Design of Air Conditioning Systems for Efficient Life-Cycle Operation using the ZEBRA Software Package.

P.G. MARSHALLSAY and R.E. LUXTON.
Department of Mechanical Engineering, The University of Adelaide, South Australia 5005.

Abstract.

Life-cycle design of air conditioning systems requires that plant should be capable of maintaining zone comfort conditions within specified limits, while minimising capital and operating costs. Lack of suitable design tools often prevents these objectives being met in conventional design practice. ZEBRA is a software package designed to simulate the performance of air conditioning plant across the whole range of operation. The structure of the package, which uses object-oriented methodologies, is described, as is its relationship to existing plant modelling codes. The programme's capabilities are illustrated by reference to a novel type of air conditioning system.

1. Introduction.

Design of a climate control system for a specified indoor environment is in essence an exercise in constrained optimisation. The designer seeks to minimise the cost of owning and operating the system while satisfying performance requirements. These will typically stipulate minimum ventilation levels and a tolerance band within which the indoor climate indices (usually dry-bulb temperature and relative humidity) must be maintained. Satisfactory system performance requires that these latter parameters be maintained within acceptable limits *across the entire operating cycle of the system*. Control of dry-bulb temperature presents no difficulties provided the plant has sufficient capacity to offset the sensible load during peak operating conditions. Control of relative humidity will generally be more difficult as part-load conditions typically pose a far more demanding test of the ability of the equipment to offset latent load than does the peak load condition. These factors clearly differentiate a system simulation code from one which merely simulates coil performance.

Evaluation of system performance using the tools available in most design offices is difficult, and much current design practice still relies upon the manual construction of process lines on a psychrometric chart. While this approach is capable of simulating the performance of a single-zone system reasonably accurately if sufficient rigour is applied, it is extremely labour-intensive. Further, it cannot readily be extended to the multizone case without making unverified assumptions concerning the manner in which the conditions in the various zones track one another. Given this situation, there is an understandable reluctance on the part of designers to explore more than a very minimal subset of the solutions possible for a given design problem, or to evaluate the performance of candidate solutions at other than the peak load conditions for which there is usually a contractual obligation. Occasionally one or two part-load conditions may be checked, but usually the part-load design problem is circumvented by relaxing the constraints on relative humidity, a fact which is reflected in the poor part-load performance of many air conditioning systems. Where contractual obligations require that the designer *guarantee* part-load dehumidification performance it is usual to overcool the supply air to the desired dew-point temperature, and then reheat to meet the target dry-bulb temperature, provided this strategy is permitted under the relevant building codes. However, the capital cost increases and the energy penalty can be huge.

The ability to simulate the behaviour of an air conditioning system across its entire operating envelope is obviously a key ingredient of any life-cycle design methodology. The ZEBRA software package has been designed to provide the desired functionality within a framework which takes a holistic approach to the entire specification and design process. The basis for the software package is a simulation engine which models the steady-state thermal performance of air conditioning systems in response to variations in imposed loads and ambient conditions. The model employed is based on a rigorous analysis of the psychrometric processes which occur within an air conditioning system. In essence, the procedure automates, generalises and refines the manual procedures currently employed by design offices. Similar rigour has been employed in the modelling of individual subsystems and equipment items, the intention being to provide a framework which can be augmented by incorporating experimental test data and manufacturers' performance data as appropriate. The required flexibility within the software has been obtained by structuring the package strictly according to object-oriented design principles. This provides a computational framework which offers maximum adaptability of configuration to accommodate a wide range of system geometries, and eases the integration of novel equipment types and system geometries into

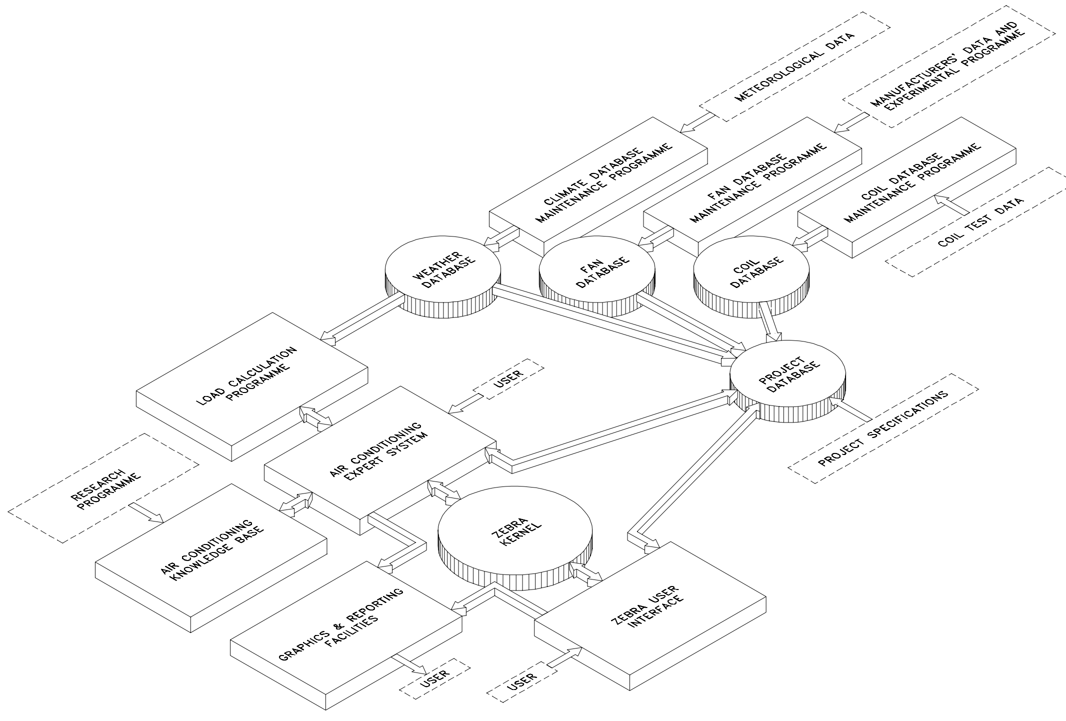
METEOROLOGICAL DATA
MANUFACTURERS' DATA AND EXPERIMENTAL PROGRAMME
CLIMATE DATABASE MAINTENANCE PROGRAMME
FAN DATABASE MAINTENANCE PROGRAMME
COIL DATABASE MAINTENANCE PROGRAMME
COIL TEST DATA
WEATHER DATABASE
FAN DATABASE
COIL DATABASE
LOAD CALCULATION PROGRAMME
USER
PROJECT DATABASE
AIR CONDITIONING EXPERT SYSTEM
PROJECT SPECIFICATIONS
RESEARCH PROGRAMME
AIR CONDITIONING KNOWLEDGE BASE
ZEBRA KERNEL
GRAPHICS & REPORTING FACILITIES
ZEBRA USER INTERFACE
USER
USER

Figure 1. Projected architecture of the ZEBRA software package.

the model.

The present paper seeks to illustrate the manner in which a structured design package such as ZEBRA may be used to enhance the designer's understanding of the behaviour of a system across its entire operating range, thus providing guidance in the selection of system configurations which will maintain specified indoor air quality conditions across the operating range of the plant, while minimising operating costs. The structure of the software is described in outline below, and a brief exposition of certain aspects of the algorithmic content of the model is provided. The structural principles of the package and its application within the context of a design situation are illustrated by reference to a novel type of air conditioning system, the High Driving Potential (HDP) system, which offers considerable flexibility in the management of indoor air conditions over a broad operating range.

2. Overview of the ZEBRA Package.

As noted by Tang and Clarke (1993), the prediction of building energy performance involves two major aspects; performance of the building envelope and performance of the plant systems. The HVAC engineer seldom has an opportunity to exert appreciable influence on the design of the building envelope, and must accept, as inputs to the design process, the loads resulting from the building design and from the climatic regime within which it operates. Against this background, ZEBRA has been conceived primarily as a tool for the HVAC designer. It accepts as input building loads calculated externally using a load calculation code such as TEMPER (Australian Construction Services, 1987) or ESP (Clarke, 1985). ZEBRA is in fact a plant-modelling tool in somewhat the same mould as TRNSYS (Klein, 1988) and ESP-r (Tang and Clarke, 1993). It is distinguished from the aforementioned codes in that considerable emphasis has been placed upon accurate modelling of the psychrometric performance of the air side of the air conditioning system, largely through a rigorous analysis of the processes occurring in the cooling coil as determined by comprehensive laboratory testing. At the same time, ZEBRA in its current form lacks the ability to model a comprehensive range of plant items (there is for instance as yet no chiller model), although the structure is such that the range of capabilities can and will be expanded in time to provide a more inclusive model of the whole air conditioning plant.

Figure 1 shows the projected structure of the ZEBRA package in its entirety. It also illustrates the relationship between the Zebra Kernel, which provides

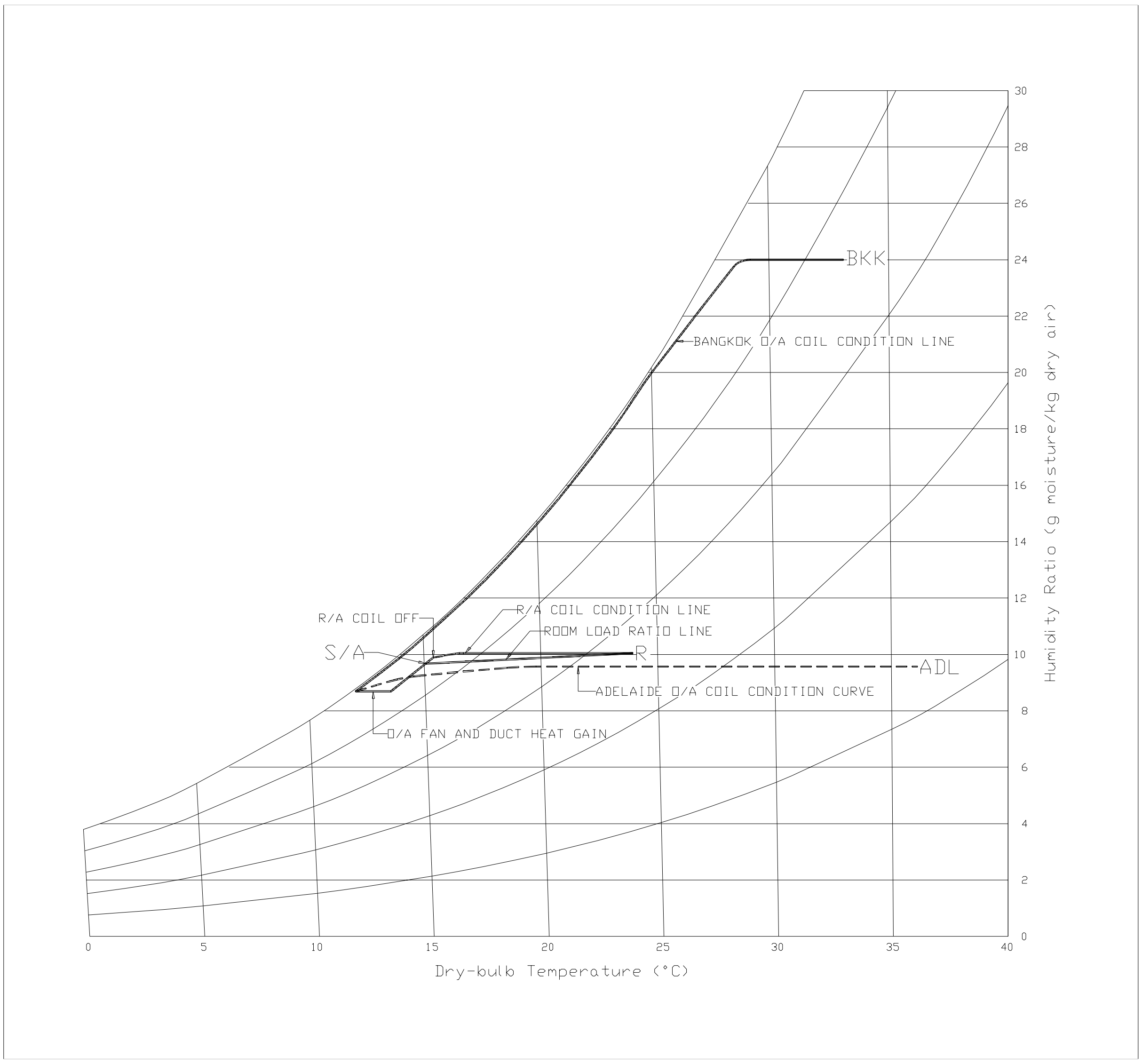

Figure 2. Psychrometrics of the High Driving Potential (HDP) System.

the simulation engine for the package, the closely-coupled user-interface with graphics and reporting facilities, and the services provided by external components such as the load-calculation programme and databases.

3. The High Driving Potential System.

Except in the rare cases where 100% outdoor air is specified, comfort air conditioning is concerned with the treatment of two air streams; the return air from the conditioned space and the outdoor air required to meet ventilation specifications. During typical operating conditions the specific enthalpies of the two streams, which each have a latent and a sensible component, will differ significantly. This is shown in figure 2, in which are plotted the process lines for an HDP system serving an idealised office building. Typical outdoor air conditions for both Adelaide and Bangkok are shown. The state of the return air in each location will be substantially the same, and is here assumed to be so. The task of the air conditioning designer is to produce a system which will offset the latent and sensible loads in the proportions in which they occur to achieve the desired climate within the conditioned space. Efficient dehumidification of the outdoor air, which may contain a significant latent component in tropical climates or during certain part-load conditions in temperate climates, requires that we make optimal use of the enthalpy potential existing between the cold coil surface and the outdoor air. This requires that the outdoor air and the return air streams be treated *separately*, a technique which is at variance with the conventional practice of mixing the two, thus degrading the potential for dehumidification, and then treating the combined air stream through a common coil. In a typical HDP application such as that described by figure 2, chilled water is fed first to the outdoor air coil and is subsequently piped to the return air coil or coils, depending on the system architecture, as discussed below. In this way, the enthalpy potential available at the outdoor air coil is sufficient not only to offset the latent component of the outdoor air stream, but also to dehumidify the outdoor air sufficiently to offset also the latent load in the return air, so relieving the return air coil of the need to remove latent heat from a stream in which the ability to remove latent heat is intimately coupled with the available sensible potential. Thus, the load on the return air coil is almost entirely sensible, and the potential offered by the now warmer chilled water from the outdoor air coil is adequate to cope with this load. The principles of the HDP system are dealt with in greater detail in Shaw et al. (1993) and Luxton (1996).

Separate treatment of the two air streams in an HDP system affords the designer considerable flexibility in configuring a system for a specific application. In essence, two basic configurations are possible:

a. The outdoor air and return air coils may be packaged within a common casing, providing a self-contained unit serving one or more zones.

b. The outdoor air may be treated in a dedicated central outdoor air treatment unit (referred to in the following as a *primary* unit), before being distributed to a network of terminal fan-coil units (referred to as *secondary* units), each serving one or more zones. As an alternative, the treated outdoor air may be distributed to a network of induction units (Petrovic and Luxton, 1996).

In designing the component data structures for the ZEBRA package, it has been necessary to provide sufficient flexibility to model not only systems of conventional configuration, but also novel system configurations such as those described above. Appropriate software design techniques are described in the sections which follow.

4. Programme Design and Structure.

The ZEBRA package has been structured from the outset using object-oriented design principles (Booch, 1991), and implemented using the C++ programming language (Stroustrup, 1991). The notation used in the following to describe object-oriented programming constructs conforms to C++ usage. The object-oriented programming paradigm is based upon the concept of the *class*, each instantiation of which, or *object*, provides a computer-based representation of some entity (which may be either real or abstract), described in terms of its *state* and *behaviour*. Object state is described by specifying *member variables*; object behaviour by specifying *member functions*. The relationship between different classes may be of one of several types, the most important of which are *using* relationships, and *inheritance* relationships. The inheritance mechanism provides a means of extending the properties of a class by defining a hierarchy in which those classes at a higher level in the hierarchy inherit and build upon the state and behaviour of classes lower down the hierarchy (known as *base* classes). In a typical application, those classes at the lower levels of a hierarchy may represent a more abstract and general concept than those at higher levels, the representation becoming more concrete and specific as one proceeds up the hierarchy. C++ also supports the concept of *multiple inheritance* in which one class may inherit attributes from two or more base classes.

In designing software systems using an object-oriented methodology, the software engineer seeks to maximise the extent to which code can be reused, and to design for ease of maintenance and extension of the design (Jacobson et al., 1992). These objectives, which have been fundamental to the philosophy underlying the design of the ZEBRA package, can be achieved by judicious selection of classes and class hierarchies, and by the placement of operations at the lowest possible level within an inheritance hierarchy.

A large number of classes have been used to represent the problem space of the ZEBRA package. Three general categories of these are of particular interest:

Framework classes are used to construct the framework within which the ZEBRA air conditioning model is built and accessed.

Sub-system classes provide, at a coarse level, the basic building blocks which may be combined as necessary to build a model of an air handling unit.

Component classes model the specific equipment items which form the internal structure of the various sub-systems.

These classes will be described in further detail in the following sections. Additional class categories provide a wide range of services such as input, editing and archiving of model specifications and encapsulation of various numerical algorithms. While these are essential to the success of the enterprise, they will only be mentioned in passing in the following.

5. Framework Classes.

The following classes provide the general framework within which the ZEBRA model is built and accessed:

Class **HVACProject**. There can be no more than one object of this class active at any time for each invocation of ZEBRA, and this provides the central conduit through which all other major components of the model may be accessed. The class maintains indexes to all systems and zones within the project, and at some level processes all requests to edit and query the state of these components, and to initiate simulations.

Class **Zone** provides a model of a space within the building. Zones may be *free* or, if served by a specific system, *conditioned*. The zone model stores the current and design loads for the space, together with the thermostat and humidistat status and settings.

Class **AHU** implements a model of the air handling unit, and defines the functions necessary to simulate the operation of this item and to access and alter its state. This is an *abstract* class in which some operations are necessarily left undefined at this level. It is designed to be used as a base class representing a generic air handling unit, from which classes providing a concrete representation of a specific type of unit can be derived. In the case of systems comprising a central outdoor-air treatment unit (see section 3 above), it has proved desirable to implement a second level of base classes (derived from class AHU) which provide generic representations of primary and secondary units.

Each system is represented by an object of class **System**, which maintains a one-to-one *using* relationship with an object of a class derived from class AHU (or class AHUPrimary where a central outdoor air treatment unit is used).

6. Sub-system Classes.

Sub-system classes simulate the major sub-systems used to build an air handling unit model. With the exception of fairly specialised system types (most notably induction units) each unit will contain a fan sub-system, represented by an object of class **FanUnit**, together with an appropriate combination of objects of class **SubSystem**, or classes derived therefrom. These latter form an inheritance hierarchy, shown in figure 3 wherein the arrows point towards the base classes, which provide the following functionality:

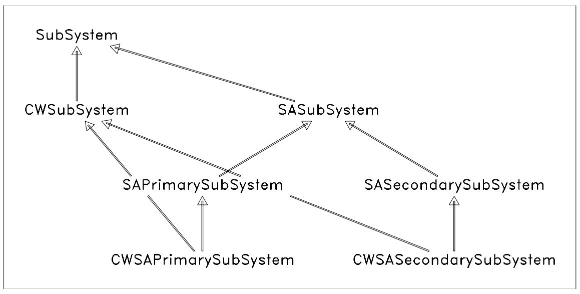

Figure 3. Inheritance hierarchy for class SubSystem and its derived classes.

- Class **SubSystem** provides a basic level of functionality, and may be used for instance to represent the outdoor-air or return-air connection to a unit of conventional construction. Member functions are provided to access the air state at entry to and exit from the sub system, and mass flow through and pressure drop across the sub-system. The sub-system may also optionally reference a filter and reheat device, using so-called ‘smart’ pointers, which facilitate memory management (Barton and Nackman, 1994).

- The additional capability of modelling a sub-system containing chilled water and/or hot water coils is provided by class **CWSubSystem**. An additional class CWCoilControlStructure has been defined for this purpose; this class packages a chilled water coil bank with a control valve, together with a set of member functions to implement the necessary control operations for the combination. ‘Smart’ pointers are again used, enabling 2-pipe and 4-pipe configurations to be simulated, with the option of specifying a shared coil or separate coils in the latter case.

- Class **SASubSystem** adds little to the base class other than a thermostat; it is a base class for classes **SAPrimarySubSystem** and **SASecondarySubSystem** which model the supply air sub-system for a draw-through unit in a primary unit in the first case, and for a secondary or self-contained unit in the second case. The first-mentioned class above maintains a list of attached secondary units and defines a set of operations on them; the second performs a similar function in respect of the zones served by a secondary or self-contained unit.

- Classes **CWSAPrimarySubSystem** and **CWSASecondarySubSystem** perform a similar function to the above for blow-through units. Note the use of multiple inheritance in respect of these classes.

Simulation of the fan sub-system is performed by class FanUnit, which maintains a using relationship with additional classes which model the following components of the sub-system:

- Fan (axial or centrifugal).
- Fan motor.
- Fan system effects at fan inlet and outlet.

7. Component Classes.

The sub-system classes described in the preceding section stand in a *using* relationship to the classes describing the individual equipment items which comprise the components from which the sub-systems are constructed. The class categories described in the preceding two sections establish a framework for system simulation. The numerical algorithms required to obtain a closure to the thermal and psychrometric processes occurring within a system are implemented within the framework and sub-system classes. The *quality* of the simulation is determined by the rigour applied to modelling the individual equipment items.

The discussion to follow is necessarily brief, and will for the most part be restricted to general principles. An extensive discussion of the algorithms used to model specific equipment items, together with the underlying theory, will be found in Marshallsay (1996).

As mentioned in section 2, the factor which we believe sets ZEBRA apart from the plant modelling strategies used by many energy estimation codes is the emphasis placed on the correct modelling of the psychrometric components of the system performance. In a cooling situation, latent heat removal is the responsibility solely of the cooling coil. The ability to predict coil performance is therefore of paramount performance, a fact which is overlooked in much conventional design practice. Judicious choice of coil configuration within a conventional setting will maximise the range over which the system will maintain satisfactory humidity levels without resorting to costly energy-consuming measures such as overcooling and reheat. In particular, coil capacity control is achieved by modulation of the flow rate of coolant through the coil. If the coolant is chilled water, the temperature rise of the water as it passes through the coil increases as the sensible load, and hence the water flow rate, decreases. If the coolant is a two-phase flow, the gas phase fraction increases as the load decreases. In each case the proportional decrease in latent heat capacity is greater than that of the sensible heat capacity. This fact, together with the fact that in many climatic regions the humidity ratio of the outdoor air frequently increases as the dry-bulb temperature decreases, which the real challenge to the air conditioning system designer.

Research at the University of Adelaide has led to development of two distinct but mutually-compatible strategies for the active control of humidity by optimising the use of available energy potentials (Luxton, 1996). The Low Face Velocity/High Coolant Velocity (LFV/HCV) method aims to maintain the potential for dehumidification by selectively deactivating sections of the coolant circuit at part-load, hence increasing coolant velocity and so maintaining a cold surface temperature through the remaining active sections. More robust and flexible control over humidity is available using the second of the strategies, the HDP method, which is described in sections 3 and 8.

Not surprisingly, the cooling coil has received a great deal of attention within the ZEBRA modelling strategy. The coils are modelled in terms of basic building blocks, using a simulation strategy which is based on a rigorous physical analysis and extensive experimental tests (Marshallsay, 1996; Marshallsay and Luxton, 1996). A coordinating set of classes permit these elementary components to be combined in a wide range of configurations, thus permitting simulation of configurations for which no direct experimental data are available, and also providing a means of simulating the case where sections of the coolant circuitry have been deactivated. The component classes are implemented as a two-level hierarchy:

- A set of classes which provide the functionality to simulate coil performance which is complete except that the relevant coolant properties (heat transfer and transport coefficients) are specified as *pure virtual* functions. These essentially serve as place holders which are overridden in derived classes to provide the behaviour appropriate to a particular coolant.

- A set of functions which are derived from the above, and which specialise their operation for a specific coolant by implementing member functions to calculate the required coefficients. The necessary classes have been implemented for chilled water; extension of the methodology to cover refrigerants and water/glycol mixtures will be implemented in the near future.

In the above manner, inheritance may be used to extend a class, and to facilitate incorporation of different working fluids into an established framework, without the need to restructure or duplicate code. In a similar manner, a properly designed inheritance hierarchy provides a mechanism to facilitate integration of new equipment types sharing common functionality and purpose with types for which a class already exists. If the base class implements a default behaviour for the type of equipment it represents, the inheritance hierarchy may also provide a mechanism for progressive refinement of a design. For instance, in the early stages of a refrigeration design exercise, one might specify a base class representing the process occurring within a compressor as one of adiabatic compression. As the design process proceeds, one might select in its place a derived class representing a specific compressor type (e.g. centrifugal), and accounting in an appropriate manner for departures from the adiabatic assumption.

8. HDP Units.

In this section we illustrate the concepts developed above by considering the manner in which sub-systems may be combined to model a self-contained HDP unit. The sub-systems required are:

- A fan sub-system.
- An outdoor air sub-system, implemented as an object of class CWSubSystem.
- A return air sub-system, also implemented as an object of class CWSubSystem.
- A supply air sub-system, implemented as an object of class SASecondarySubSystem.

In addition, we postulate the existence of a valve (V in figure 4) controlling the flow of coolant to the unit. Each of the valves shown (V, O and R) can assume one of four states:

- Open.
- Closed.
- Set (to a specified position between Open and Closed).
- Modulating.

Most combinations are valid, the major exception being that valve V cannot be set to Modulating if either of valves O and R are modulating. If it is desired to omit any valve from the simulation, this can be achieved by specifying its state as Open. Setting one valve to Modulating, it is possible to control zone dry-bulb temperature; zone humidity must be retained within acceptable limits *by design*. The range of applicability of a design may be extended, or operation of the unit optimised across its range, by the use of *staging*. Each stage specifies a set of valve positions, and may

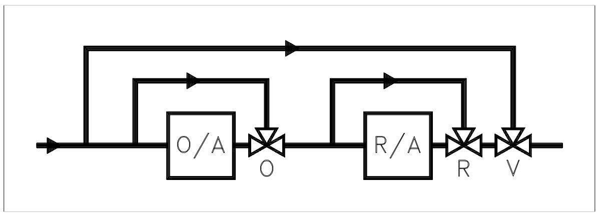

Figure 4. Coolant circuiting diagram for self-contained HDP unit.

optionally specify deactivation of circuits in one or both coils. Transition from one stage to the next may be triggered by a rise or fall in some parameter such as coolant velocity. If both valves O and R are set to Modulating, it is possible to control humidity levels explicitly within a specified band, provided at least one of the zones served by the unit has a humidistat. Thus:

- If zone humidity rises above the upper limit, valve O is opened and valve R is closed simultaneously to meet zone dry-bulb temperature conditions, until the upper humidity constraint is again satisfied.

- If zone humidity falls below the lower limit, valve O is closed and valve R is opened simultaneously to meet zone dry-bulb temperature conditions, until the lower humidity constraint is again satisfied.

9. Design Strategy.

Life-cycle design requires that a system should provide satisfactory performance across its range of operation. For this purpose, ZEBRA makes available to the user a partial automation of the calculation, known as the *metafile* system. A metafile contains a set of instructions to run a candidate design solution through a pre-determined sequence of zone loads and their associated ambient conditions. The load sequence may represent a time-based sequence output by a load calculation programme. As an alternative, a sequence of critical loads derived from a design locus may be used as the basis for a systematic evaluation of system performance (Koptchev and Luxton, 1996).

It is important to bear in mind that the design process involves more than simply selecting equipment with sufficient capacity to offset peak sensible loads; it also involves sensible zoning of the conditioned space and partitioning of the load among a number of units, together with selection and evaluation of equipment and control strategies (staging) to meet the design constraints in a cost-effective manner. With the ability to simulate system performance rapidly across its entire range of operating conditions, it becomes possible to make an objective comparison of various candidate design solutions.

10. Conclusions and Further Directions.

The ZEBRA software package currently provides a comprehensive tool for modelling the air-side performance of air conditioning systems employing chilled water as a cooling medium. The package is constructed using object-oriented design principles, and a number of extensions are being developed, these including the implementation of a direct expansion (DX) system model and development of a duct system model. Development of the software systems is continuing in parallel with an experimental programme to measure and parameterise equipment operating characteristics, and with research into system design methodologies.

References.

Australian Construction Services, 1987, *User Guide for Computer Program TEMPER*, Mechanical Engineering Services Design Aids No. DA14.

Barton, J.J. and Nackman, L.R., 1994, *Scientific and Engineering C++: An Introduction with Advanced Techniques and Examples*, Addison-Wesley, Reading, MA.

Booch, G., 1991, *Object-Oriented Design with Applications*, Benjamin/Cummings, Redwood City, Calif.

Clarke, J.A., 1985, *Energy Simulation in Building Design*, Adam Hilger, Bristol.

Jacobson, I., Christerson, M., Jonsson, P. and Övergaard, G., 1992, *Object-oriented Software Engineering: A Use Case Driven Approach*, Addison-Wesley, Reading, MA.

Klein, S.A., 1988, *TRNSYS - A Transient System Simulation Program*, Technical Report of the Solar Energy Laboratory, University of Wisconsin, Madison, WI.

Koptchev, I.D. and Luxton, R.E., 1996, A new look at air conditioning design, *Proc. Joint Meeting of Commissions E1, E2, B1 and B2, IIR*, Melbourne, Feb. 11-14, 1996.

Luxton, R.E., 1996, The changing climate of air conditioning design - confronting the challenge, *Proc. Joint Meeting of Commissions E1, E2, B1 and B2, IIR*, Melbourne, Feb. 11-14, 1996.

Marshallsay, P.G., 1996, A Methodology for Modelling the Steady-State Thermal Performance of Air Conditioning Systems, *Ph.D. Thesis*, The University of Adelaide.

Marshallsay, P.G. and Luxton, R.E., 1996, Cooling coil performance simulation using the ZEBRA software package, *Proc. Joint Meeting of Commissions E1, E2, B1 and B2, IIR*, Melbourne, Feb. 11-14, 1996.

Petrovic, V.M. and Luxton, R.E., 1996, Induction air conditioning system for humid climate, *Proc. 1st International Conference on Heating, Ventilation, Refrigeration, Purification and Air-conditioning, Singapore*, 23-25 July.

Shaw, A., Luxton, R.E., and Marshallsay, P.G., 1993, Integration of dehumidification into life-cycle system design, *Proc. Conference on Building Design, Technology and Occupant Well-being in Temperate Climates*, Brussels, 17-19 February.

Stroustrup, B., 1991, *The C++ Programming Language*, 2nd Edition, Addison-Wesley, Reading, Mass.

Tang, D., and Clarke, J.A., 1993, Application of the object oriented programming paradigm to building plant system modelling, *Proc. IBPSA 3rd International Conf.*, Adelaide, Aug. 16-18, 1993.