	}while(pInfo->m_pPD->GetCopies()>5);// Cant print more than 5 copies of the program

	 if(!pInfo->m_bPreview)
	AfxMessageBox("Please re-verify your printer settings. The selected Device is  " +pInfo-
	>m_pPD->GetDeviceName()+ " on  " +pInfo->m_pPD->GetPortName()+ " port and with  "
	+pInfo->m_pPD->GetDriverName()+ " driver.",  MB_OK);// == IDCANCEL)

	return DoPreparePrinting(pInfo);
}
```

J.12 'Mission categories' class

The class provides the option to select the desired category of mission requirements based on the operational needs as follows:

```
// MRDlg.cpp : implementation file
// Declaration of header files
#include "stdafx.h"
#include "VTUAV_Expert_System.h"
#include "MRDlg.h"
#include "MR1ADlg.h"
#include "MR1BDlg.h"
#include "MR1CDlg.h"
#include "MR1DDlg.h"
#include "MR1EDlg.h"
#include "InputDlg.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

CMRDlg::CMRDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CMRDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CMRDlg)
	//}}AFX_DATA_INIT
}
```

// Declaration & initialisation of class variables

```
void CMRDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CMRDlg)
	DDX_Control(pDX, IDMR5, m_mr5);
	DDX_Control(pDX, IDMR4, m_mr4);
	DDX_Control(pDX, IDMR3, m_mr3);
	DDX_Control(pDX, IDMR2, m_mr2);
	DDX_Control(pDX, IDMR1, m_mr1);
	//}}AFX_DATA_MAP
}
```

// Limiting variables values

```
BEGIN_MESSAGE_MAP(CMRDlg, CDialog)
	//{{AFX_MSG_MAP(CMRDlg)
	ON_BN_CLICKED(IDMR1, OnMr1)
	ON_BN_CLICKED(IDMR2, OnMr2)
	ON_BN_CLICKED(IDMR3, OnMr3)
	ON_BN_CLICKED(IDMR4, OnMr4)
	ON_BN_CLICKED(IDMR5, OnMr5)
	ON_BN_CLICKED(IDC_BUTTON1, OnButton1)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

int CMRDlg::DoModal()
{
	return CDialog::DoModal();
}

void CMRDlg::OnMr1()
{
	EndDialog(IDOK);
	CMR1ADlg a;
	a.DoModal();
}

void CMRDlg::OnMr2()
{
	EndDialog(IDOK);
	CMR1BDlg a;
	a.DoModal();
}

void CMRDlg::OnMr3()
{
	EndDialog(IDOK);
	CMR1CDlg a;
	a.DoModal();
}

void CMRDlg::OnMr4()
{
	EndDialog(IDOK);
	CMR1DDlg a;
	a.DoModal();
}

void CMRDlg::OnMr5()
```

```
{
	EndDialog(IDOK);
	CMR1EDlg a;
	a.DoModal();
}

void CMRDlg::OnButton1()
{
	EndDialog(IDOK);
	CInputDlg a;
	a.DoModal();
}
```

J.13 'Mission requirements' class

The class involves mission requirements based on the selected mission category. The class is programmed as follows as an illustration for 'Generic' mission category:

```
// MR1ADlg.cpp : implementation file
// Declaration of header files
#include "stdafx.h"
#include "VTUAV_Expert_System.h"
#include "MR1ADlg.h"
#include "MRDlg.h"
#include "MR1A1Dlg.h"
#include "MR1A2Dlg.h"
#include "MR1A3Dlg.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

CMR1ADlg::CMR1ADlg(CWnd* pParent /*=NULL*/)
	: CDialog(CMR1ADlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CMR1ADlg)
		// NOTE: the ClassWizard will add member initialization here
	//}}AFX_DATA_INIT
}


void CMR1ADlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CMR1ADlg)
		// NOTE: the ClassWizard will add DDX and DDV calls here
	//}}AFX_DATA_MAP
}

// Limiting variables values
BEGIN_MESSAGE_MAP(CMR1ADlg, CDialog)
	//{{AFX_MSG_MAP(CMR1ADlg)
	ON_BN_CLICKED(IDMR1A1, OnMr1a1)
	ON_BN_CLICKED(IDMR1A2, OnMr1a2)
	ON_BN_CLICKED(IDMR1A3, OnMr1a3)
	ON_BN_CLICKED(IDC_BUTTON2, OnButton2)
```

```
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

void CMR1ADlg::OnMr1a1()
{
	EndDialog(IDCANCEL);
	CMR1A1Dlg a;
	a.DoModal();
}

void CMR1ADlg::OnMr1a2()
{
	EndDialog(IDCANCEL);
	CMR1A2Dlg a;
	a.DoModal();
}

void CMR1ADlg::OnMr1a3()
{
	EndDialog(IDCANCEL);
	CMR1A3Dlg a;
	a.DoModal();
}

void CMR1ADlg::OnButton2()
{
	EndDialog(IDCANCEL);
	CMRDlg a;
	a.DoModal();
}

BEGIN_EVENTSINK_MAP(CMR1ADlg, CDialog)
	//{{AFX_EVENTSINK_MAP(CMR1ADlg)
	//}}AFX_EVENTSINK_MAP
END_EVENTSINK_MAP()
```

J.14 ‘Application’ class

The class responds to menu and taskbar interactions as follows:

```
// VTUAV_Expert_System.h : main header file for the VTUAV_Expert_System application
#if
!defined(AFX_VTUAV_Expert_System_H__9090EBC3_9518_11D8_A133_0004AC6E2E50__INCL
UDED_)
#define
AFX_VTUAV_Expert_System_H__9090EBC3_9518_11D8_A133_0004AC6E2E50__INCLUDED_
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
#ifndef __AFXWIN_H__
	#error include 'stdafx.h' before including this file for PCH
#endif

// Declaration of header files
#include "resource.h"       // main symbols
#include "VTUAV_Expert_System_i.h"
```

```
class CVTUAV_Expert_SystemApp : public CWinApp
{
public:
	CVTUAV_Expert_SystemApp();

	// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CVTUAV_Expert_SystemApp)
	public:
	virtual BOOL InitInstance();
	virtual BOOL InitApplication();
	virtual int ExitInstance();
	//}}AFX_VIRTUAL

	// Implementation
	//{{AFX_MSG(CVTUAV_Expert_SystemApp)
	afx_msg void OnAppAbout();
	afx_msg void OnDatabaseViewBvi();
	afx_msg void OnDatabaseViewDim();
	afx_msg void OnDatabaseViewOperst();
	afx_msg void OnDatabaseViewPerf();
	afx_msg void OnDatabaseViewPwrpt();
	afx_msg void OnDatabaseViewWt();
	afx_msg void OnMemoryUsage();
	afx_msg void OnWinappExplorer();
	afx_msg void OnWinappNotepad();
	afx_msg void OnWinappProgman();
	afx_msg void OnWinappWordpad();
	afx_msg void OnWinappCalculator();
	afx_msg void OnViewExplorer();
	afx_msg void OnFileFtp();
	afx_msg void OnViewFilemanager();
	afx_msg void OnWinappDrwatson();
	afx_msg void OnWinappRegistoryscanner();
	afx_msg void OnWinappSfc();
	afx_msg void OnDatabaseAddnewrecord();
	afx_msg void OnDatabaseUpdateexisitngrecord();
	afx_msg void OnDatabaseDeleteexisitingrecord();
	afx_msg void OnDesignprogramRun();
	afx_msg void OnDatabaseDisplayviewsEnginemodeclassificationElectricalengine();
	afx_msg void OnDatabaseDisplayviewsEnginemodeclassificationGliding();
	afx_msg void OnDatabaseDisplayviewsEnginemodeclassificationNelecengine();
	afx_msg void OnDatabaseEnginelistModify();
	afx_msg void OnDatabaseEnginelistView();
	afx_msg void OnIdsMemoryDevice();
	afx_msg void OnAppAboutHelp();
	afx_msg void OnHelpFinder();
	afx_msg void OnDesignprogramShadeoutthe3dprofile();
	afx_msg void OnDesignprogramDotoutthe3dprofile();
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()

private:
	BOOL m_bATLInited;
	BOOL InitATL();
	int ncount;
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.
```

```
#endif //
!defined(AFX_VTUAV_Expert_System_H__9090EBC3_9518_11D8_A133_0004AC6E2E50__INCL
UDED_)
```

J.15 ‘View’ class

The class responds to interactions involving input devices including mouse and keyboard as follows:

```
// VTUAV_Expert_SystemView.h : interface of the CVTUAV_Expert_SystemView class
#if
!defined(AFX_VTUAV_Expert_SystemVIEW_H__9090EBCB_9518_11D8_A133_0004AC6E2E50_
_INCLUDED_)
#define
AFX_VTUAV_Expert_SystemVIEW_H__9090EBCB_9518_11D8_A133_0004AC6E2E50__INCLU
DED_
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000


class CVTUAV_Expert_SystemView : public CView
{
        protected: // create from serialization only
        CVTUAV_Expert_SystemView();
        DECLARE_DYNCREATE(CVTUAV_Expert_SystemView)

// Attributes
public:
        CVTUAV_Expert_SystemDoc* GetDocument();

// Operations
public:

// Overrides
        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CVTUAV_Expert_SystemView)
        public:
        virtual void OnDraw(CDC* pDC);  // overridden to draw this view
        virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
        virtual void OnPrepareDC(CDC* pDC, CPrintInfo* pInfo = NULL);
        protected:
        virtual BOOL OnPreparePrinting(CPrintInfo* pInfo);
        virtual void OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo);
        virtual void OnEndPrinting(CDC* pDC, CPrintInfo* pInfo);
        virtual void OnActivateView(BOOL bActivate, CView* pActivateView, CView*
        pDeactiveView);
        virtual void OnPrint(CDC* pDC, CPrintInfo* pInfo);
        //}}AFX_VIRTUAL

// Implementation
public:
        virtual ~CVTUAV_Expert_SystemView();
        #ifdef _DEBUG
        virtual void AssertValid() const;
        virtual void Dump(CDumpContext& dc) const;
#endif
```

```
protected:
// Generated message map functions
protected:
        //{{AFX_MSG(CVTUAV_Expert_SystemView)
        afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
        afx_msg void OnMouseMove(UINT nFlags, CPoint point);
        afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
        afx_msg void OnRButtonDown(UINT nFlags, CPoint point);
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
private:
        CRect m_printrect;
        int py;
        int px;
};

#ifndef _DEBUG  // debug version in VTUAV_Expert_SystemView.cpp
inline CVTUAV_Expert_SystemDoc* CVTUAV_Expert_SystemView::GetDocument()
   { return (CVTUAV_Expert_SystemDoc*)m_pDocument; }
#endif

//{{AFX_INSERT_LOCATION}}
#endif //
!defined(AFX_VTUAV_Expert_SystemVIEW_H__9090EBCB_9518_11D8_A133_0004AC6E2E50_
_INCLUDED_)
```

I.16 'Document' class

The class involves database usage vis-à-vis the connection string as follows:

```
// VTUAV_Expert_SystemDoc.h : interface of the CVTUAV_Expert_SystemDoc class
#if
!defined(AFX_VTUAV_Expert_SystemDOC_H__9090EBC9_9518_11D8_A133_0004AC6E2E50__I
NCLUDED_)
#define
AFX_VTUAV_Expert_SystemDOC_H__9090EBC9_9518_11D8_A133_0004AC6E2E50__INCLUD
ED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

class CVTUAV_Expert_SystemDoc : public CDocument
{
        protected: // create from serialization only
        CVTUAV_Expert_SystemDoc();
        DECLARE_DYNCREATE(CVTUAV_Expert_SystemDoc)

// Attributes
public:

// Operations
public:

// Overrides
        // ClassWizard generated virtual function overrides
```

```
        //{{AFX_VIRTUAL(CVTUAV_Expert_SystemDoc)
        public:
        virtual BOOL OnNewDocument();
        virtual void Serialize(CArchive& ar);
        //}}AFX_VIRTUAL

// Implementation
public:
        virtual ~CVTUAV_Expert_SystemDoc();
        #ifdef _DEBUG
        virtual void AssertValid() const;
        virtual void Dump(CDumpContext& dc) const;
        #endif

protected:

protected:
        //{{AFX_MSG(CVTUAV_Expert_SystemDoc)
        // NOTE - the ClassWizard will add and remove member functions here.
        // DO NOT EDIT what you see in these blocks of generated code !
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif //
!defined(AFX_VTUAV_Expert_SystemDOC_H__9090EBC9_9518_11D8_A133_0004AC6E2E50__I
NCLUDED_)
```