SchedSync – a Handy Planning Tool for Airline and Other Transport Networks

Wenhua. Wei[1, 2], Christopher. Kissling[1]
[1] Lincoln University, PO Box 84, Christchurch, New Zealand
[2] Roslin Institute, Roslin, Midlothian, EH25 9PS, UK

1. Introduction

One of the author's (Kissling) has been involved in research on South Pacific Island aviation since the mid 1970s (Kissling 1981, Taylor 1983, Kissling 1990). It was one of his papers on rationalization of airline networks that sparked the interest of a fellow academic professor in applied computing as the paper included an aircraft utilisation chart showing how four airlines in the South Pacific could adjust their schedules to provide better direct connections for passengers trying to fly between island states rather than to force them to fly to the Pacific Rim states and back again. The professor of applied computing wished to use the diagram to demonstrate visualisation techniques to his computing students. He also recommended one of his PhD scholars, Wenhua. Wei, to work with Professor Kissling to develop the model we have presented here.

Our desire was to create a user friendly transport route planning tool which could be adapted for many different situations. It was also our desire to use software familiar to potential users, particularly our students. We could find no parallel applications in the literature that incorporated the convenience of the Microsoft Excel platform in the manner developed in our SchedSync model. The programming challenge for Wenhua Wei resulted in some very creative software development. The outcome is a tool that can be adapted to other transport scheduling problems.

A regional air transport network can be very complicated when a large amount of flight data are analysed. These data, for example flights listed from the Official Airline Guide (OAG), are normally organized in a tabulated format, either indexed by airport, or by airline or something else. Based on such a format, it may not be too difficult to find a transport route, but could be difficult for administrators or business developers to discover/develop regional cooperation via coordination of airline routes.

To seek greater integration and point-to-point accessibility, a graphic platform that can display scheduled flights and allow for the testing of possible rationalization scenarios is a highly desired tool. Such a tool is expected to meet the following requirements:

1. be able to handle flight data and allow easy navigation and manipulation
2. be able to graphically display many flights precisely to match flight properties such as departure and arrival airports and times
3. be able to handle differences among time zones
4. allow easily changing flight schedules as well as creating new flights
5. allow manipulating of user-defined routes

As a teaching and research tool, SchedSync has been successfully applied to the air networks of South Pacific Island airlines. To share with the transport community how SchedSync is developed, this paper discussed the design philosophy and the algorithms used in the program development. Some thoughts for further refinement are also discussed.

2. System Development Context

A computer program, named SchedSync, was developed under the Windows environment using object-oriented (OO) style, to meet the requirements described above. It has two inter-

connected components: a VB component and a Microsoft Excel platform. The VB component contains an embedded OO database, dedicated to manipulate flight data including airports, airlines, flight numbers, aircraft types, flights and routines. The Excel platform, written as a template using VBA, has a well-developed graphic interface to plot flights as a 2-D graph, where airports are aligned vertically based on their longitudes, time is distributed horizontally in a two-week cycle (accurate to one minute), and flights are represented as Excel line objects.

3. System structure

It is well known that an air flight is described by a set of properties including an official flight number (FlightNo used in code sharing), an airline that provides an aircraft, the aircraft, departure and arrival airports, departure and arrival local times, and other properties that are not relevant to this topic. These properties can be further categorized into dimensional and non-dimensional groups. The dimensional group covers departure and arrival airports and times, whereas the non-dimensional group contains FlightNo, airline and aircraft that are not related to time or spatial dimensions. This understanding forms the core concept used in the SchedSync design and implementation.

Based on the core concept above, a simple data model (Figure 1) was composed where flight data are input into a database, manipulated, and saved back to disk. These data can also be plotted to an Excel graphic platform. On the platform, plotted flight data can be edited and new flights can be added. Changes made on the platform can be saved back to the database for further use.

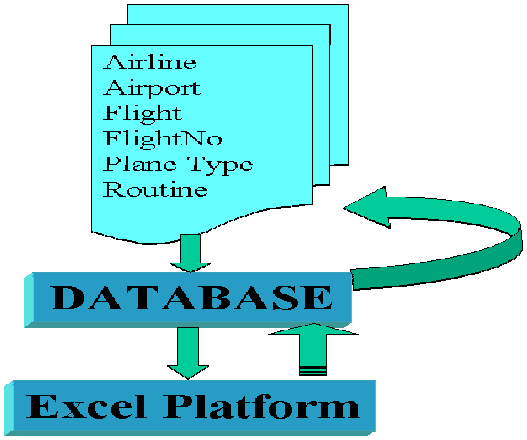

Figure 1 Data model of SchedSync

The data model contains major system requirements that led to a two-component design (Figure 2). When SchedSync starts, it allows either to load a working model from an Excel file resulted from the platform, or to load flight data from system data files saved from the database. After flight data are ready for plotting, a work form controls plotting and saving from the Excel platform. A filtering function is also built in the SchedSync to allow data filters to be defined by either airlines, or airports, or weekdays, or combined properties to assist rationalization.

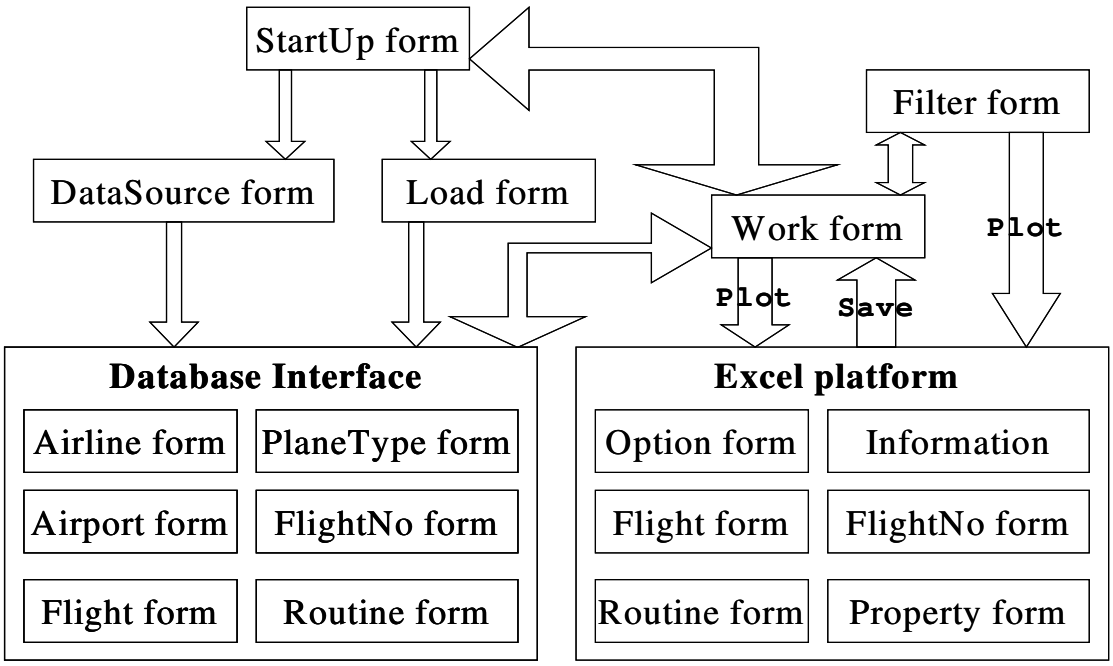

Figure 2 SchedSync overall system structure

3.1 Database design

An embedded database to store and maintain flight data was implemented using the OO concepts and techniques where each record is actually an object that has properties and behaviours that allows better implementation and system maintenance. The database was developed as simply as possible, containing Airline, Airport, PlaneType, Flight and Routine classes (Figure 3).

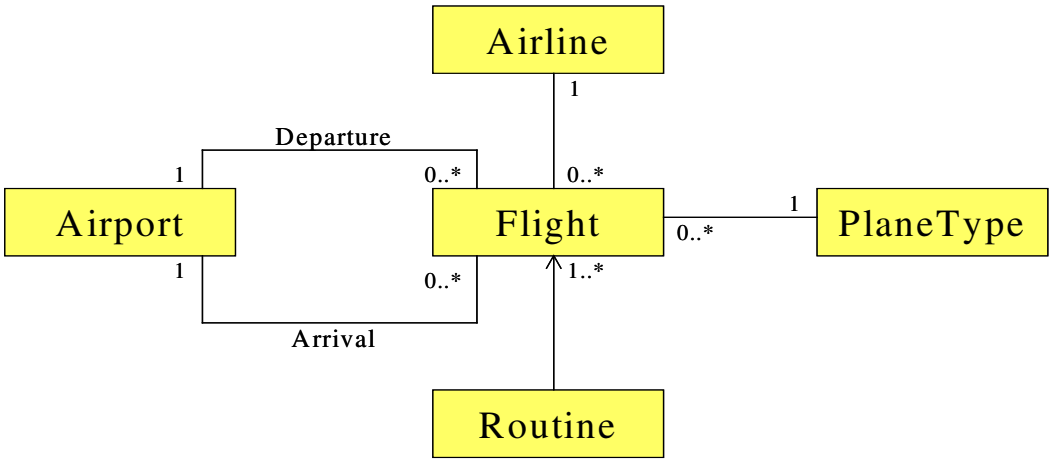

Figure 3 SchedSync flight database structure

Because SchedSync is not for air traffic control, airplane objects are ignored in SchedSync. Instead, a PlaneType class that contains information of seats or space is used for rationalization. A FlightNo, as a sharing code, is actually associated with an airline, a departure airport and an arrival airport. However, to simplify the implementation, FlightNos are actually stored in a global string array that is referenced directly by Flight objects managed in the database.

The interrelationship among the five classes (Figure 3) is straightforward. In the database, a Flight object must be associated with one airline, one aircraft type, one departure airport and one arrival airport (FlightNo, departure time and arrival time are defined as properties of such a Flight object). On the other hand, an airline or an airport or a plane type can have none to many flight objects associated. A Routine object is actually a collection of sequentially linked (by airport) flight objects. Consequently, a routine must have at least one flight but a flight can be part of none to many routines.

It is worthwhile to point out here that, from the function viewpoint, the database does not have to be implemented in the OO style. However, because a flight database design is relatively standard, it was therefore determined to implement that way to allow maximum code reuse in future extension or similar software development.

3.2 Excel graphic platform design

Microsoft Excel was chosen to develop the graphic platform for mainly two reasons: 1) it is large enough to display many airports and many flights in a two-week cycle; 2) it has line objects that can be used to simply represent a flight. To provide a consistent graphic platform for each flight data plotting, a template was developed where airports are aligned vertically based on their longitudes and local times are aligned horizontally (Figure 4). To assist manipulating flights on the platform, a new menu item and a number of user forms are developed to provide operation functions such as to display flight information, create and save flights, create/save routines. It also has a Dateline function to remind users whether a flight is crossing the date line when considering the time difference between departure and arrival.

With such a platform, users can see the whole picture immediately and find gaps easily to develop new flights and routes in the system. It is also handy to change a flight schedule via changing the line (flight) object and to create a routine. Any changes made to a flight object, will be permanent to the Excel file (the “PlotData” worksheet) after being saved. However, these changes will only be available to the current model/plot unless users also save the flight data back to SchedSync database described above.

4. Algorithms

One of the major challenges of the platform is to display each flight precisely. This essentially requires a good management of Excel line objects that are controlled by two pairs of coordinates. Unfortunately, Microsoft Excel does not provide sufficient controls of the line objects. Consequently, some research was conducted to develop algorithms of formatting the platform to allow consistence and high precision in displaying a flight, and matching a line object accurately against properties of a flight.

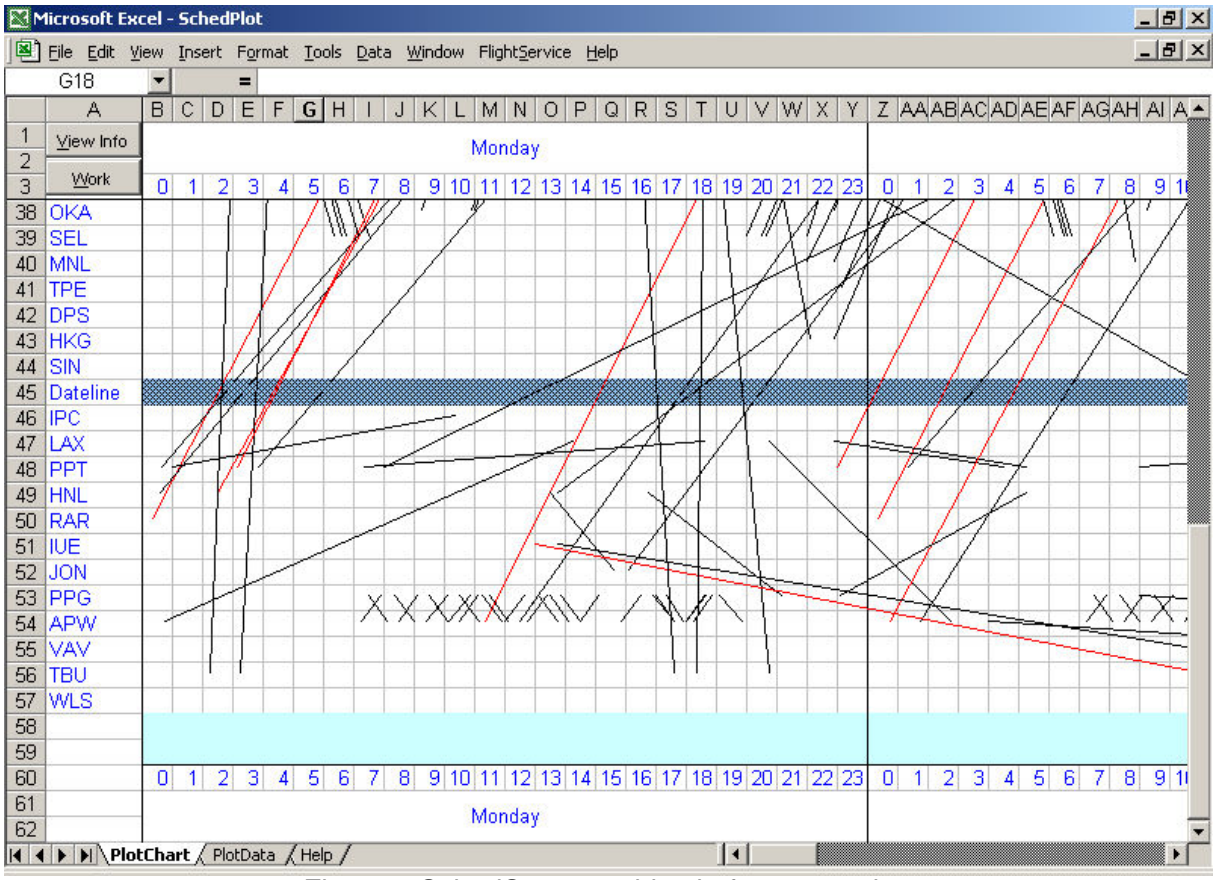

Figure 4 SchedSync graphic platform snapshot

4.1 Platform formatting

A line object in Microsoft Excel has a start point and an end point, both of which are measured in a pair of Top and Left properties, with a unit named point (also pixel, used in describing screen resolution). A Top property is a screen position relative to the top end of a worksheet (eg. zero for the beginning of row one), whereas a Left property is a screen position relative to the left boundary of the worksheet (zero for the beginning of column one). Consequently, to display a flight (line) object precisely in the 2D platform (Figure 4), it is critical to have fixed row height, column width and screen resolution.

Microsoft Excel uses different mechanisms to control row height and column width. The row height is measured in points, where one point is 1/72th of an inch or 0.35 mm. A default row height is 12.75 points that are roughly 0.50 cm. The column width, however, is equal to the number of digits that can be fitted in a cell when a default font is used. If a default font uses Arial with a size of 10, the column width is 8.43, which means that the digits 123456789 (9 digits) will just exceed the column width.

Therefore, the row height can be exactly set. But setting of the column width has to be established by measurements and calculations to work out the regression relationship between column width and points to pixels under a given font setting (roughly resolution). The reason is that computers use only integer values of points/pixels to display objects. If a calculation of departure time returns a decimal point value, it will be either truncated or rounded to the nearest pixel for display purposes.

Consequently, when setting the font as Arial with a size of 10, the optimum column width discovered is 2.14 (fitting between 2 and 3 digits) that will create 15 points per column. If

such a column is used to represent one hour, one minute is represented by 0.25 points, which is the minimum interval under the current settings. It is apparent that such a detecting precision is perhaps only sensible to a computer rather than human eyes. However, fortunately, within one minute difference in time is not a real issue in the real world so this algorithm is acceptable.

4.2 Work with Flights

Once flights are plotted on the platform, they are just Excel line objects (a group of shapes in terms of a routine). For such a line object, all that Excel knows about it are object name (eg Line 1) and properties of Top, Left, Width and Height (Figure 5). The platform should be able to work out properties of a flight: to fetch departure and arrival airports from the Top and Height properties, departure and arrival time from the Left and Width properties. Different alignment cases complicate the matching algorithms (Figure 5)

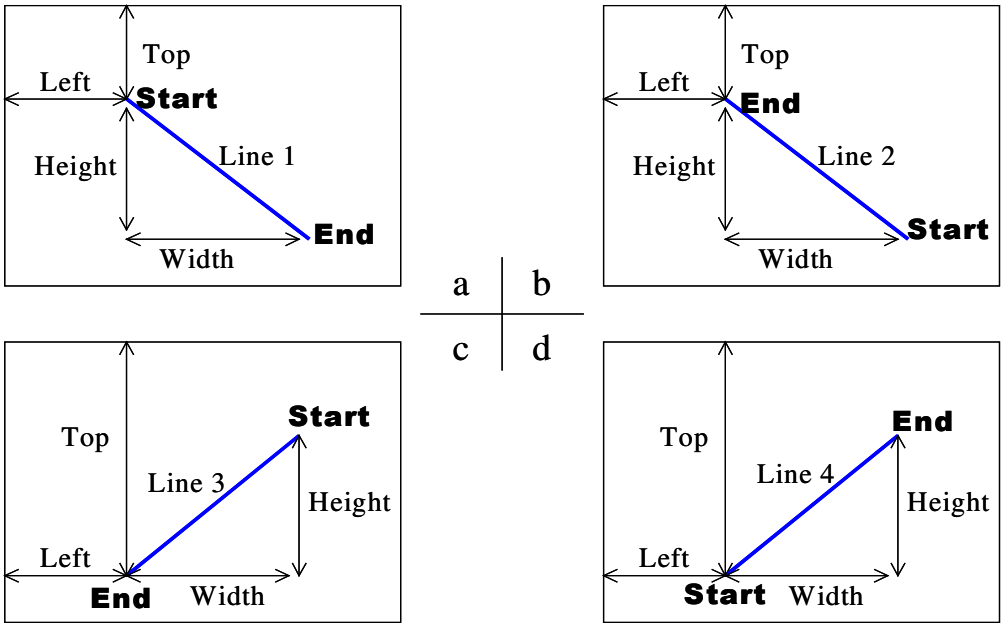

Figure 5 Four flight alignment cases

Set up lookup tables

When plotting flight data, SchedSync takes the following sequences:

To align airports

1. work out how many airports are involved
2. sort them in the order of longitudes
3. calculate exact positions (in points) for each airport based on the longitude and the fixed row height (Section 3.1)
4. plot airports vertically with consideration of Dateline (if necessary)

To align date and time

1. work out how many weekdays are involved
2. calculate exact positions (in points) for each weekday based on the day number, fixed column width
3. plot weekdays and hour numbers within each day

It becomes obvious immediately that lookup tables are necessary for both airports and weekdays. Once data are first plotted, these lookup tables are fixed implying that the user is not expected to add new airports or weekdays into the current model.

A lookup table is also necessary for each flight because both departure and arrival times have extra information of hours and minutes. Since there is a linear relationship between time and screen points, it is relatively straightforward to convert between them. The flight lookup table also contains other flight properties such as a unique Flight code and PlaneType that cannot be represented by a line object. Furthermore, the name of the line object, for example Flight1 that is defined at plotting, is also included in the table to provide a quick lookup.

To facilitate the needs of creating new flights that may require new FlightNos and aircraft types, lookup tables for FlightNo and PlaneType are also set up to prevent duplicate entries.

Lookup airport and time

To lookup an airport is straightforward – users just need to find out the point value (either the Top value or the sum of Top and Height, Figure 5) and use the lookup table to match an airport.

To lookup a time is not difficult either. Once a point value is found (either the Left value or the sum of Left and Width, Figure 5), one needs to firstly work out in which weekday block it is located, then calculate hours (constant 15 points per hour) and then minutes (constant 0.25 points per minute)

Lookup flight

To lookup a flight is even simpler because it is easy to use a predefined line object name to search through the flight lookup table. In this case, SchedSync just needs to find a matched object as a whole instead of calculating individual properties.

Manipulate flights

There are two ways to manipulate flights on the platform: via user forms and via line objects. It is easy to change properties of a flight or create a new flight using user forms. This is almost the same as working in the database because the line objects are not directly involved. After changes are made, user just needs to replot the flight data to see the changes graphically.

However, if changes are made directly on a line object through normal drawing operations, to save the changes is a bit tricky because, as demonstrated in Figure 5, the platform needs to be sure which airport and at what time is the departure! When this happens, the platform will ask user to supply this information before saving the changes.

The platform also uses red colour to highlight flights that have local departure time later than local arrival time to catch user's attention (eg. departing at 9:00 am Saturday and arriving at 10:00 pm Friday). This case is easy to understand when a flight is to cross the Dateline. However, it is also possible that data were entered incorrectly.

Data synchronization

It is important to keep data synchronized both within the platform and between the platform and the database. Once a set of flights are plotted, the platform can be used as a standalone Excel application, where users can rationalize different transport scenarios using the given data set. As mentioned above, flight properties can be changed via manipulating the corresponding line objects or using relevant forms. If changes are made from the user forms, a save-replot algorithm, ie save to lookup tables first then replot flights, is used to keep the line objects and the lookup tables in sync. Otherwise, it is the users' responsibility to save changes made via Excel line objects. However, if the user forgets to save those changes, when the Excel application is about to close, the platform can still detect the

changes and ask the user to optionally save them. When change for a flight is saved, the platform keeps a flag based on the comparison against the original Flight code that was recorded in the initial plotting from SchedSync.

When the user wants to merge changes made on a platform back to the database, SchedSync allows this to happen by checking the flag of change. If there is no change in a flight, it is skipped as there is no need to save. Otherwise, users will be asked to update the original flight or to create a new flight object in the database.

5. Discussion

Some airports do not operate 24 hours a day as they have curfews imposed. A few are restricted such as when flights on Sundays are prohibited except in emergencies. The charts can show airports that have operationally restricted times simply by shading. If plotted flights arrive or depart in the shaded horizontal row for a particular airport, it will immediately be obvious that corrections will be needed.

Again, many airports have a maximum number of parking positions for aircraft either on the apron in front of the terminal, or at air-bridge positions. If too many aircraft arrive at an airport and exceed the parking slot limits, alerts could be implemented and flight timings adjusted accordingly.

Visual representation of transport movements can be a powerful means of testing numerous 'what if' situations. Some parts of the operational environment are likely to be beyond the control of a particular agency, but they can fiddle with their own sub system to seek best possible integration into the overall system. Several agencies seeking to co-operate, can use this tool to test alternative scenarios before agreeing upon a best fit model.

As users become familiar with this planning tool, there will inevitably be suggestions made that could lead to further refinement.

6. Conclusion

As yet, South Pacific aviation routes involving Pacific Island microstates have been the main focus for input to SchedSync. More recent investigations have considered how multiple aircraft in an all-business class configuration under the management control of a single airline might be deployed on Trans-Tasman services involving the Australian airports of Brisbane, Sydney, Canberra and Melbourne, and the New Zealand airports of Auckland, Wellington and Christchurch. The objective is to schedule the most desirable weekday departure and arrival times whilst maintaining high utilisation of as few aircraft as possible commensurate with aircraft maintenance requirements and the need to avoid airport curfews at Sydney and Wellington.

Another aviation application is in the field of scheduling flights from regional centres to connect smoothly with the main trunk services between major population centres. Anything that can minimise waiting in terminals to make a connection would seem to be a bonus for travellers. This is especially so for international tourists who of necessity have to change flights from the large jets used on long-haul international services, to smaller regional jets for connections between main centres, and then again to smaller turboprop aircraft to reach more remote target destinations such as Milford Sound in New Zealand. Missing any connections where flight frequency is low can cause severe anguish for travellers, and added cost for airlines if passengers have to be accommodated until they can be rebooked.

A long-haul trucking firm in New Zealand, involved in scheduled movements on trunk routes in both North and South Islands, is looking to use SchedSync to help coordinate regional freight transfers between routes. Sometimes it is counterproductive to transfer freight at the

first available opportunity as that stoppage for a small volume will adversely affect the delivery time for the majority of the consignments on board the vehicle. SchedSync may also help managers identify points of convergence where drivers can swap rigs thereby enabling them to return to their home bases and family dwellings within timeframes that do not violate strict regulations on driving hours. Such an application is appropriate for a final year undergraduate research project in the BCom(Transport and Logistics) degree at Lincoln University, New Zealand. It helps forge positive relationships between industry and our students with a strong possibility that it will lead to a job offer.

References

Kissling, C C (1981) Rationalisation for Survival in South Pacific Civil Aviation, *Australian Geographical Studies*, 19 No 2, 205-216

Taylor, M J and Kissling, C C (1983) Resource Dependence, Power Networks and the Airline System of the South Pacific, *Regional Studies*, 17 No 4, 237-250

Kissling, C C (1990) Management Issues in Pacific Island Aviation and Tourism, in *Destination South Pacific: Perspectives on Island Tourism*, Christopher Kissling (ed.), *Cahiers du Tourisme*, Series B, No 59, Centre des Hautes Etudes Touristiques, Aix-en-Provence, pp 13-30