出國報告(出國類別：實習)

赴美國研習「航空氣象現代化作業系統之網頁產品處理技術」報告書

服務機關：交通部民用航空局飛航服務總臺
姓名職稱：張友忠 臺　長
　　莊清堯 預報員
派赴國家：美國
出國期間：100 年 10 月 1 日~100 年 10 月 21 日
報告日期：100 年 12 月 15 日

提 要 表

系統識別號：	C10004680
計畫名稱：	赴美國研習「航空氣象現代化作業系統之網頁產品處理技術」報告書
報告名稱：	赴美國研習「航空氣象現代化作業系統之網頁產品處理技術」報告書
計畫主辦機關：	交通部民用航空局
出國人員：	姓名 / 服務機關 / 服務單位 / 職稱 / 官職等 / E-MAIL 信箱 張友忠 / 交通部民用航空局飛航服務總臺 / 臺北航空氣象中心 / 臺　長 / 薦任(派) 莊清堯 / 交通部民用航空局飛航服務總臺 / 臺北航空氣象中心 / 預報員 / 委任(派) / 聯絡人 ufvejuan@ms1.anws.gov.tw
前往地區：	美國
參訪機關：	美國國家大氣科學研究中心(NCAR)
出國類別：	實習
出國期間：	民國 100 年 10 月 1 日 至 民國 100 年 10 月 21 日
報告日期：	民國 100 年 12 月 15 日
關鍵詞：	AOAWS-TE，AOAWS 系統，AWOS 顯示系統，JMDS
報告書頁數：	62 頁
報告內容摘要：	「航空氣象現代化作業系統氣象技術增強計畫」(AOAWS-TE)今年爲進入計劃第一年，隨著各種資料顯示系統不斷的升級改版，系統管理維護人員勢必面臨更嚴峻的維護與調整課程，因此今年針對如何在 AWOS 顯示系統新增機場資料的主軸上，藉由課程講解與實機操作，使參訓人員徹底了解 AWOS 顯示介面的架構與維護升級辦法，對於航空氣象現代化系統中 AWOS 資料顯示介面結構有更深的了解與認識，同時也向自行排除問題及維護的層次達到更上一層樓的目的。
電子全文檔：	C10004680_01.doc
出國報告審核表：	C10004680_A.doc
限閱與否：	否
專責人員姓名：	陳碧雲
專責人員電話：	02-23496197

目次

壹、 目的

航空氣象現代化作業系統(Advanced Operational Aviation Weather System，AOAWS)，91 年 6 月驗收完成並正式啓用，已在桃園、松山、高雄機場的氣象臺和諮詢臺以及區管中心和臺北航空氣象中心，分別建置了多元化產品顯示系統（Multi-dimensional Display System，MDS），擔負起臺北飛航情報區的航空氣象服務業務，包含了提供本區及氣象中心收集的各項氣象觀測資料，如各機場觀測報告、雷達回波觀測顯示、衛星雲圖等資料；同時亦提供本區的預報資料，如機場天氣預報、氣象模式預報資料。此外，臺北航空氣象中心針對本區的航路天氣發布低空危害天氣警報(AIRMET)及顯著危害天氣警報(SIGMET)的資料及警示區域也可即時顯示更新資料供使用者查詢使用。

95 年開始的「航空氣象現代化作業系統強化及支援計畫」(Advanced Operational Aviation Weather System Enhancement and Support，AOAWS-ES)，採用天氣研究與預報模式（the Weather Research and Forecasting model, WRF）取代原中尺度數值天氣模式（Mesoscale Model version 5，MM5），而 WRF 模式預報範圍以外之全球資料，則接取國際民航組織的世界區域預報系統(World Area Forecast System，WAFS)的資料，同時以 JAVA 語言爲基礎，建置全新的顯示介面，稱爲新一代爪哇版多元化產品顯示系統(Advanced Java-based Multi-dimensional Display System，JMDS)。新的產品顯示介面可以透過網際網路在不同的作業系統上執行，不再受到作業系統及主機位置的限制，使得航空氣象資訊服務得以有效地擴展到各個使用者與單位。

直至今(100)年起，爲期四年之「航空氣象現代化作業系統氣象技術增強計畫」（Technical Enhancement for the Advanced Operational Aviation Weather System，AOAWS-TE）將下列六項主要的氣象技術列爲增強發展重點，以再次提升航空氣象服務品質：

（一）航空數位資料視覺化服務技術；
（二）氣象雷達亂流偵測技術；
（三）即時積冰潛勢偵測技術；
（四）圖形化亂流指導技術；
（五）機場雲幕高和能見度預測技術；
（六）危害天氣資訊與航管系統(ATM)系統整合技術；

爲持續提升航空氣象服務品質，臺北航空氣象中心配合民航局賡續辦理派員出國研習計畫，學習 AOAWS 系統資料處理及網頁顯示之進階技術，加強未來面臨問題之處理能力，以便返國後能夠將所學運用於工作中，爲航空氣象服務品質提升，盡一份心力。而今年度爲了使得本總臺氣象中心人員，對於航空氣象服務網中之自動天氣觀測系統(AWOS)資料顯示介面維護，更上一層樓。特選定此爲訓練主軸，前往美國國家大氣科學研究中心(NCAR)進行完整相關之產品訓練。

貳、 過程

10月1日星期六傍晚，我們一行四人，包括職等二人和資拓宏宇科技公司(IISI)兩名工程師，搭乘下午1840分BR-12長榮班機飛往洛杉磯。由於時差的關係，一行人於洛杉磯時間同一日的下午1335左右抵達了洛杉磯國際機場。

接著轉往洛杉磯國際機場第七航站搭乘聯合航空UA-364於洛杉磯時間下午1834分起飛前往丹佛班機。但因美國升高安檢規格，約莫花了2個多小時才通過美國入境海關。所幸趕上原定班機前往丹佛，經過2小時20分的航程，在丹佛時間將近晚上2200分，我們抵達丹佛機場後，並搭乘計程車到住處休息。

10月2日星期日，在NCAR人員Celia的協助下，前往住處安頓，並去超市採買食品及蔬果。由於舟車勞頓，因此我們並沒有別的計畫便早早回住處休息，準備明日在NCAR準備登場的課程。

10月3日星期一，由於本次訓練安排較多的實機操作時間，所以當天上午由NCAR人員Gary Cunning及Jim Cowie協助，設定未來爲期三週課程需要的主機及網路環境。而下午由Gary Cunning爲我們說明AWOS資料顯示系統架構，資料接收、資料轉化、資料除錯、參數設定及主機程序等等。由於本次受訓人員皆已對於系統架構有相當程度的了解，故顯得駕輕就熟。

10月4日星期二，由NCAR人員Paul Prestopnik帶領我們進行於前日完成設定之主機實機課程。我們選定四個機場(分別爲馬祖北竿機場、臺東豐年機場、綠島機場及蘭嶼機場)，未來將陸續納入航空氣象服務網之機場AWOS資料進行訓練。由於上述機場之AWOS尚無真實資料輸出，所以今天的課程重點在於如何創造虛擬環境？而我們經由NCAR人員的指導下，將目前松山機場的AWOS過去資料，依受訓人員所負責進行之機場名稱及跑道名稱等等參數進行調整。最後寫入專用之資料發送主機，並經由該主機資料輸出程式，將上述機場之AWOS資料模擬成經由串列設備連網伺服器(MOXA)輸出，與實際作業相同情況。

10月5日星期三，同樣由NCAR人員Paul Prestopnik繼續帶領我們在電腦主機前，設定資料接收程式。將昨日寫入發送主機之資料，透過程式進行接收。這對於未來接收新的AWOS資料相當重要，因爲資料接收是完整資料處理的開始，不過由於NCAR所撰寫的資料接收程式相當完整，所以只要將放置資料的目錄及接收資料來源的網路位置寫入參數之中，即可開始接收資料發送主機的資料。另外學習如何將資料接收及發送程序寫入主機自動啓動程序的規則，未來在主機進行重新啓動後，所有應用於作業的程序將自動啓動。當天下午，我們爲了記錄本次訓練的完整過程，經過討論後，決定撰寫AWOS資料顯示介面資料程序調整及開發手冊。

10月6日星期四，依然由NCAR人員Paul Prestopnik帶領我們進行資料轉化的

課程。昨日已將資料順利接收至資料主機之中，但在航空氣象現代化作業系統中，資料必須進行資料轉化後，才能供系統讀取使用。但資料轉化牽涉到系統原始碼的調整，故需要進行相關程式的調整與撰寫。由於我們使用的是訓練主機，所以相關撰寫與調整程式的作業環境參數尚未設定完成，故也趁此機會了解各環境參數的設定方式。在設定完成環境參數後，我們開始分頭進行各機場AWOS資料轉化程序之原始碼調整。完成後並設定對應的程序參數，最後將已經接收到的原始資料經由調整後的程序處理，最後查詢資料是否成功轉化。剩餘時間繼續撰寫AWOS資料顯示介面資料程序調整及開發手冊，並將今日課程納入其中。

10月7日星期五，我們自行練習前兩天所學部分，確保學習後的熟悉程度。並學習在不同的查詢參數下，查詢資料的方式。當天下午我們開始將共四個版本的資料轉化程序，整合成一個可同時處理現有松山、桃園、高雄及金門機場與未來將增加之馬祖北竿、臺東豐年、綠島及蘭嶼機場之資料轉化程序。並於整合過程中學習如何偵錯、除錯及調整程式碼或參數檔設定錯誤的處理方式。經過一番努力後，終於將資料轉化程序整合完成。剩餘時間則繼續撰寫AWOS資料顯示介面資料程序調整及開發手冊，並將今日課程納入其中。

10月8日、9日為週休，未安排課程。

10月10日星期一，我們將前一周所撰寫之AWOS資料顯示介面資料程序調整及開發手冊進行互相討論，並納入認為需要說明清楚與加強以及容易發生錯誤的地方，使得未來使用此手冊的人可以更容易進入狀況。我們也趁機再次了解與熟悉前週之課程內容。而下午則進行系統監控畫面的設定，當新的資料納入後，航空氣象現代化作業系統必須新增相關之資料與處理程序的監控。下午則由NCAR人員Paul Prestopnik帶領，由於參訓人員對於這部分已有相當的實務處理經驗，這部分之課程進行相當順利。

10月11日星期二，我們持續將之前所學納入AWOS資料顯示介面資料程序調整及開發手冊，並反覆確認程序之正確性。最後與NCAR人員Gary Cunning及Jim Cowie討論後，決定將本手冊在本次訓練結束後，將由NCAR進行最後確認，並翻譯成中英兩個版本，方便日後進行系統維護升級使用。

10月12日星期三，今日是今年度航空氣象現代化作業系統氣象技術增強計畫第14號執行辦法(IA#14)於NCAR舉行之專案管理會議的日子，總臺氣象中心陳副主任海根、許主任氣象員依萍、飛航業務室張課長翠分、臺北進場管制臺簡管制員義逢及資拓宏宇科技公司鄧協理秀明前來與會。而我們一行四人亦參加本次會議，共同與NCAR與會人員討論今年度工作項目執行情況，明年度工作項目之重點，會中達成多項共識。同時也讓我們參訓人員能夠更進一步了解航空氣象現代化作業系統各項工作之意涵與完成後之效益。

10月13日星期四，由NCAR人員Aaron Braeckel進行帶領AWOS資料顯示介

面的設定與調整，其主要撰寫程式爲Java軟體。而負責授課的Aaron Braeckel已具有十多年相關軟體的撰寫經驗。目前航空氣象現代化作業系統的新一代多元產品顯示系統(JMDS)及自動天氣觀測系統(AWOS)資料顯示介面皆爲其作品。這部分的課程同樣牽涉到Java原始碼的調整，同樣的我們依然遇到了Java程式撰寫環境的問題，不過在Aaron協助下我們很快的調整了訓練主機相關參數。我們也在此課程中複習了AWOS資料顯示介面的運作原理及系統架構。最後於今日下課之前，參訓學員皆順利將自己於課程中所負責之機場資料，顯示於調整後的資料顯示介面中。

10月14日星期五，課程重點在於整合昨日所進行之顯示介面部分。有了昨日的經驗，我們順利的將原AWOS資料顯示的四個機場外，成功新增了馬祖北竿、臺東豐年、綠島及蘭嶼機場另外四個機場。另外今年本總臺氣象中心曾經向NCAR所提出關於積冰預報產品(FIP)的問題，又剛好至此我們本次訓練的課程已逐漸邁入尾聲，NCAR於今日下午請對於積冰有相當研究的科學家Cory Wolff，爲我們解釋FIP的演算原理與解答該項產品於實務上的運用方式，覺得受益匪淺。

10月15日、16日爲週休，未安排課程。

10月17日星期一，由NCAR人員Arnaud Dumont說明，新一代航空氣象多元產品顯示系統(JMDS)於明年度將進行系統升級，過去該系統各種設定參數皆寫入系統原始碼之中，所以在維護上相當不容易。未來將採用Jadeite的工作平臺持續開發本系統，而JMDS將可經由維護人員容易維護調整的控制檔案加以設定，將有利於未來系統維護及因應未來使用者的需求而進行的調整。當天下午我們進行實機操作，學習如何經由系統設定檔將以Jadeite爲開發平臺之JMDS，目標在將其調整成與現在提供服務用的JMDS一致。

10月18日星期二，我們繼續昨天的課程，繼續學習如何利用設定檔調整未來新架構的JMDS。下午與Gary Cunning及Jim Cowie進行本次訓練課程的簡短討論後，我們將最後完整版本的AWOS資料顯示介面資料程序調整及開發手冊交給NCAR進行後續檢查與翻譯。另外我們由Gary Cunning及Jim Cowie的手中接到本次訓練的結業證書。本次爲期三週的訓練也在此畫下句點。

10月19日星期三，我們於住處整理行李後，前往Celia家中稍事休息後，踏上回臺灣的路程，最後在21日清晨6點抵達臺灣桃園國際機場。

參、 研習內容

一、<u>AOAWS系統結構與資料流</u>：

首先，在進行主要課程前，照慣例的由NCAR人員Gary Cunning爲我們複習AOAWS系統及其資料處理流程。

AOAWS系統由NCAR的RAL實驗室所發展，目前最新所使用的作業系統平臺爲Debian Linux lenny版本，帳號管理部分則使用C-shell來編寫。而系統本身可分成處理程序以及系統管理兩大區塊：

AOAWS系統處理程序可分成兩類：

1.Data-Driven爲資料導向的設計：Data-driven的設計使得伺服器在接收到新資料時就會自動傳送到目標伺服器，當有新資料時就會自動傳送的動作稱爲process，必須預先設定好，若平時沒有新資料時process就會呈休息狀態，避免系統的資源過度消耗。

2.Schedule-Driven爲功能導向的設計：Schedule-Driven的功能則是利用cron的功能監控新資料是否適時且正確的傳送到目標伺服器，cron是一種定時執行的程式，當新資料延遲一定的時間時，就可以使圖框變色以利監控，目前都已經整合在網頁上。

而Application(應用程序)可以用來扮演上述兩種設計間資料傳輸的橋樑，例如LdataWatcher、InputWatcher、LdataWriter等，安插在適當的資料流程之中，扮演資料傳輸及紀錄資料抵達狀態的工作。

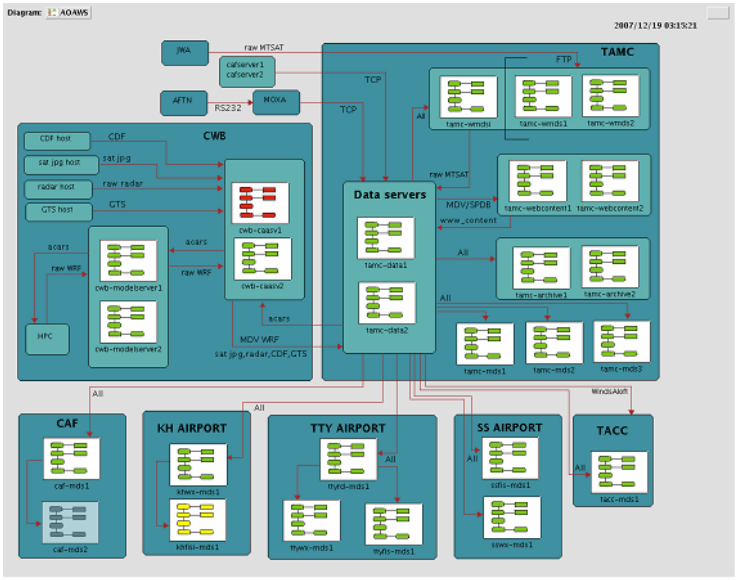


圖1 AOAWS系統監控畫面

而整體AOAWS系統又分成系統程序管理與資料管理兩類。程序管理主要在於管理系統程序之運作，系統若發現程序異常或終止，系統會自動清除異常程序並重新啓動程序，以確保程序正常。而資料管理部分，利用各資料目錄中之紀錄檔紀錄資料最後抵達目錄的時間，並設定資料合理到達時間間距，作爲監控門檻，當系統發現資料抵達時間已超過合理時間門檻時，將透過系統監控畫面通知系統管理人員，由人員追查並排除資料異常情況。

位於氣象中心AOAWS的主機群，依角色的不同可分爲7類，分別爲:

1.Data主機：包括data1、data2。

主要工作爲處理來自各資料接收主機所送來之資料，由於各原始資料種類繁多，必須經過資料轉化程序進行轉化，將各種資料轉化成單一格點的SPDB資料格式或涵蓋大範圍網點的MDV資料格式,並將轉化完成資料送至各主機進行後續處理。以下爲Data主機接收及傳送資料之對象：

接收資料對象：

1. 向中央氣象局(CWB)的CAA SERVER接收衛星圖檔、雷達、CDF(日本氣象廳傳真天氣圖)等。
2. 透過網路接收來自AFTN(現爲AMHS系統)資料。
3. 透過網路接收空軍資料，如軍用機場METAR、SPECI、TAF及探空資料。
4. 接收WMDS SERVER接收之如日本衛星資料、WAFS(世界區域預報系統等資料。
5. 接收WEBCONTENT主機資料，包括航空氣象服務網中網頁圖形資料等。
6. 接收松山、桃園、高雄、金門航空氣象臺的AWOS資料。

傳送資料對象：

1. 向CAA SERVER傳送ACARS資料。
2. 向WMDS主機傳送航空氣象服務網中網頁圖形資料及MDV、SPDB資料。
3. 向WEBCONTEN主機傳送MDV及SPDB資料，供其製作網頁圖形資料。
4. 向ARCHIVE主機傳送資料，供其進行資料備份。
5. 向MDS主機傳送資料，供資料顯示並由使用者查詢之用。

2.Wmds主機：包括wmdsi、wmds1、wmds2。

1. 接收資料功能： wmds1及wmds2分別接收來自JWA(日本氣象協會)及NOAA(美國海軍氣象中心)的MTSAT衛星資料及WAFS越洋航線顯著天氣圖並傳送給data servers。
2. 網頁伺服器功能：航空氣象服務網就是設置在wmds server，wmdsi爲內網

網頁伺服器，僅內部電腦可以進入，wmds1,wmds2爲外網網頁伺服器，提供給內部以外的使用者。

3. Webcontent主機：包括webcontent1、webcontent2。

   接收MDV,SPDB資料，將資料轉成圖檔，傳送回Data主機。最後再由Data主機轉送給Wmds主機。

4. Archive主機：包括archive1、archive2。

   備份AOAWS中包含網頁圖形資料、MDV、SPDB及原始資料，供日後查詢使用。

5. Mds主機：包括氣象中心、松山、桃園及高雄之氣象臺與諮詢臺、區管中心共11部主機。

   爲資料顯示用之工作站，可供使用者查詢所需之資料。

6. Caa Server主機：包含caasev1、caasv2。

   主要工作在於扮演中央氣象局與民航局資料傳輸的跳板，中央氣象局將衛星圖檔資料、雷達資料、全球通訊資料(GTS)、CDF資料及由Modelserver主機傳來經過進一步演算所得之資料(如積冰、亂流預報)傳輸至Data主機。

7. Model Server主機：包含modelserver1、modelserver2。

   主要工作在於接收中央氣象局氣象數值預報模式資料，並納入積冰亂流資料演算法進行運算。

二、AOAWS系統流程控制與偵錯：

AOAWS系統執行程序主要都在/home/aoaws/projDir目錄下，包含系統控制目錄(system)、資料目錄(data)和記錄檔目錄(logs)以及其他目錄等。

(1)系統目錄 (system)：此目錄中包含所有系統控制程序和參數檔。

(2)資料目錄：

此目錄連結到/d1/aoaws/data/目錄，AOAWS系統資料以結構性和階層式分類，包含raw、mdv及spdb等主要目錄，各目錄下再依照資料種類區分，資料目錄內容會因主機的特性不同而不同。

AOAWS系統資料的傳遞採完全自動化的方式，系統執行LdataWatcher程序隨時監控各資料目錄新資料抵達，以採取相對應的動作。LdataWriter、DsCopyServer或是其他應用程式將資料傳入時，會同時更新_latest_data_info和_latest_data_info.xml等檔案，紀錄最新一筆資料資訊以供系統監控使用。

在判斷資料是否該傳輸的時候，其資料傳遞對象則主要根據/d1/aoaws/data/_distHostList檔案中所設定的主機傳送。如果針對特定資料要傳送部份特定主機則遵照該特定資料目錄中之_DsFileDist設定檔傳遞資料。參數設定檔中明確的說明使用時機，設定之修改相當容易。

系統爲避免資料量無限制成長，在資料目錄(/d1/aoaws/data)下設有_Janitor，控制資料儲存時間，當資料儲存時間長於最大保存天數(MaxNoModDays)時，系統便會自動將該資料刪除。每個下層資料目錄可以設定個別的_Janitor設定檔，如果沒有設定時則以上層資料夾設定爲依據。

AOAWS系統結構可藉由流程圖表現資料的傳輸或程式的控制，藉圖樣的顏色辨別流程是否停滯。透過這樣的結構圖，系統監控人員便可以在最短的時間內找出問題（如圖2）。

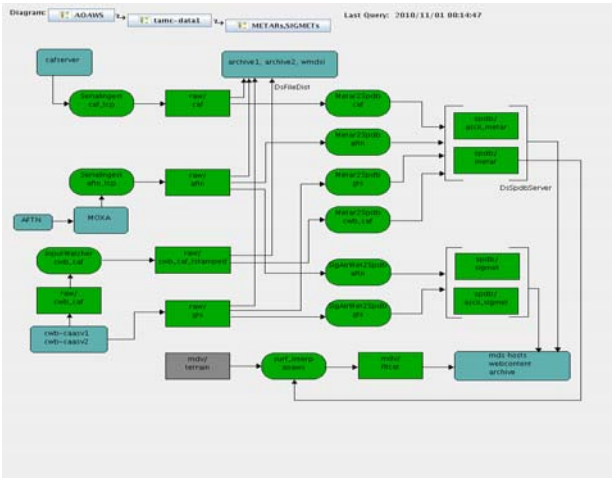


圖2 AOAWS系統結構流程圖

圖3內方塊爲資料類別，橢圓爲程式類別，由箭頭串聯各色塊代表運作過程，綠色代表正常，黃色甚至紅色就是故障的情況；若發現故障再點選方塊就可得知資料所在路徑及故障時間。

在系統結構方面，氣象中心分爲data server等伺服器，這樣的分類法是以功能爲區分，以data type爲分類方式，每個data type都有不同的功能，從事不同的程序(process)，輸入不同的資料。可以說不同的data type間一定有process作控制，但是大部分的process在data type下的host間做控制的工作。在$proj_DIR/下有三個process做控制工作，有control、params、scripts等檔案，稱爲roles，control role負責每個host處理process所產生的log files，params role紀錄了每次處理process所使用的參數，而scripts role則代表控制control process的scripts(scripts就是將許多指令寫在同一個script檔裡面，只要執行script檔就能夠完成所有的設定的流程)，scripts中紀錄了所有process names、parameters等紀錄。

一個完整的process是由input data、instances、control condition、crontab、output data等所組成，這幾項元素可分別詳述如下:

Input data(or output data):輸入或輸出資料，分別代表process的原料與產品。

Instance:process的參數，每個process只能有一個instance值。

Control condition:條件判斷，在instance參數值符合某種條件時，process將被執行。

Crontab:控制process的程式，用以檢查process未執行的閒置時間，判斷process是否發生錯誤。

在此以auto_restart(爲host間的application)監控process爲例，以perl script寫成，控制procmap及proc_list中的數個process，當auto_restart發現有process未正確執行時，auto_restart中的指令開始使其控制的所有process重新執行一遍，這便是本監控process的執行流程（如圖3）。

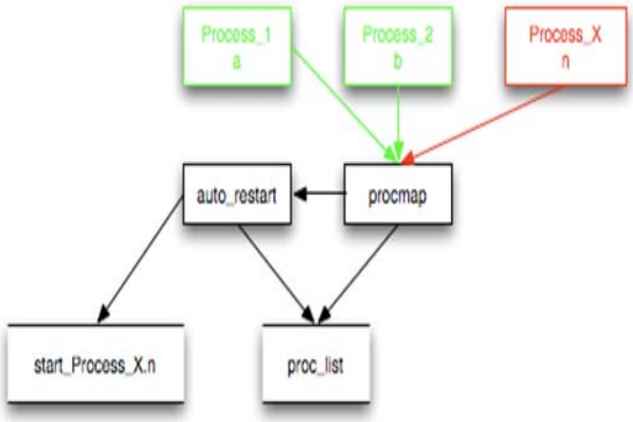


圖3 監控process執行流程

另以SerialIngest的application爲例，SerialIngest可監控各機場的AWOS資料經由區臺的MOXA輸入到AWOS顯示系統之內的過程是否出了問題（如圖4）。

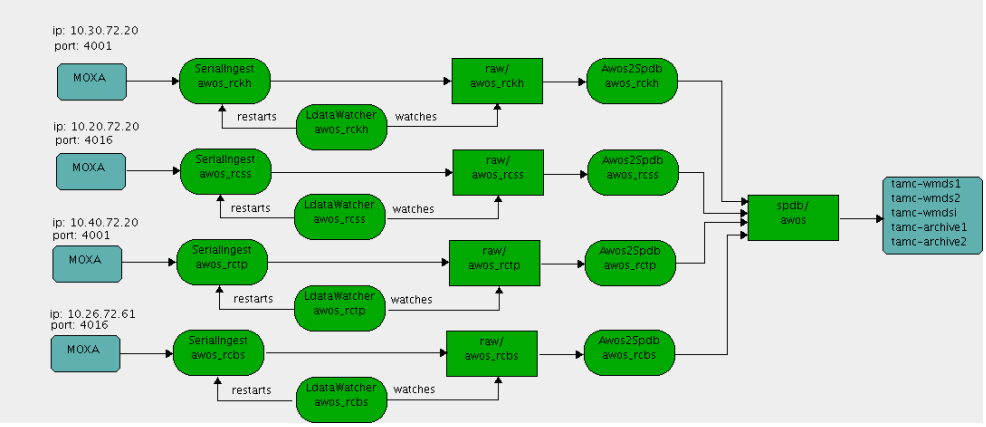


圖4 AWOS資料流監控

三、**AWOS DISPLAY顯示介面及擷取資料方式**：

AWOS　DISPLAY類似JMDS，是JAVA程式編寫成的一個顯示介面，將各氣象臺的AWOS即時資料透過航空氣象服務網網頁，讓其他使用者能在家上網就能了解目前各機場即時天氣資料狀況。

圖5爲實際的松山機場AWOS DISPLAY顯示介面，目前最新的AWOS DISPLAY爲第10.0版本，軟體設定檔爲XML格式，與HTML類似但較爲複雜；平臺介面包括標籤、工具列、箭頭樣式等則利用JAVA SWING來編寫，所使用的程式語言則爲JAVA 6，SWING所提供的平臺使得AWOS DISPLAY在WINDOWS、LINUX、MAC作業系統都能相容。

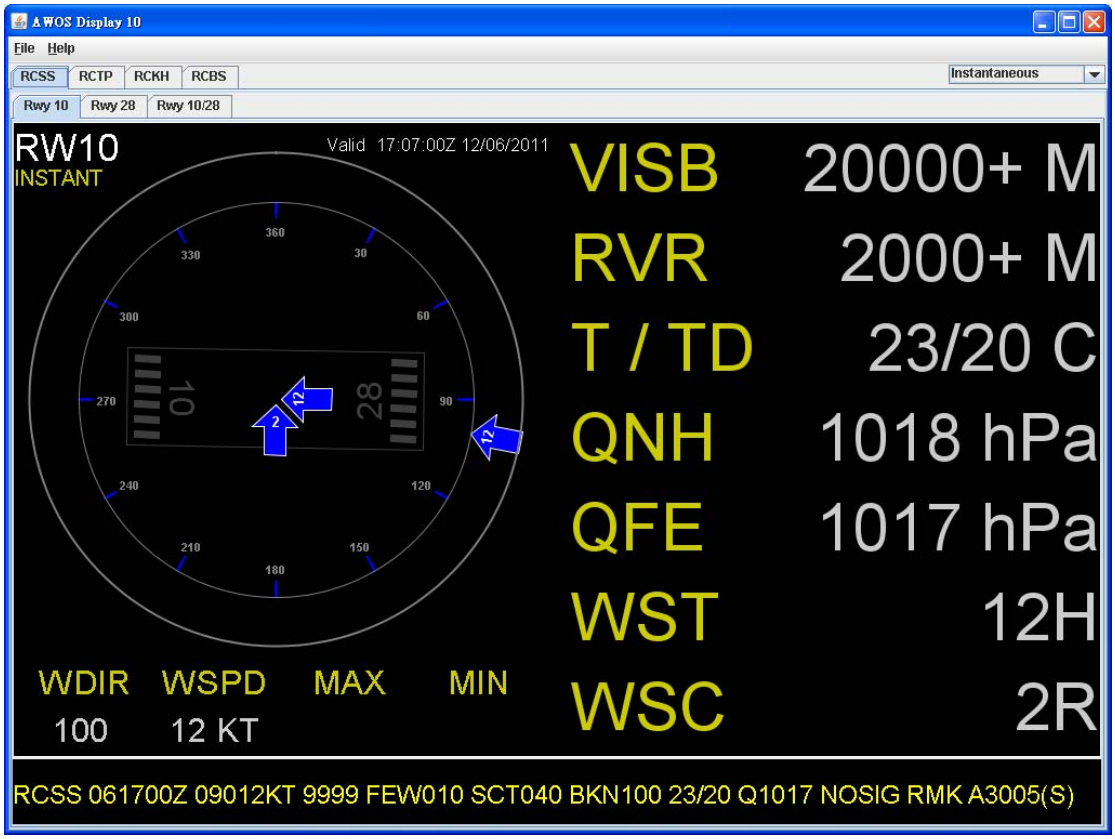


圖5 AWOS顯示介面

SWING 提供一個介面 ToolBarBotton 及 ToolBarCheckBox 的設計，ToolBarBotton下分爲三個部份Panel、SelectTab及TextArea，可修改及新增標籤的名稱、select button、check box等等（如圖6）。同時也介紹了利用SWING來設計AWOS DISPLAY的優點，如佔用比較少的資源、可隨程式設計者任意修改及變換組態、可依設計需要自行增減工具的元件等。像局屬10個民航機場有不同的名稱，每個機場有不同方向的跑道，機場的跑道數量有一條或兩條，下拉式選單可選擇即時風向、兩分鐘平均風、十分鐘平均風等，由此可以看出SWING這樣的

開發工具的確功能相當強大、讓設計者能很自由地依照機場的特性來撰寫需要的介面。

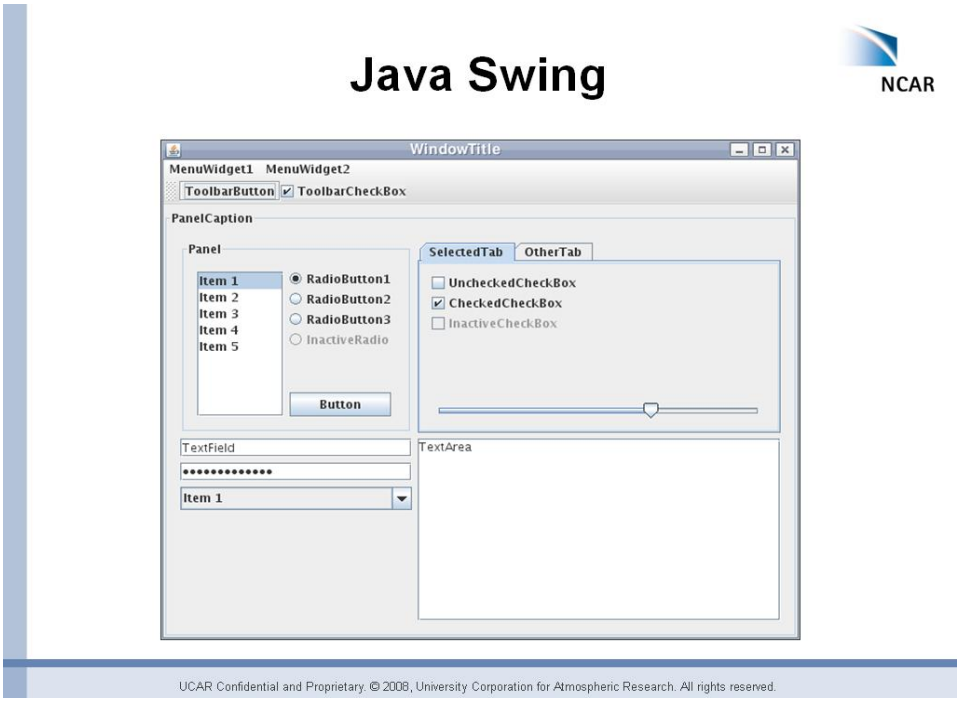


圖6　JAVA SWING介面

以JAVA語言設計出顯示平臺後還需要實際抓取氣象臺的AWOS資料，透過JAVA WEB START程序，將以JAVA撰寫完成的資料顯示介面呈現在使用者之電腦螢幕上。而系統在資料擷取方面，AWOS DISPLAY則採用DATA FINDER、DATA RETRIVER的方式向各機場的每個跑道分開擷取資料，而後在顯示介面中加以整合，大致流程如圖7及圖8所示:

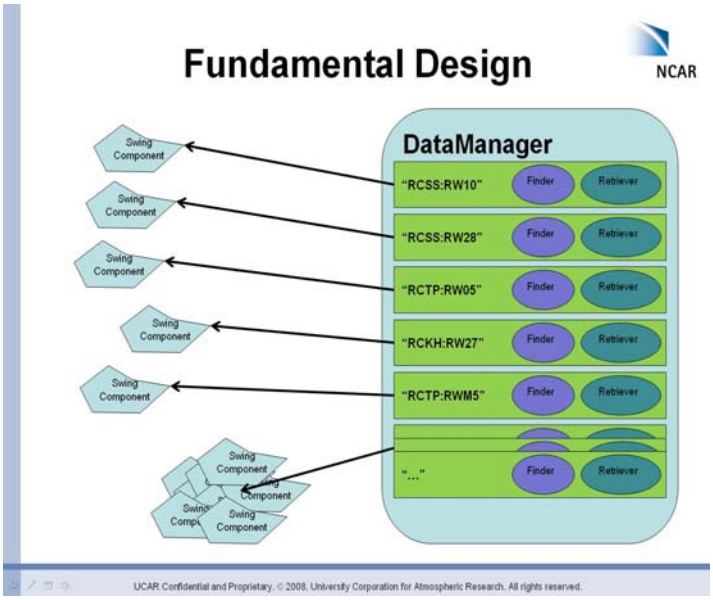


圖7 AWOS系統擷取資料方式

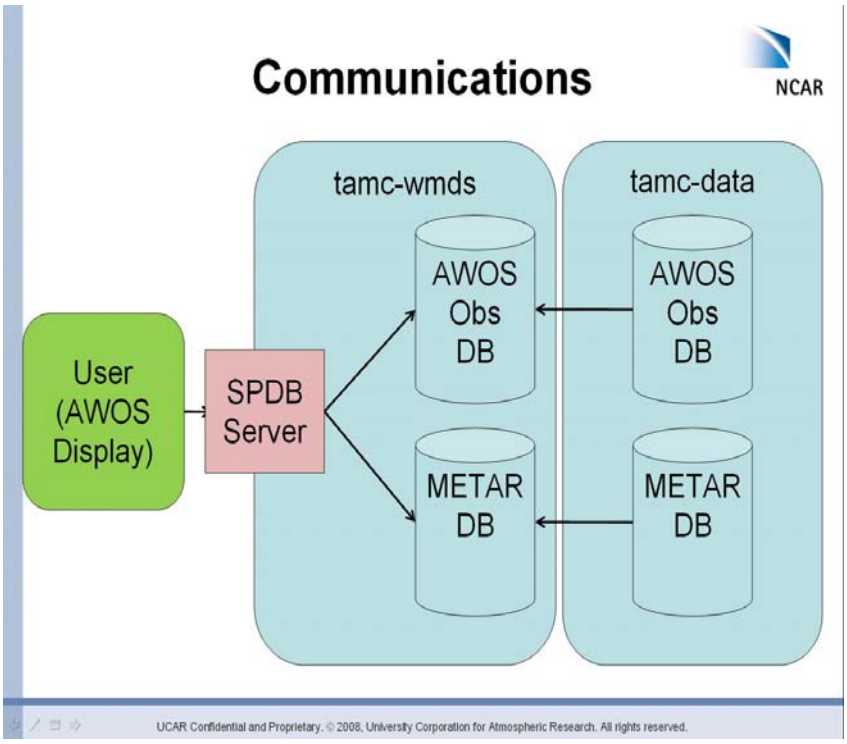


圖8 AWOS DISPLAY資料架構

四、**AWOS資料顯示系統之實際操作**：

本課程重點在於AWOS資料顯示介面的設計與調整，以因應未來民航局所屬機場之AWOS資料納入AOAWS後，系統維護人員可自行進行設定資料接收與轉化，最後於航空氣象服務網內顯示，供使用者使用。

AWOS資料顯示介面共分成以下三部分：

1.資料接收：主要在設定當AWOS資料經由各機場當地之MOXA或由IP廣播方式輸出後，如何設定接收程式與其參數，使AOAWS順利將資料進行存取。

2.資料轉化：主要在於如何調整及增加原本程序讀取新增之AWOS資料的能力。

3.資料顯示：主要在於如何經由調整AWOS資料顯示介面，使其可顯示轉化完成後的資料。

以下依此順序說明：

(一) 資料接收：

目前松山、桃園、高雄及金門機場的MOXA輸出當地AWOS資料，AOAWS經由SerialIngest程序進行接收，且個別機場有專屬的接收程序。

此次參訓學員分別安排設定馬祖北竿(RCMT)、臺東豐年(RCFN)、綠島(RCGI)及蘭嶼(RCLY)四個機場之AWOS資料進行模擬訓練。

因上述四個機場目前皆無真實資料可用，故使用松山機場之資料進行模擬。以馬祖北竿機場爲例，首先將松山機場之資料，利用以下程式碼進行機場代碼及跑道名稱的調整，以符合實際機場狀況。松山機場代碼爲RCSS，應修改爲馬祖北竿機場之代碼RCMT；松山機場跑道名稱爲10-28跑道，應修改爲馬祖北竿機場之03-21跑道。程式如下：

```
#!/bin/sh
#
files=`ls *awos_rcss`
for name in $files ;do
    new_name=`echo $name | sed ‘s/rcss/rcmt/g’`
    sed ‘s/RCSS/RCMT/g’ ./$name | ‘s/AWS_10/AWS_03/g’ | sed ‘s/AWS_28/AWS_21/g’ > $new_name
    /bin/rm –f ./$name
done
exit 0
```

完成後，即可產生訓練用之馬祖北竿機場之 AWOS 資料。並將其存入 NCAR

所設定的資料發送主機的特定資料夾中，該主機主要是模擬成各地機場所架設 MOXA，用以負責輸出訓練用的 AWOS 資料。接著設定資料發送主機中 ~projDir/awos_ingest/params/之 file_repeat_day.awos_rcmt 參數檔，使其將模擬資料經由 start_TwnFiles2TCP.awos_rcmt 程序連續性的經由特定的網路連接阜發送資料，此情況與實際 MOXA 類似，唯一不同的是 MOXA 所輸出的是真實的 AWOS 資料，而 fie_repeat_day.awos_rcmt 輸出的是虛擬資料。

在製造出訓練用之虛擬資料後，將其程序寫入~projDir/control/proc_list 及 ~projDir/awos_ingest/control/proc_list 中，以確保資料發送主機在重開機後，依然可連續性的輸出訓練用資料。

接著設定接收主機(在實際 AOAWS 作業環境中爲 Data 主機)之相關接收程式。可利用以下指令得到目前接收程式的參數檔情況：

SerialIngest –print_params > SerialIngest.test

SerialIngest 的參數說明，如圖 9：

debug	DEBUG_OFF, DEBUG_NORM, DEBUG_VERBOSE
instance	Used for registration with procmap
connection	SERIAL or TCP
input_device	name of device driver for serial port, /dev/ttyS1
baud_rate	Baud rate for incoming serial data port
dataIs7Bit	Set TRUE for 7-bit data, FALSE for 8-bit data.
twoStopBits	If TRUE, 2 stop bits. If FALSE, 1 stop bit.
enableParity	TRUE or FALSE
oddParity	TRUE or FALSE
tcp_server_host_name	Name of TCP server host
tcp_server_port	TCP server port number.
send_tcp_handshake	Option to send TCP handshake sequence to the server to trigger the data flow.
tcp_handshake_bytes	List of bytes to be sent to server for handshaking.
filter_ctrlm	TRUE or FALSE
end_of_message_check	Option to check for end of message before closing an output file.
output_interval	Interval at which output files are created (secs).
force_output_if_stalled	Flag for forcing the file output if the input stream in stalled.
discard_zero_length_file	Flag for discarding 0-length files.
output_dir_path	Name of output directory.
output_file_ext	Extension for output file.

圖 9 SerialIngest 參數說明

了解參數檔中之資料接收網路 IP 位址及存取路徑後，新增馬祖北竿機場專屬的接收參數 SerialIngest.awos_rcmt，同時也於~prjDir/awos/scripts 中新增 start_SerialIngest.awos_rcmt 程序。完成後即可啓動程序，並檢查資料存取目錄 ~peojDir/data/raw/rcmt/是否已成功將資料存入。如果正常，則將程序寫入 ~projDir/control/proc_list 及 ~projDir/awos/control/proc_list 中。另外在

SerialInget.awos_rcmt 與 ~projDir/data/raw/awos_rcmt 資料夾之間，設有 LdataWatcher，負責監看資料是否持續存入資料夾中。若資料出現中斷情況，則 LdataWatcher 會重新啓動 SerialIngest 的程序，確保資料不會因爲程序問題而中斷。而同樣的，我們在了解其運作原理後，新增了專屬於個人負責機場 AWOS 資料的 LdataWatcher.awso_rcmt(以馬祖北竿爲例)。在完成之後，同樣的將其寫入 ~projDir/control/proc_list 及~projDir/awos/control/proc_list 中。

(二) 資料轉化

在確定資料已經正常接收，並且設定完成資料監控的程序後。緊接著就要進行資料轉化的部分。但這部分開始與調整原始碼有關，而 NCAR 已將 AWOS 資料轉化的原始碼存放的位置爲各 AOAWS 主機中之 /d1/aoaws/build/cvs/apps/aoaws/src/。而負責將原始資料轉成 spdb 格式的程序爲 Awos2Spdb，圖 10 爲其程式架構：

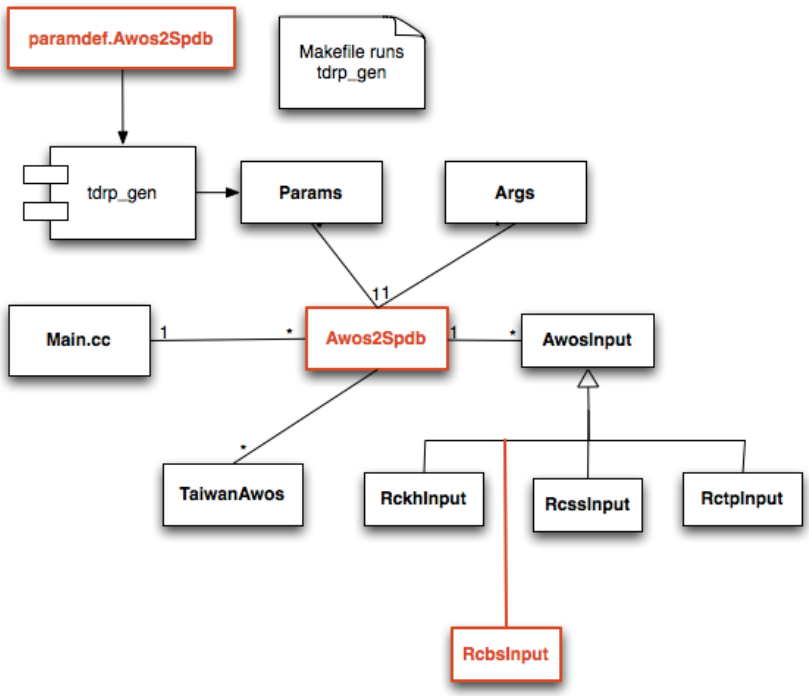


圖 10 Awos2Spdb 程式架構

而本部分我們將調整圖 10 紅色部分。必須調整的檔案共有五個，分別爲：

1. RcmtInput.cc
2. RcmtInput.hh
3. Awos2Spdb.cc (新增 RCMT 相關資訊)
4. Awos2Spdb.hh (新增 RCMT 相關資訊)
5. paramdef.Awos2Spdb (新增 RCMT 相關資訊)

由於有關 Awos2Spdb 程序爲 C++語言撰寫，所以在上述檔案調整完畢後，必須進行編譯(compile)的動作。在編譯過程中，同時也可以知道撰寫的部分是否存在錯誤，如果發現錯誤則必須調整後，重新執行 compile。與之前說明 SerialIngest 一樣，所有的 AOAWS 程序都有其設定檔，而 Awos2pdb 的設定檔可利用./Awos2spdb –print_params >> Awos2Spdb_params.rcmt 輸出。輸出後的檔案，同樣需要將其調整，使其與新增機場的 AWOS 資訊一致才行。

在完成調整部分後，可執行以下指令：

./Awos2spdb –params Awos2Spdb_params.rcmt –f 原始檔案位置

先前調整完成的程序正常，其會順利的將新增 AWOS 資料轉成系統可用的 SPDB 檔案格式。另外亦可執行以下指令，用以查詢 SPDB 資料中是否已有新增 AWOS 資料，指令如下：

./SpdbQuery –url SPDB檔案目錄 –mode latest

當然這部分也寫入~projDir/control/proc_list及~projDir/awos/control/proc_list中。

到此階段，已算是將AWOS資料接收及資料轉化進行完畢，在我們的訓練主機上，已成功的接收由資料發送主機傳送的資料，並且也成功的將資料轉成系統所需的SPDB格式。

進入資料顯示之前，另外還有一件事情需要進行調整。由於在AOAWS中各系統程序與資料狀態皆會被系統監控程式(Sysview)所管理。所以在我們完成四個機場的AWOS資料接收及轉化程序後，必須在Sysview中進行設定。這部分相對簡單許多，只要在資料主機上的~pojDir/sysview/script/ 資料夾中，執行./stat_Sysview.edit即可啓動Sysview的編輯畫面，只要將資料接收流程及資料對應目錄，以現成的程式繪製工具加在目前的Sysview中即可。

(三) 資料顯示

AWOS 資料顯示介面本身由 Java 寫成的原始碼程式(.jar files)、JNLP 檔、XML 檔三種檔案組成。jar file 是由 Java 語言寫成，當 AWOS 資料顯示介面執行時，jar file 會開始由檔案中的路徑收集所有 AWOS 資料資訊並由系統進行整合，最後由介面顯示程式讀取 XML 設定檔。當 Client 端與主機連線時，Client 端的瀏覽器如果可以支援 Java 平臺，JNLP 檔會經由 Java Web Start 與 Web Brower 機制開啓並檢查是否有需要更新的檔案並予更新，jar files 則經由 HTTP 協定傳輸至 Client 端電腦。所以 Client 端的電腦可看見 AWOS 資料顯示介面。目前最新的 Java Web Start 需要使用者安裝 Java 6.0 以上版本的 Java 執行環境(Java Runtime Enviorment)，才可順利執行 AWOS 資料顯示介面。AWOS 資料顯示介面所使用的 Java Web Start 原始碼如下圖：

```
<?xml version="1.0" encoding="utf-8"?>
<jnlp
  spec="1.0+"
  codebase="http://aoaws.caa.gov.tw/wmds/content/aoaws/awos/">

  <information>
    <title>Experimental AWOS Display</title>
    <vendor>National Center for Atmospheric Research (NCAR) Research Applications Laboratory (RAL)</vendor>
    <description>AWOS Display Version 8, an experimental component of the AOAWS system</description>

    <homepage href="http://aoaws.caa.gov.tw/wmds/content/aoaws/awos/"/>

  </information>
  <security>
     <all-permissions/>
  </security>
  <!-- Always require an update.  If this is not done, then the authentication screen times out before new versions can be downloaded -->
  <update check="always" policy="always"/>
  <resources>
    <j2se version="1.6+" max-heap-size="128M"/>

    <property name="sun.net.client.defaultConnectTimeout" value="20000"/>
    <property name="sun.net.client.defaultReadTimeout" value="20000"/>

    <jar href="awos.jar"/>
    <jar href="commons-digester-1.8.jar"/>
    <jar href="commons-beanutils-1.8.0.jar"/>
    <jar href="commons-logging-1.0.4.jar"/>
    <jar href="conf.jar"/>
  </resources>

  <application-desc main-class="edu.ucar.rap.aoaws.apps.awos.AwosDisplay">
     <argument>-config_file</argument>
     <argument>edu/ucar/rap/aoaws/apps/awos/config.taiwan.xml</argument>
  </application-desc>
</jnlp>
```

圖 11 AWOS 資料顯示介面之 Java Web Start

這部分的學習重點在於如何設定顯示介面，使其得以顯示訓練課程中所新增的機場 AWOS 資料。而學習設定控制 AWOS 資料顯示介面的 XML 格式檔案成爲這部分之重點。

在 AWOS 資料顯示介面之 XML 將 AWOS 資料顯示介面分成三個群組，分別爲：

1.機場名稱(Airports)：用以標示機場名稱。

2.AWOS 群組(AWOS GROUP)：用以標示機場之各跑道名稱。

3.AWOS：用以顯示機場之跑道資料。

詳細如下圖所示：

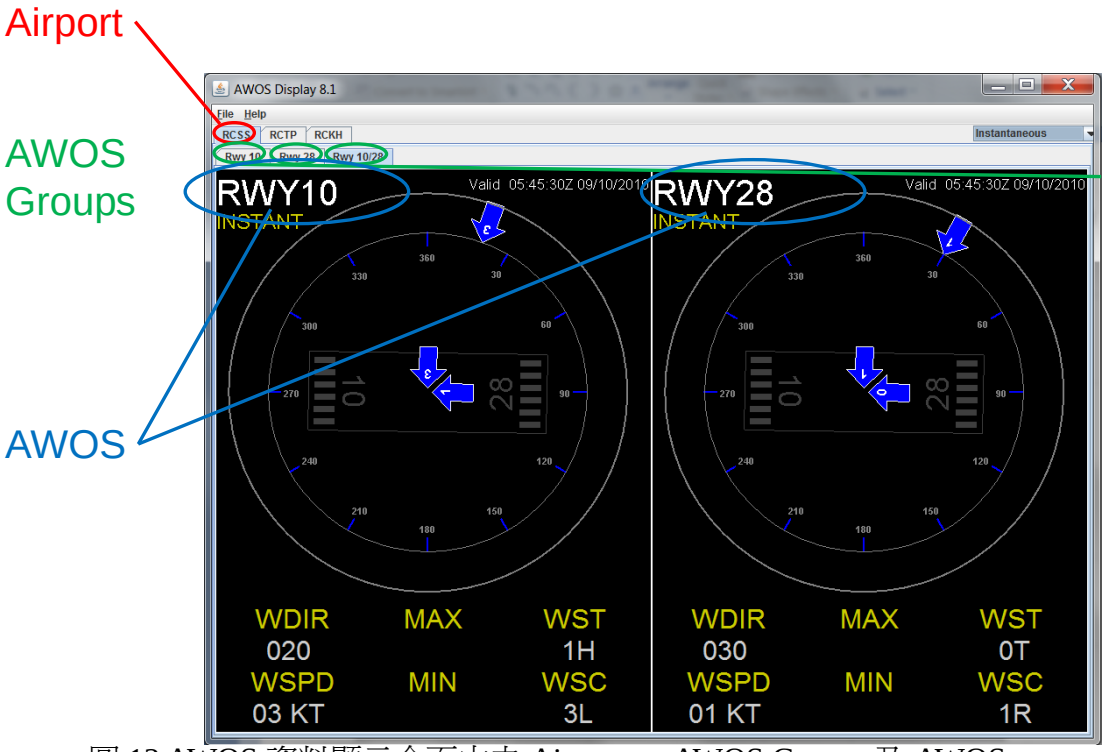


圖 12 AWOS 資料顯示介面中之 Airport、 AWOS Groups 及 AWOS

而其控制之 XML 原始碼如下：

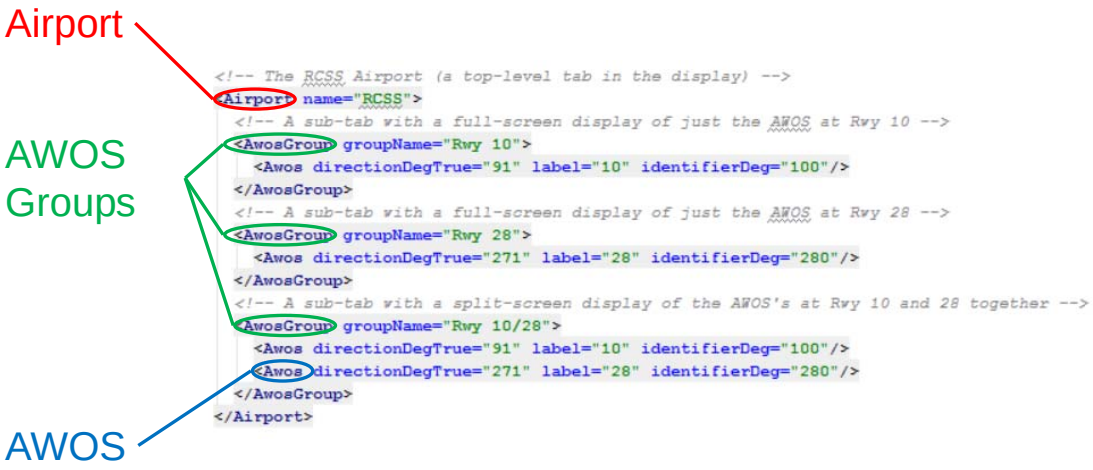


圖 13 AWOS 資料顯示介面各群組的原始碼

NCAR 將 AWOS 資料顯示介面的原始碼放在 /d1/aoaws/build/cvs/java/src/edu/ucar/rap/aoaws/apps/awos，在調整設計 AWOS 顯示介面部分主要調整的檔案爲 config.xml。我們將新增的機場資訊加入前述檔案之中後，就可以進行 Java 編譯。依序執行 ant clearn；ant jar 以及 ant dist(這步驟需要輸入 Java 憑證)。如果發現錯誤，則需要進行檔案檢查及環境變數檢查，編

譯完成後。於/d1/aoaws/build/cvs/java/dist/signed 執行
java -cp
awos.jar:commons-beanutils-1.8.0.jar:commons-digester-1.8.jar:commons-logging-1.0.4.jar:conf.jar edu.ucar.rap.aoaws.apps.awos.AwosDisplay -config_file edu/ucar/rap/aoaws/apps/awos/config.xml，測試新增 AWSO 資料機場後的顯示畫面。如圖 14：

圖 14 新增馬祖北竿機場後的 AWOS 資料顯示畫面

五、**Jadeite架構之JMDS**：

目前航空氣象服務網之JMDS，爲民航局飛航服務總臺臺北航空氣象中心提供航空氣象資料的主力系統。其具有以下幾個特點：

1. 不必受使用者作業系統的限制：無論使用者所使用的作業系統爲Windows、Linux或麥金塔系統，只需要事先安裝完成Java執行環境後，就可以經過航空氣象服務網之連結，啓動JMDS。
2. 打破空間的限制：過去MDS系統只能執行於特定的資料顯示主機，這些主機位於松山、桃園及高雄機場之氣象臺與諮詢臺，區域管制中心以及氣象中心等地。所以使用者必須前往上述位置進行資料查詢。而JMDS只要使用者可透過手邊電腦，經過網際網路執行JMDS，就可以取得航空氣象資訊。
3. 與使用者有高度的互動功能：JMDS利用Java本身可與使用者產生高度互動的特性，設計許多與使用者保持互動的功能，透過這些高度的互動功能，讓使用者更快速取得所需要的資訊，而不是以往面對冷冰冰的資料顯示介面。

但在明(101)年度航空氣象現代化作業系統氣象技術增強計畫第15號執行辦法，將開始著手更新JMDS。更新的JMDS以Jadeite的Java Framework打造，其將具有以下幾個優點：

1. 維護更爲便利：由於過去的JMDS主要以Jade打造，各項功能是直接寫入系統之中。所以在維護上，必須面臨調整撰寫JMDS原始碼的情況。而JMDS由於本身功能強大，所以其原始碼部分已經有11MB的容量，程式碼更高達十萬行上下。所以再維護上往往是牽一髮而動全身，必須特別小心每個環節。而Jadeite打造的JMDS可利用設定檔的方式，將經常性需要調整的部分寫入設定檔中，如此可有效的減少程式碼的大小，便於日後維護。
2. 可依使用者不同的需求調整：目前的JMDS常有使用者要求微調部分顯示功能，如色階、範圍及區域等等。但這往往爲少數使用者的特殊需求，並非多數使用者所希望。但在Jadeite打造的JMDS架構下，使用者可經由調整個人所屬的XML控制檔，使得JMDS變成可以因個人需求而調整的航空氣象多元產品顯示介面。

Jade與Jadeite的發展歷史如下圖：

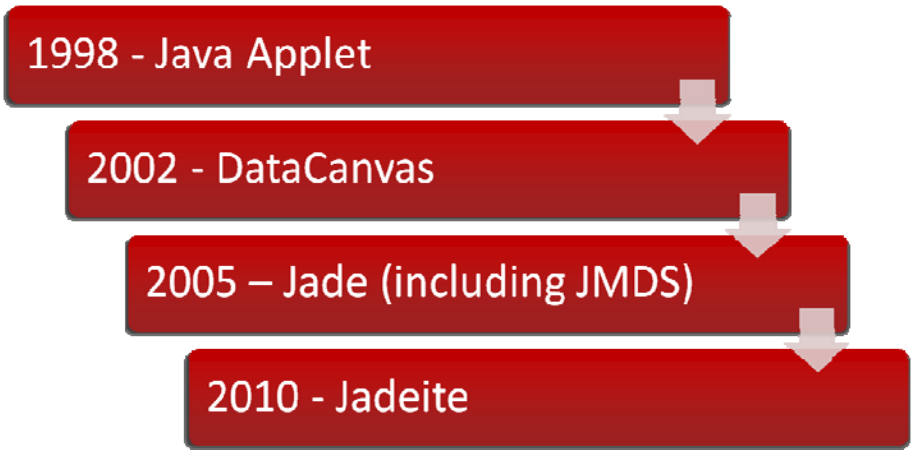


圖15 Java應用程式發展時間

由上圖可知，JMDS於2006年正式上線，當時最新的Java應用程式技術爲Jade。而相關應用程式發展至今，於2010年已經發展至Jadeite。

而以下是目前與未來JMDS的架構圖：

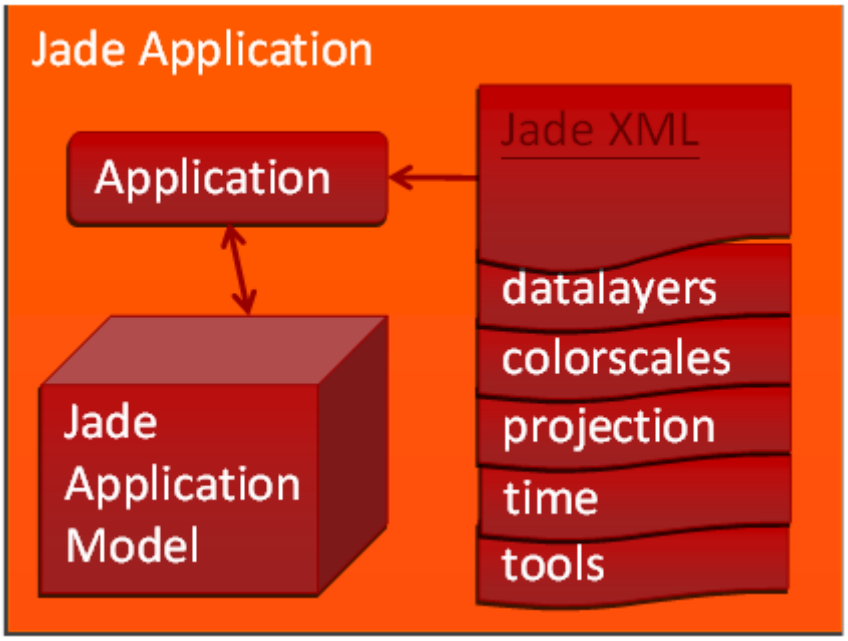


圖16 目前的JMDS

由上圖得知，所有控制Application的XML是被寫入Jade Application的，所以如前面所述，這將造成維護與管理的困難。

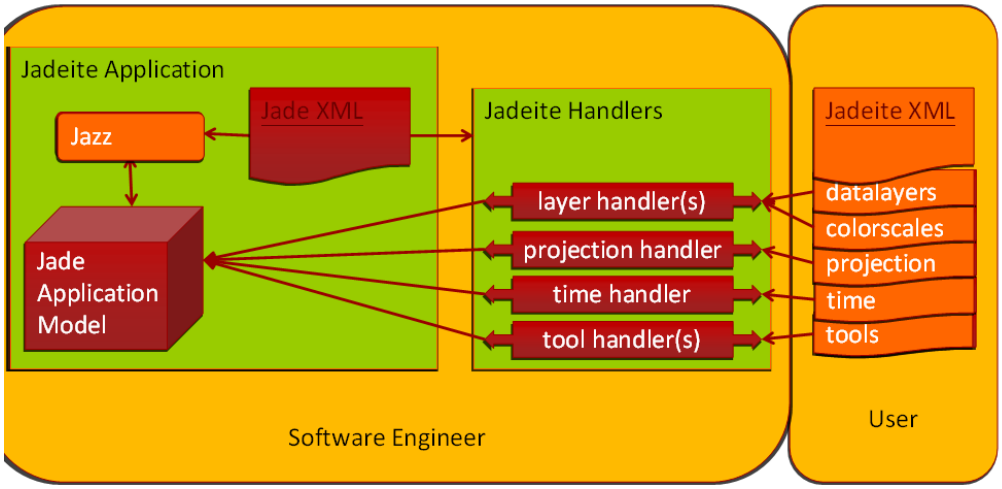


圖17 未來以Jadeite打造的JMDS架構

由圖17可知，未來的JMDS將分成兩個部分，使用者可利用Jadeite XML控制JMDS，以達到個別使用者之需求，而系統維護人員只需要維護系統核心部分即可。不過由於JMDS部分功能相當特殊，我們在訓練主機上進行以Jadeite XML調整未來的JMDS時，仍遇到一些問題。不過在詢問NCAR的授課人員後，了解因爲Jadeite仍有部分需要調整，目前尚無法支援全部已使用於當前JMDS的系統物件。這部分已由NCAR的程式工程師持續進行研發。

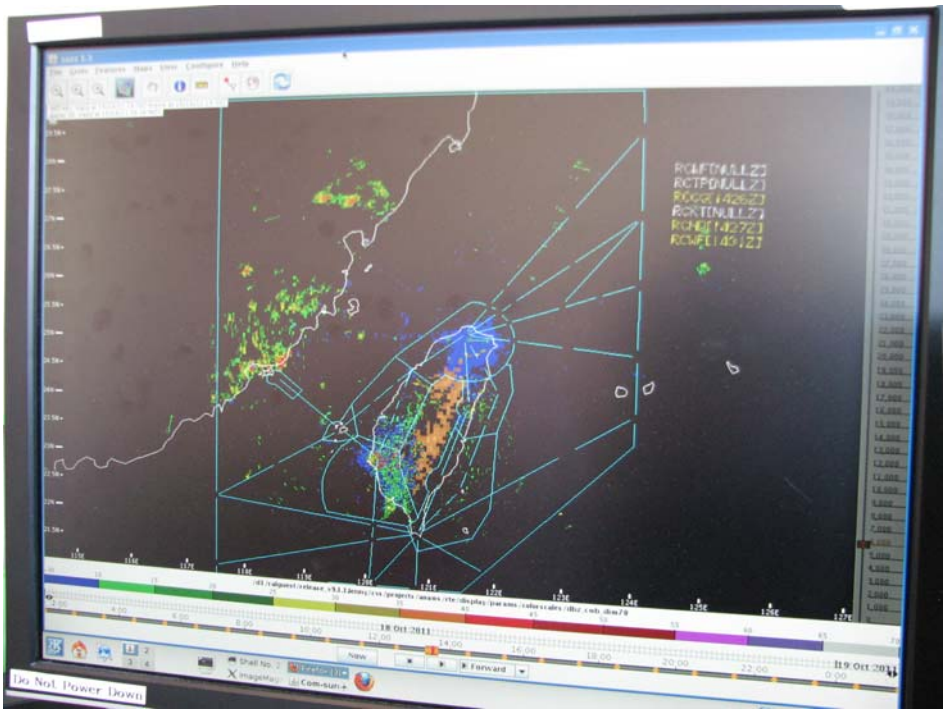


圖18 Jadeite打造之JMDS畫面(訓練時拍攝主機畫面所得)

肆、 心得

一、今年雖然航空氣象現代化作業系統第14號執行辦法預算，受到政府預算總額管控，相較往年減少。NCAR於今年初已向本總臺氣象中心提出，由於預算不足將導致無法提供完整之技術轉移訓練，希望本總臺氣象中心得以調整訓練方式。但在氣象中心眾人努力思考研擬下，以今年選定單一課程主軸並配合較往年更多的實際操作時間方式進行訓練。在人員選定上，則考量由於實務課程佔大多數，所以選派有多年系統管理經驗的張臺長與職參與訓練，一方面可減少NCAR授課講師的授課時間，另一方面希可藉此更精進氣象中心自行維護及排除系統異常狀態的能力。卻意外得到比往年更好的成效，日後可朝此方向與NCAR協調年度技術轉移訓練課程。

二、本次訓練發現NCAR在進行系統撰寫與規劃時，對於架構已有相當充足的預先規劃。本次訓練之主軸課程「AWOS資料顯示介面」無論在研習資料接收、資料轉化及資料顯示上，往往可在相當有系統的架構下進行，絕不會發生雜亂無章，使參訓人員無所適從的情況。顯示NCAR在規劃關於航空氣象現代化作業系統之程式時，已經預先研擬完成相關程式架構與內容，並且依循NCAR對於程式撰寫的規則進行。此將大大減少日後人員維護與系統調整的負擔。

三、NCAR使用CVS(Control Version System)進行系統管理。NCAR人員撰寫程式時，一定遵循以下程序：

1.先行於自己的電腦進行撰寫與調整，並且將資料Copy至電腦中進行測試。並若遇到沒有現成資料可供測試時，則設法以程式創造可供測試的虛擬資料。

2.在自己的電腦測試穩定後，將其移入測試環境中進行再測試，此時將進行長時間運作的穩定度測試。

3.最後移入正式作業環境(如航空氣象現代化作業系統AOAWS)中，並確定穩定運作後，納入CVS進行版本控管。

在此完整的管理架構下，系統維護者可清楚的知道目前程式系統的情況，確保系統保持在最新且最穩定的狀態。

四、由本次訓練情況得知，NCAR對於測試環境的重視。本次訓練因爲著重在實機操作，我們參訓人員一行四人，NCAR可以提供四部供訓練用的測試環境，實有我們可以效法的地方。因爲任何測試都可能影響到系統的正常運作，而NCAR所提供的訓練用主機，其作業環境與線上作業主機相近，所以參訓人員可以很清楚的了解課程內容，並且學成後亦將很容易的理解線上作業主機的相關程序位置。同時在訓練過程中，參訓人員的作業並不影響

實際作業環境，哪怕學員學習中不慎發生不可挽回的錯誤，可以利用CVS很快的將訓練用主機上相關程式，恢復到先前設定的狀態。

五、航空氣象現代化作業系統爲一完整且複雜的系統，其所牽涉的層面相當廣泛。在系統管理上，對於多爲氣象背景的本總臺氣象中心之管理維護人員，日後的維護與管理，勢必將有相當的技術門檻需要跨越。但就學習層面上而言，參與人員可因此了解本身專長以外的領域，實爲不可多得的機會。而在於行政管理上，可藉此使得參與人員了解各項管理流程與細節。就以上兩方面而言，航空氣象現代化作業系統計畫實爲民航局及本總臺孕育人才的搖籃。

伍、建議事項

今年爲航空氣象現代化作業系統氣象技術增強計畫四年期計畫之第一年，而航空氣象現代化作業系統執行至今已12個年頭，航空氣象現代化作業系統已經將臺北飛航情報區的航空氣象資訊提供方式，由過去人工與紙本方式轉爲電子化、網路的方式。由過去多部主機分散性的航空氣象資料，轉爲單一主機整合性的航空氣象資料顯示查詢介面。由過去查詢資料受到主機位置及作業系統限制，轉爲只需拿起手邊的電腦，透過網際網路連線就可以取得高品質的航空氣象資訊。由過去單調的查詢資料介面，進一步轉爲可與使用者高度互動的查詢介面，供使用者更容易取得航空氣象資訊。此系統歷經逐年發展改善，已讓使用者有更多的思考時間，進行資料的研判，以進一步保障飛安。

因此職等提出以下建議：

一、持續提升自行維護能力：由本次訓練的經驗得知，由於NCAR對於系統設計與規劃已相當成熟，且相當有系統性的規畫。而職參與這次訓練，雖然一開始曾有自身能力不足的疑慮，但在NCAR授課講師循序漸進的教導下，發現其實並非想像中難。所以在此建議，未來可考慮規劃以類似課程及學習方式爲主軸的訓練，如此可經由實機操作訓練中學習到更進一階層的系統維護與管理能力。

二、回國後的經驗分享：由於員額與預算關係，往年前往NCAR受訓的人員通常只有一或兩位。所以回國後的經驗分享，將更顯得重要。出國人員的訓練成果，可經由類似目前於本總臺氣象中心每月所執行的預報技術討論會，與未能出國的系統管理人員分享，以藉此精進整體系統管理的能力。

三、邁出系統自行維護升級的第一步：過去的訓練，主要著重在理解系統架構、資料運用及產品製作的原理，此多被運用於系統之障礙排除部分。而本次相當不一樣的是針對單一主軸進行深度實習。就職等而言，收穫相當多，並且對於未來AWOS資料顯示介面的管理與升級，深具信心。所以在此建議未來機場之AWOS資料介接、資料整合、資料轉化及資料顯示部分，可嘗試由氣象中心自行辦理，而NCAR可轉爲諮詢與協助的角色。未來隨著單一課程的深度學習，逐步將各種工作項目轉爲由本總臺辦理，NCAR轉爲輔助之角色，邁向系統自行維護的境界。

四、加強人員訓練：由於本總臺氣象中心人員，多以氣象背景爲主，對於資訊系統管理仍有需要加強的部分。而就未來航空氣象資訊系統發展的趨勢而言，友善查詢介面以及提供與使用者保持高度互動的系統設計將是未來趨勢，所以職在此建議未來氣象人員可針對以下課程進行學習：

1. Java課程：目前以Java設計的系統軟體已可符合上述之特性，同時目前臺北

飛航情報區提供航空氣象資訊的主力系統新一代航空氣象多元產品顯示系統(JMDS)以及自動天氣觀測系統(AWOS)資料顯示介面，皆爲Java所打造。所以Java相關課程的進修，顯得相當重要。

2. XML課程：如本報告書前面所述，JMDS及AWOS資料顯示介面除本身以Java打造之外，另外還需利用XML檔案進行參數設定，所以在XML檔案的管理與設計上，亦應擬爲進修課程項目。

3. 網頁嵌入式技術：JMDS及AWOS資料顯示介面，目前都是以Java Web Start技術供使用者經由航空氣象服務網連結點選開啓。另外目前航空氣象資料服務已有逐漸將Java技術嵌入網站中，提供高度之網頁互動性的資料查訊介面。未來可考慮將目前航空氣象服務網的資料展示方式，改爲更加活潑生動的Java Application架構，提升服務品質。

五、儘速取得Java安全憑證：本次訓練過程中，NCAR的授課講師曾經提到，目前編譯AWOS資料顯示介面及JMDS的Java安全憑證，爲NCAR申請的。由於NCAR辦理的航空氣象資訊系統，遍及世界各地，在爲了保障程式的安全。未來若民航局許可下，應自行辦理專屬於本身的Java安全憑證。同時未來若開始進行自行系統調整升級部分，所有Java程式編譯過程中，皆必須輸入安全憑證，故專屬於本身的Java安全憑證更是刻不容緩。

陸、附錄

本次研習課程表：

Training Objectives:

• Learn how to add new AWOS sites to AOAWS from ingest to display and update

system monitoring.

• Create an AWOS Ingest and Display maintenance manual.

• Learn how to use FIP in a forecasting situation.

• Gain familiarity with the user-configurable features available to a JADE-based display system that utilizes the Jadite framework.

Week 1:

Mon Oct 3:

Morning: UCAR visitor arrival orientation

Afternoon: Training introduction & logistics

Tue Oct 4:

Morning: AWOS data flow overview

Afternoon: AWOS Simulation set up

Wed Oct 5:

Morning: AWOS raw data ingest

Afternoon: AWOS raw data ingest & manual development

Thu Oct 6:

Morning: AWOS data check and configuration

Afternoon: AWOS data check and configuration & manual development

Fri Oct 7:

Morning: AWOS data conversion

Afternoon: AWOS manual development

Week 2:

Mon Oct 10:

Morning: AWOS data conversion

Afternoon: AWOS data flow monitoring & manual development

Tue Oct 11:
Morning: AWOS data flow monitoring
Afternoon: FIP forecaster training

Wed Oct 12:
Morning: AWOS manual development
Afternoon: AWOS manual development
(Management meeting 9AM – 5PM)

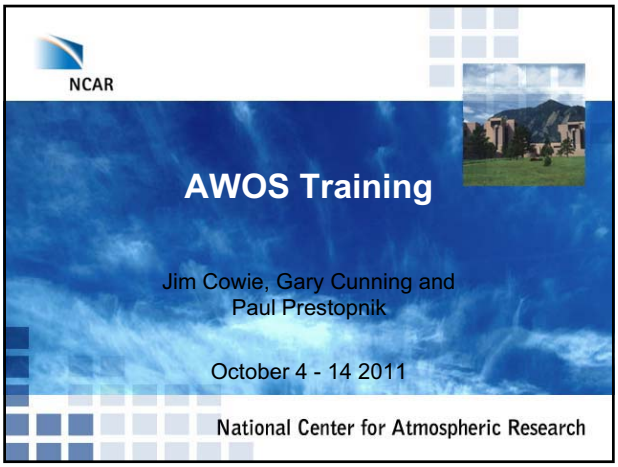
Thu Oct 13:
Morning: AWOS data display configuration
Afternoon: AWOS data display configuration & manual development

Fri Oct 14:
Morning: AWOS data display configuration
Afternoon: AWOS manual development

Week 3:
Mon Oct 17:
Morning: Jadite overview
Afternoon: Jadite configuration

Tue Oct 18:
Morning: Jadite configuration
Afternoon: Wrap-up

一、 AWOS系統架構與資料結構



Introduction

NCAR

- AWOS Review
- RCMT Development
- AWOS Display

System Concepts

NCAR

- AOAWS is primarily a data-driven design
  - processes stay running
  - new data triggers processing
- AOAWS relies on a schedule-driven design
  - data ingest
  - system monitoring (web page stats)
  - depends on cron
- Applications exist to bridge the two designs
  - LdataWatcher
  - InputWatcher
  - LdataWriter
- The data-driven design can be decomposed into two areas:
  - process management
  - data management

Host types

NCAR

- Host types serve to break processing and data into related or common areas
- Host types are used in the build and install process
- The AOAWS has the following host types:
  - model-server
  - caa-server
  - data-server
  - web-content
  - archive
  - wmds
  - mds

Roles

- Groups related activities that take place on a given host
- Expressed in the form of proc_list file and crontab
- Identifies services for particular host during the installation
  - has_printer
  - is_archive
- files are under $PROJ_DIR/<role>
  - control
  - params
  - scripts
- Installation process collects role proc_list files and crontabs into host-level files under $PROJ_DIR/control
- Configuration files that assign roles to host types are in $PROJ_DIR/system/control; files are name roles.<host_type>
- Roles can be active or inactive

Process management

- Each host has a process list file named proc_list located in $PROJ_DIR/control
- Processes have a name and an instance
- Start scripts are used to ensure that only one particular instance of process is running

Process management

- auto_restart
  - perl script
  - queries procmap for missing processes
  - restarts processes
- procmap and auto_restart activity

Data management

- Data is moved through AOAWS using RAL's Data Server (DS) infrastructure or framework
- It follows a socket-base client/server design
- DS infrastructure is supported by libraries and utilities.
- Supported languages
  - C/C++
  - Java
- Command line utilities allow a great deal of flexibility to work with many scripting languages

Process logging

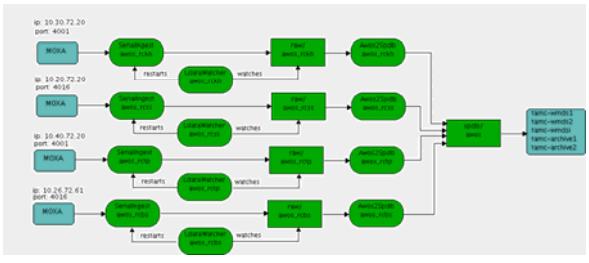
- In general all processes create log files
- Content of log file depends on process
- Use application LogFilter to filter messages into daily files under date subdirectories
- Logging directory layout
  - $LOG_DIR/<type>/<YYYYMMDD>/
  - type
    - distrib – file distribution
    - errors - processes
    - restarts – process restarts

System Design

- Host types & Roles
  - data → awos
- Applications
  - SerialIngest
  - Awos2spdb
  - DsFileDist

System Design

- Processing on tamc-data1 & tamc-data2



AWOS Ingest Applications

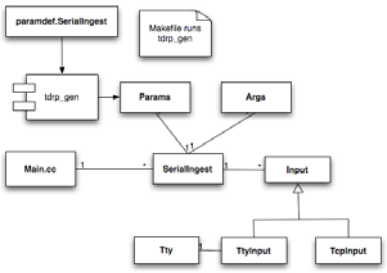
- Design
- Implementation
- Command line
- Configuration
- Data Structures
- Setup

SerialIngest

- Design
  - Reads a continuous stream of ASCII data from either a serial connection (RS232) or a port (TCP)
  - Writes stream to files at regular intervals
  - Highly configurable through parameter file
    - RS232 parameters
    - TCP parameters
    - End-of-line termination characters
    - Output file location
    - Write interval

SerialIngest

- Class diagram



SerialIngest

- Implementation
  - Source code located in apps/ingest/src/SerialIngest
  - Written in C++
  - Library dependencies
    - dsserver
    - didss
    - rapformats
    - toolsa
    - dataport
    - tdrp
    - math (C standard library)

SerialIngest

- Command line

```
SerialIngest [options as below]
options:
     [ --, -h, -help, -man ] produce this list.
     [ -debug ] print debug messages
     [ -verbose ] print verbose debug messages

TDRP args: [options as below]
  [ -params path ] specify params file path
  [ -check_params] check which params are not set
  [ -print_params [mode]] print parameters
    using following modes, default mode is 'norm'
      short:   main comments only, no help or descr
               structs and arrays on a single line
      norm:    short + descriptions and help
      long:    norm  + arrays and structs expanded
      verbose: long  + private params included
      short_expand:   short with env vars expanded
      norm_expand:    norm with env vars expanded
      long_expand:    long with env vars expanded
      verbose_expand: verbose with env vars expanded

  [ -tdrp_debug] debugging prints for tdrp
  [ -tdrp_usage] print this usage
```

SerialIngest

- Command line examples
  - SerialIngest –print_params > SerialIngest.test
  - SerialIngest –params SerialIngest.test -verbose

SerialIngest

- Configuration

debug	DEBUG_OFF, DEBUG_NORM, DEBUG_VERBOSE
instance	Used for registration with procmap
connection	SERIAL or TCP
input_device	name of device driver for serial port, /dev/ttyS1
baud_rate	Baud rate for incoming serial data port
dataIs7Bit	Set TRUE for 7-bit data, FALSE for 8-bit data.
twoStopBits	If TRUE, 2 stop bits. If FALSE, 1 stop bit.
enableParity	TRUE or FALSE
oddParity	TRUE or FALSE
tcp_server_host_name	Name of TCP server host
tcp_server_port	TCP server port number.
send_tcp_handshake	Option to send TCP handshake sequence to the server to trigger the data flow.
tcp_handshake_bytes	List of bytes to be sent to server for handshaking.
filter_ctrlm	TRUE or FALSE
end_of_message_check	Option to check for end of message before closing an output file.
output_interval	Interval at which output files are created (secs).
force_output_if_stalled	Flag for forcing the file output if the input stream in stalled.
discard_zero_length_file	Flag for discarding 0-length files.
output_dir_path	Name of output directory.
output_file_ext	Extension for output file.

SerialIngest

- Data Structures
  - There are not any data structures critical to the application's functionality.

SerialIngest

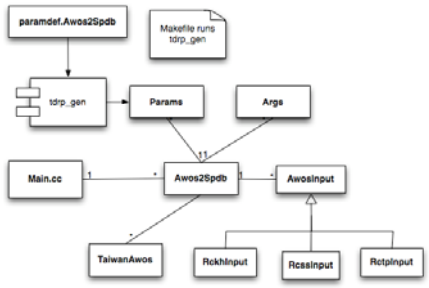
- Setup
  - Use of environment variables in start script can be useful. Look at start script in lab.
  - Use utilities like less and od (octal dump) to identify end-of-line termination characters.
  - Use telnet to test TCP port configuration settings.
  - Use kermit or a similar utility to test RS232 connection.

Awos2Spdb

- Design
  - Reads the raw ASCII files from either of the three AWOS system
  - A separate class exists to read and parse observations from raw files from each AWOS location.
  - Observations are represented by TaiwanAwos class.
  - TaiwanAwos is a wrapper for the SPDB message format
  - Observations are to an SPDB dagtabase

Awos2Spdb

- Class diagram



Awos2Spdb

- Implementation
  - Source code located in apps/aoaws/src/Awos2Spdb
  - Written in C++
  - Library dependencies
    - Spdb
    - dsserver
    - didss
    - rapformats
    - physics
    - toolsa
    - dataport
    - tdrp

Awos2Spdb

- Command line

```
Usage: Awos2Spdb [options as below]
options:
 [ --, -help, -h, -man ] produce this list.
    [ -debug ] debugging on
    [ -verbose ] verbose debugging on
    [-i instance_name] Instance string (no blanks)
    [ -f files ] specify input file list.
      forces FILELIST mode
    [ -start "yyyy mm dd hh mm ss"] start time
                ARCHIVE mode only
    [ -end "yyyy mm dd hh mm ss"] end time
                ARCHIVE mode only
    [ -out_url url] Output URL

TDRP args: [options as below]
  [ -params path ] specify params file path
  [ -check_params] check which params are not set
  [ -print_params [mode]] print parameters
    using following modes, default mode is 'norm'
      short:   main comments only, no help or descr
               structs and arrays on a single line
      norm:    short + descriptions and help
      long:    norm  + arrays and structs expanded
      verbose: long  + private params included
      short_expand:   short with env vars expanded
      norm_expand:    norm with env vars expanded
      long_expand:    long with env vars expanded
      verbose_expand: verbose with env vars expanded
  [ -tdrp_debug] debugging prints for tdrp
  [ -tdrp_usage] print this usage
```

Awos2Spdb

- Command line examples
  - Awos2Spdb –print_params > Awos2Spdb.test
  - Awos2Spdb –params Awos2Spdb.test -verbose

Awos2Spdb

- Configuration

debug	DEBUG_OFF, DEBUG_NORM, DEBUG_VERBOSE
instance	Used for registration with procmap
mode	ARCHIVE, REALTIME, FILELIST
start_time	start time for ARCHIVE MODE
stop_time	end time for ARCHIVE MODE
input_dir	Input directory
latest_data_info_avail	Set to true if there is a latest_data_info file available in the input directory
max_realtime_valid_age	Max valid age of input files in realtime mode (secs).
strict_sub_dir_check	When set, only checks input_dir for subdirs of the form YYYYMMDD.
file_name_check	When set, check file name contains specified sub-string.
file_match_string	String to check check file name against
write_all_obs	When set to true, all observations in file will be written to output.
sleep_interval	Sleep interval in seconds.
output_url	The ASOS/AWOS data is written to this URL in SPDB format.
data_set_info	This text is placed in the product info part of the output
expire_seconds	Expire interval in seconds for each AWOS site message.
awos_input	sets the AWOS input type
airport_name	the airport name. this name must be four characters in length.
awos_list	name and location of the AWOS.
timestamp_source	ource of timestamp for SPDB message valid time. Options are TIMESTAMP_SOURCE_OBSERVATION TIMESTAMP_SOURCE_FILENAME.

Awos2Spdb

- Data Structures

```
typedef struct {
   char airport_id[AIRPORT_ID_STR_LEN];
   char awos_id[AWOS_ID_STR_LEN];
   char info[INFO_STR_LEN];
   fl32 latitude;
   fl32 longitude;
   fl32 altitude;
   fl32 spare_1;
   fl32 spare_2;
   fl32 spare_3;
   fl32 spare_4;
   si32 buf_len;
   si32 airport_id_len;
   si32 awos_id_len;
   si32 info_len;
   si32 spare_5;
   si32 spare_6;
   si32 spare_7;
   si32 spare_8;
   si32 spare_9;
 } awos_header_t;
```


Awos2Spdb

- Data Structures

```
the units are:
  wind speeds      -- knots
  wind directions  -- degrees (from true North?)
  pressure         -- hPa
  RVR              -- meters
  visibility       -- meters
  rainfall acc.    -- millimeters
  temperature      -- Celsius
  dew point        -- Celsius
  humidity         -- %
  cloudiness       -- enum: NONE (0); FEW (1, 2); SCT (3, 4); BKN (5-7); OVC (8)
  cloud height     -- feet

  In awos_obs_t, min_rvr, min_rvr_10_min_avg, min_low_cloud_hgt,
  min_med_cloud_hgt and min_high_cloud_hgt are minimum height flags.
  flag values are 0 or 1 to indicate that rvr, for instance,  is a minimum height.
```

Awos2Spdb

- Data Structures

```
typedef struct {
   ti32 valid_time;
   fl32 instant_wind_speed;
   fl32 instant_wind_dir;
   fl32 avg_wind_speed_2_min;
   fl32 avg_wind_dir_2_min;
   fl32 max_wind_speed_2_min;
   fl32 min_wind_speed_2_min;
   fl32 max_wind_dir_2_min;
   fl32 min_wind_dir_2_min;
   fl32 avg_wind_speed_10_min;
   fl32 avg_wind_dir_10_min;
   fl32 max_wind_speed_10_min;
   fl32 min_wind_speed_10_min;
   fl32 max_wind_dir_10_min;
   fl32 min_wind_dir_10_min;
   fl32 qnh;
   fl32 qff;
   fl32 qfe;
   fl32 rvr;
   si32 min_rvr;
   fl32 rvr_10_min_avg;
   si32 min_rvr_10_min_avg;
   fl32 vis;
   si32 min_vis;
   fl32 vis_10_min_avg;
   si32 min_vis_10_min_avg;
   fl32 temperature;
   fl32 dewpoint;
   fl32 humidity;
   fl32 rainfall_acc_1_hr;
   fl32 rainfall_acc_6_hr;
   fl32 rainfall_acc_12_hr;
   fl32 rainfall_acc_24_hr;
   si32 low_cloudiness;
   si32 med_cloudiness;
   si32 high_cloudiness;
   fl32 low_cloud_hgt;
   fl32 med_cloud_hgt;
   fl32 high_cloud_hgt;
   si32 min_low_cloud_hgt;
   si32 min_med_cloud_hgt;
   si32 min_high_cloud_hgt;
   fl32 spare_1;
   fl32 spare_2;
   fl32 spare_3;
   fl32 spare_4;
   si32 spare_5;
   si32 spare_6;
   si32 spare_7;
   si32 spare_8;
} awos_obs_t;
```


Awos2Spdb

- Setup
  - Use SpdbQuery to examine output.
  - Compare raw file to SPDB output.

RCMT Development

- Development process
- Work with raw files
- Create RCMT simulation
- Refactor Awos2Spdb
- Refactor AOAWS architecture
- Deploy changes on lab system
- Deploy on changes on operational system

Development process

- Develop and test applications and components on desktop
- Develop and test architecture and infrastructure on lab system
- Deploy on operational system
- Monitor operational system

Work with raw files

- Get sample of raw data
  - Use telnet to examine stream form port

  telnet neptune 4003 | od -c

  telnet 10.26.72.61 4001 | od –c
  - Compare with ICD

  RCSS ICD
  - Save sample for testing

  telnet neptune 4003 > rcbs.test

Create RCMT simulation

- Collect one day of RCMT files or transform files from existing AWOS site
  - Collect files by temporarily setting up SerialIngest on operational system
  - Transform RCSS files using sed
- Explore Transformation method:

Use raw RCSS AWOS as a basis to create RCMT files
perform the following replacements

- *RCSS --> RCMT*
- *AWS_10 --> AWS_03*
- *AWC_28 --> AWS_21*
- filename changes from HHMMSS.awos_rcss --> HHMMSS.awos_rcbs

The following command will perform all the replacements in a file:

```
sed 's/RCSS/RCMT/g' 235930.awos_rcss | sed 's/AWS_10/AWS_03/g' | sed
    's/AWS_28/AWS_21/g' > 235930.awos_rcmt
```


Create RCMT simulation

A shell script to transform all the files looks like:

```
#!/bin/sh
#
files=`ls *awos_rcss`
for name in $files do
   new_name=`echo $name | sed 's/rcss/rcbs/g'`
   sed 's/RCSS/RCMT/g' ./$name | sed 's/AWS_10/AWS_03/g' | sed
   's/AWS_28/AWS_21/g' > $new_name /bin/rm -f ./$name
done
exit 0
```

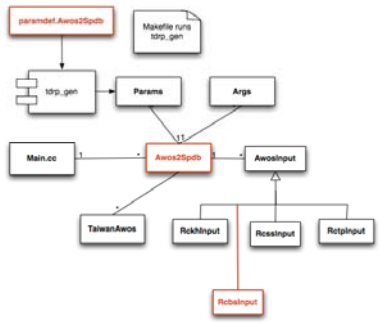

Create RCMT simulation

- Set up simulating raw RCMT data using file_repeat_day on neptune
  - create start script
  - create parameter file
  - update process list,
  - install _DsFileDist under data/raw/awos_rcbs.
  - restart system

Refactor Awos2Spdb

- Review design and code
- Add INPUT_RCMT to awos_input_t
- Create RcmtInput class
  - Copy RcmtInput as a starting point
  - Implement readNext() method
    - Use ICD as guide
- Update Awos2Spdb class
- Compile code
- Test code
  - Use print statements and debugger
- Verify
  - Use SpdbQuery

Refactor Awos2Spdb



Refactor Awos2Spdb

```
-rw-rw-r-- 1 cunning rap4  5897 2009-04-08 17:47 Args.cc
-rw-rw-r-- 1 cunning rap4  1759 2008-10-25 22:11 Args.hh
-rw-rw-r-- 1 cunning rap4 14275 2010-09-08 19:59 Awos2Spdb.cc
-rw-rw-r-- 1 cunning rap4  3100 2009-05-28 23:22 Awos2Spdb.hh
-rw-rw-r-- 1 cunning rap4  8889 2009-10-03 21:51 AwosInput.cc
-rw-rw-r-- 1 cunning rap4  5078 2009-10-03 21:51 AwosInput.hh
drwxrwxr-x 2 cunning rap4  4096 2010-09-13 22:09 CVS/
-rw-rw-r-- 1 cunning rap4  1969 2009-04-08 17:47 Main.cc
-rw-rw-r-- 1 cunning rap4  1065 2010-09-08 19:59 Makefile
-rw-rw-r-- 1 cunning rap4  6894 2010-09-08 19:59 paramdef.Awos2Spdb
-rw-rw-r-- 1 cunning rap4 28932 2010-09-08 19:58 RcmtInput.cc
-rw-rw-r-- 1 cunning rap4  4413 2010-09-08 19:58 RcmtInput.hh
-rw-rw-r-- 1 cunning rap4 28932 2010-09-08 19:58 RcbsInput.cc
-rw-rw-r-- 1 cunning rap4  4413 2010-09-08 19:58 RcbsInput.hh
-rw-rw-r-- 1 cunning rap4 19320 2009-10-03 21:51 RckhInput.cc
-rw-rw-r-- 1 cunning rap4  3330 2009-08-18 19:53 RckhInput.hh
-rw-rw-r-- 1 cunning rap4 28934 2009-10-03 21:51 RcssInput.cc
-rw-rw-r-- 1 cunning rap4  4413 2009-09-03 16:22 RcssInput.hh
-rw-rw-r-- 1 cunning rap4 29158 2009-10-03 21:51 RctpInput.cc
-rw-rw-r-- 1 cunning rap4  4370 2009-09-27 01:16 RctpInput.hh
```


Refactor AOAWS architecture

- Add rcmt instance of SerialIngest to awos role to data-server host type
  - Create parameter file and start script
  - Create data directory structure
  - Modify process list file, proc_list.awos
- Add rcmt instance of Awos2Spdb
  - Create start script
  - Create parameter file
  - Modify process list file, proc_list.awos
- Add rcmt instance of LdataWatcher
- Modify SysView diagram

Deploy changes on lab system

- Install RCMT simulation on neptune
  - Checkout files from cvs
  - Copy files to ~/projDir
  - Restart system
- Install new version of Awos2Spdb on tamc-data1
  - Checkout source from cvs
  - Make application
  - Install under ~/projDir/bin
- Install AOAWS Architecture changes on tamc-data1
  - Check out files from cvs
    - SerialIngest
    - Awos2Spdb
    - SysView diagram
    - Data directories
  - Install files
    - Copy under ~/projDir
  - Restart system

Deploy changes on lab system

- Test System
  - All processes are running
  - Check that raw files are copied to tamc-data1
  - Check that RCMT messages are in SPDB database
    - Use SpdbQuery
  - Commit any bug fixes to cvs
  - Build new distribution file for AOAWS
  - Test build by installing on lab system

Deploy on changes on operational system

- Two ways to deploy on operational system
  - Use cvs to update files
  - Install distribution
- Install changes
  - All changes will be on tamc-data1 and tamc-data2
- Restart system
  - Restart all on tamc-data1
  - Restart SysView on tamc-data2
- Monitor system

AWOS Display Overview

- Technologies
  - Java Web Start
  - XML
- AWOS Display Configuration

Technologies

The AWOS Display is built on top of the Java language
- Version 8 requires Java 6 or newer
- Makes use of Java Web Start
- Makes use of XML config files

Java Web Start

- Allows Java applications to be distributed and run across the Internet
- Provides security features
  - Signing and certificates ensure application creators are who they say they are
  - Sandbox - user has to grant applications permission to use the network, access files, etc.
- Client/server oriented
  - Application configuration and files reside on server, are downloaded via HTTP by the clients
  - Internet access and Java are all that is required

Java Web Start file

```
<?xml version="1.0" encoding="utf-8"?>
<jnlp
  spec="1.0+"
  codebase="http://acaps.caa.gov.tw/mmda/content/acaps/awos/">

  <information>
    <title>Experimental AWOS Display</title>
    <vendor>National Center for Atmospheric Research (NCAR) Research Applications Laboratory (RAL)</vendor>
    <description>AWOS Display Version 0, an experimental component of the ACAWS system</description>

    <homepage href="http://acaps.caa.gov.tw/mmda/content/acaps/awos/"/>

  </information>
  <security>
     <all-permissions/>
  </security>
  <!-- Always require an update.  If this is not done, then the authentication screen times out before new versions can be downloaded -->
  <update check="always" policy="always"/>
  <resources>
    <j2se version="1.6+" max-heap-size="128M"/>

    <property name="sun.net.client.defaultConnectTimeout" value="20000"/>
    <property name="sun.net.client.defaultReadTimeout" value="20000"/>

    <jar href="awos.jar"/>
    <jar href="commons-digester-1.8.jar"/>
    <jar href="commons-beanutils-1.8.0.jar"/>
    <jar href="commons-logging-1.0.4.jar"/>
    <jar href="conf.jar"/>
  </resources>

  <application-desc main-class="edu.ucar.rap.acaps.apps.awos.AwosDisplay">
    <argument>-config_file</argument>
    <argument>edu/ucar/rap/acaps/apps/awos/config_taiwan.xml</argument>
  </application-desc>
</jnlp>
```


XML

- XML used for AWOS Display configuration files
- Similar to HTML, but more general-purpose and more strict about closing tags
- A text standard to encode structured/hierarchical information
- Elements and attributes
- Developed by the W3C, the organization who created the HTTP standard

Fundamental Concepts

Airport

The location of one or more weather observing sensors. RCTP, RCSS, etc.

AWOS

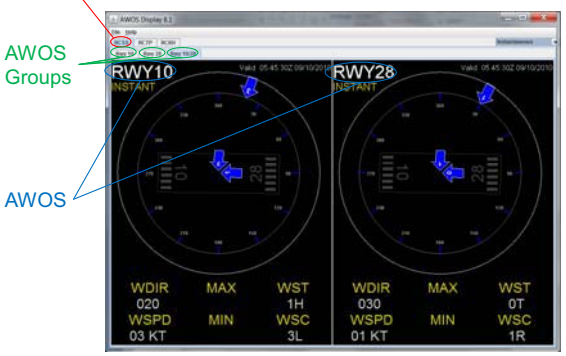
A sensor that is observing weather conditions

AWOS Group

A group of one or more AWOSs that are typically viewed together. For example, the RCTP 05/06 pair is often viewed together

AWOS Display Concepts

AWOS Display Concepts

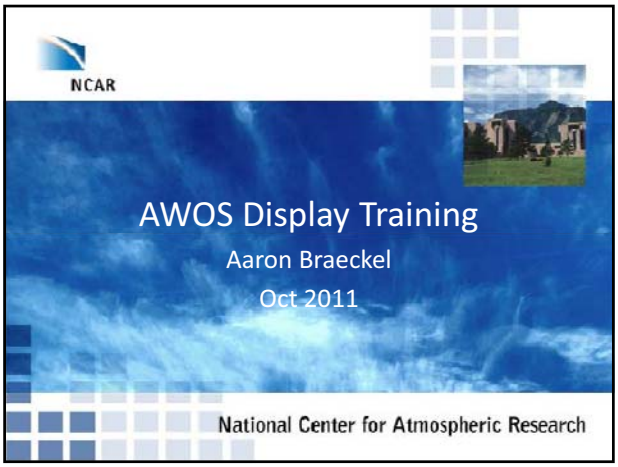


AWOS Configuration

AWOS Configuration (contd)

directionDegTrue – Degrees true north of the associated runway
label – name shown in the display
identifierDeg - "public" identifier direction. This may be any value (in 10 degree increments) from 0-350. This direction is different from directionDegTrue when the identified direction differs from the actual runway direction (for example: RCTP 05/06)

二、 AWOS資料顯示介面設計



Overview

NCAR

- Technologies
  - Java
  - Swing
  - Java Web Start
  - XML
  - SPDB
- AWOS Display
- Basic Concepts
- Data Framework (DataManager)
- Configuration

Technologies

NCAR

The AWOS Display is built on top of the Java language
  – Java 6 or newer
- Makes use of Java Web Start
- Complicated rendering, but overall much simpler than the JMDS
- Makes use of XML config files

Differences from the JMDS

NCAR

- Much simpler
- Much smaller
  – JMDS is over 11Mb
  – AWOS Display is under 1Mb
- Not based on JADE, but uses a simpler data framework
- Focused on real-time data display vs overlaying multiple products

Other basics

- Uses Ant as the build tool
- Located in the RAL CVS repository

Experience

- Low, Medium, High
  - Java
  - Swing
  - Java Web Start
  - XML
  - SPDB
  - Ant
  - CVS

JavaDoc

Standard Java utility

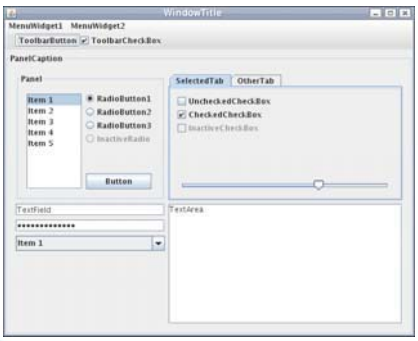
Markup Java source code, then run 'javadoc' to generate HTML documentation

View JavaDoc

Java Swing

- Swing is a cross-platform interface toolkit
  - Buttons, windows, text boxes, and other widgets
- AWOS Display user interface written using Swing
- Looks similar across platforms, with variations
- Pluggable look and feel (runtime or compile-time)
  - Custom or roughly emulating platform-specific frameworks
- Swing comes bundled with modern Java installations

Java Swing



XML

- XML used for AWOS Display configuration files
- Similar to HTML, but more general-purpose and more strict about closing tags
- A text standard to encode structured/hierarchical information
- Elements and attributes
- Developed by the W3C, the organization who created the HTTP standard

XML continued

- Well-developed standard with many related standards
  - XSLT, XPath, XQuery, etc.
- XML Schemas can be used to formally define the expected XML structure

Java Web Start

- Allows Java applications to be distributed and run across the Internet
- Provides security features
  - Signing and certificates ensure application creators are who they say they are
  - Sandbox - user has to grant applications permission to use the network, access files, etc.
- Client/server oriented
  - Application configuration and files reside on server, are downloaded via HTTP by the clients
  - Internet access and Java are all that is required

Java Web Start process

- AWOS Display application is on the WMDS
- User clicks on a link, which:
  - Downloads the application and stores it on the user machine
  - Asks the user whether they want the AWOS Display to be able to access the network and have access to the file system
- Whenever the AWOS Display is started, it checks the WMDS for new files
  - If there are any, the new files are downloaded and used

Java Web Start conclusions

Therefore:

- Simple and automatic installation and updates
- Does not require that the TAMC or NCAR perform a software installation for each user (besides on the WMDS)
- Centrally managed

Java Web Start commands

javaws –viewer

View/modify the installed JWS applications

javaws http://aoaws.caa.gov.tw/wmds/content/aoaws/awos/AWOSDisplayTaiwan.jnlp

Run or install the AWOS Display

The browser associates .jnlp files with the javaws command, so running through the browser is equivalent to this command

*The javaws command is typically in the path, but is located in $JAVA_HOME/bin/

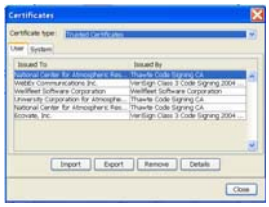
Certificates and JAR signing

Restricting application access

JNLP file has a security section to request permissions

If ANY permissions are requested, all JARs must be signed with a certificate and all must be signed with the same certificate

Certificates



*Certificates are part of a certificate chain, where “root” certificates/organizations are deemed trustworthy and credible to authorize others

Java Web Start permissions

Without additional privileges:

- No access to local disk
- All your jars must be downloaded from the same host. Note, however, that you can download extensions and JREs from any host as long as they are signed and trusted
- Network connections are allowed only to host from which your jars were downloaded
- No security manager can be installed
- No native libraries (not even in extensions)
- Limited access to system properties

Java Web Start file

Fundamental Concepts

Airport

The location of one or more weather observing sensors. RCTP, RCSS, etc.

AWOS

A sensor that is observing weather conditions

AWOS Group

A group of one or more AWOSs that are typically viewed together. For example, the RCTP 05/06 pair is often viewed together

Data Framework (DataManager)

The DataManager framework was originally developed in the JAWS/Juneau display system

It handles data timing and update rates, and fires simple events to interested listeners

Data Framework Concepts

Data Key

A unique identifier for a data product. Has an associated Finder and Retriever

Finder

Called on a regular period to look for newly-available data. Configured with an update rate and a data late period

Retriever

When new available data is found by the Finder, the Retriever is tasked with retrieving it. Needs data URLs to access the data

Data Framework Concepts

Event Listener

An object that is interested in being notified of events relating to a particular data key.

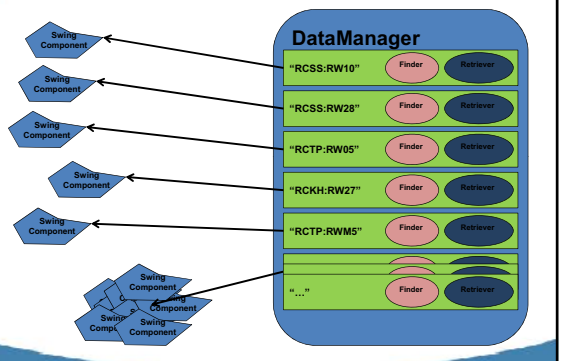
dataAvailable – new data has been found. Event object has an object representing the data that was discovered

dataRetrieved – new data was found, and was successfully retrieved. Event object has an object representing the data that was discovered, and the actual data object that was retrieved

dataLate – no newly-available data was found within the configured data late window

dataMissing – there was a problem finding or retrieving data. An unexpected Exception was thrown by a Finder or Retriever. This could result from server problems, network issues, or Finder/Retriever bugs

Fundamental Design



Therefore…

There is a Swing component that is an Event Listener

This component is interested in being notified of events relating to a single airport/AWOS combination

dataAvailable – new AWOS data is available

dataRetrieved – new AWOS data was found and was retrieved. It's time to render it

dataLate – data was not found in time. Visually indicate that the data we are showing is late

dataMissing – there was a problem finding or retrieving data. Visually indicate missing data but continue showing the latest data we have

Just a bit more complicated…

For the sake of simplicity, rendering the data is broken into two parts:

- Rendering the AWOS text values
- Rendering the AWOS gauge (visual)
  - This is fairly complicated code!

One more piece

The latest METARs are shown for the airports

Similarly, each METAR for a particular airport has a data key and associated DataManager classes

A METAR-specific component knows how to render METAR text

Concurrency

Generally to maximize performance it is desirable to have I/O on separate threads

The current DataManager implementation maintains a separate thread for each data key. Therefore, Finder and Retriever logic is done on a unique thread for each data key

Data late checks are done on a separate thread

A full description of the concurrency model is beyond the scope of this discussion

(Java 5 concurrency libraries)

Subtle Issues

Time synchronization among AWOS sites, and meaningful/accurate display of data times (ingest time vs AWOS observation time)

Client-side time issues

Retrieved data timestamp checks and SPDB LATEST mode

Back-end data Sources

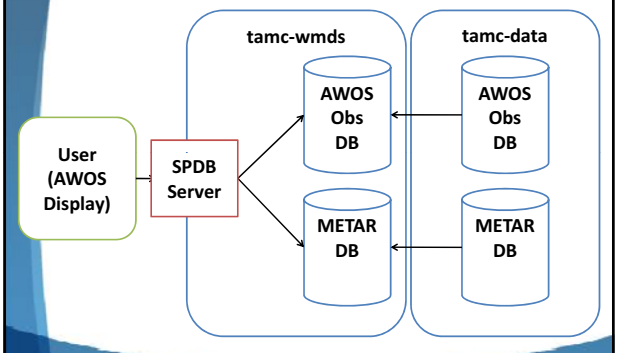
A single SPDB server

Two data products

- AWOS observations
  - 30 second client-side update rate*
- METAR observations
  - 30 second client-side update rate*

*at present

Communications



SPDB Messages

SPDB Messages:

- **dataType (integer)**
- **dataType2 (integer)**
- **time**

METAR SPDB Messages

METAR Messages:

- **dataType (integer) –** hash of station identifier ("RCTP")
- **dataType2 (integer) –** not used
- **Time –** observation time

AWOS SPDB Messages

AWOS Messages:

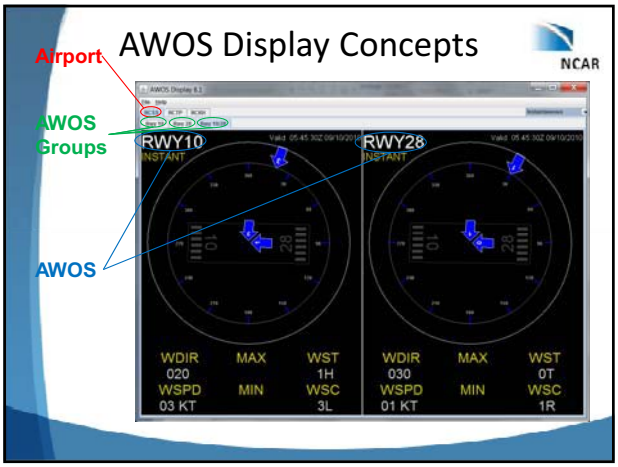
- **dataType (integer) –** hash of airport identifier ("RCTP")
- **dataType2 (integer) –** hash of AWOS identifier ("Rw10", "Rw28","RwM5")
- **Time –** system ingest time

SPDB Message Details

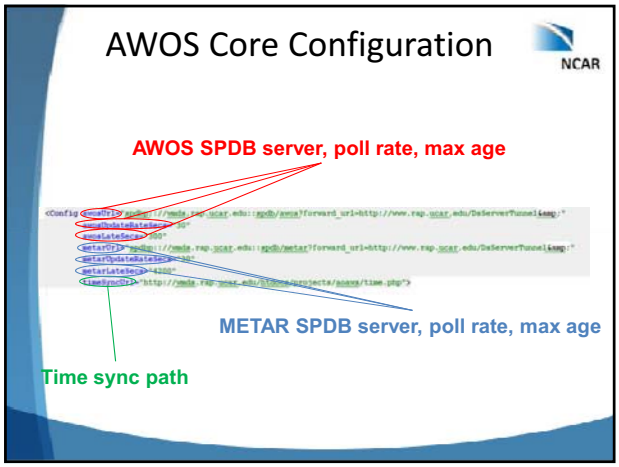
Every message type differs...

AWOS Display Concepts

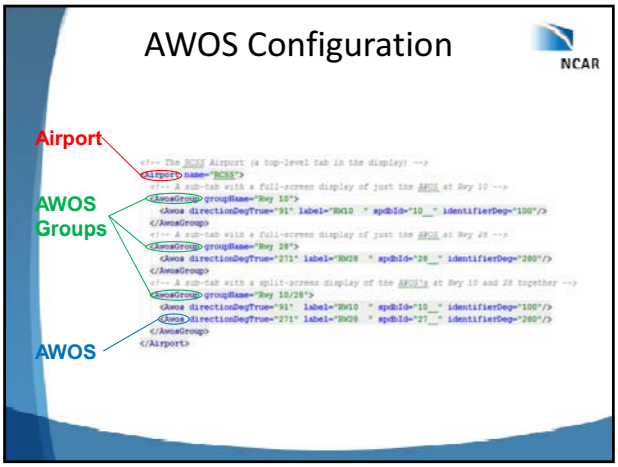
AWOS Display Concepts



AWOS Core Configuration



AWOS Configuration



AWOS Configuration (contd)

directionDegTrue – Degrees true north of the associated runway
label – text shown in the display
spdbId – SPDB dataType2 (AWOS identifier)
identifierDeg - "public" identifier direction. This may be any value (in 10 degree increments) from 0-350. This direction is different from directionDegTrue when the identified direction differs from the actual runway direction (for example: RCTP 05/06)

Deployment

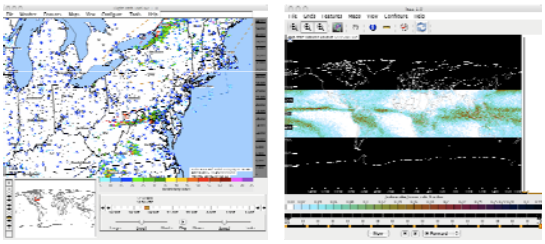
Let's make our changes and walk through deploying the latest updates to the AWOS Display...

三、 Jadeite介紹

Jade, Jadeite, Jazz

A Family of Visualization Tools

Arnaud Dumont – 10/17/2011



Topics

- Background
- Architecture
- Sample Applications
- XML Configuration
- Resources

Background

Java Visualization Frameworks

1998 - Java Applet

2002 - DataCanvas

2005 – Jade (including JMDS)

2010 - Jadeite

Background

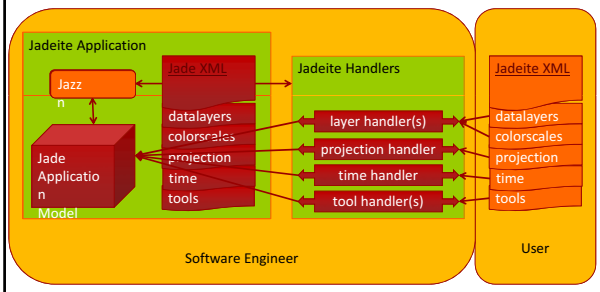
jade /jād/
Noun: 1. A hard, typically green stone used for ornaments and implements 2. A Java application development environment

- Used for many RAL displays
  - Flight Path Tool, HEMS Tool, Jviz, JMDS, EDR Viewer

jade·ite / jādīt/
Noun: 1. A green, blue, or white mineral that is one of the forms of jade $NaAlSi_2O_6$ 2. Jade for scientists

- Provides a CIDD-like configuration for Jade
  - CIDD is in its end-of-life
  - Jade is too complex for scientists to configure

Architecture



Architecture

- How to launch a Jadeite application?
  - ① Run the default Jazz application from the web
  - ② Check out and build the application from CVS
  - ③ Set up a custom Jazz application for a project
- How to specify a Jadeite XML Configuration File
  - ① On the command line
    when launching the Jadeite application locally
  - ② In the Java Network Launching Protocol (JNLP) file
    when launching the Jadeite application on the web
  - ③ In the application with a file chooser
    if neither of the other cases were satisfied

Sample Applications

- Jazz project:
  http://rap.ucar.edu/projects/jazz/trmm_climo_monthly1.jnlp
- Experimental ADDS:
  http://test.weather.aero/fpt.jnlp
- High Ice Water Content project:
  http://rossi/htdocs/display/hiwc_darwin_jazz.jnlp

XML Configuration

- Layer Definition
  - Type
  - Data Location
  - Colorscale and Rendering
- Geographic Projection
- Default Time Range and Mode
- Default Altitude Range and Unit
- Optional Tools

XML Configuration

- Demonstration
  - Sample Jadeite File
  - JMDS CIDD File

Resources

- Jazz project page:
  http://www/projects/jazz
- Jadeite and Jazz wiki pages:
  http://sdg/confluence/display/crosspgm/Jadeite
  http://sdg/confluence/display/crosspgm/Jazz
- Jadeite and Jazz Jira projects:
  http://sdg/jira/browse/JADTE
  http://sdg/jira/browse/JAZZ
- Nancy, Arnaud, Aaron, Rob

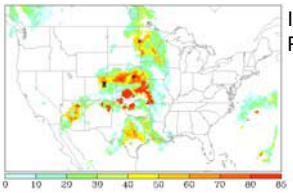
四、　積冰預報產品(FIP)說明

FIP Forecaster Training

Cory Wolff
cwolff@ucar.edu
11 October 2011

NATIONAL CENTER FOR ATMOSPHERIC RESEARCH

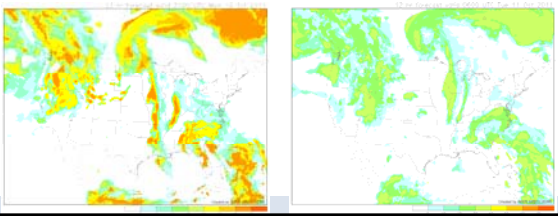
FIP



Icing Probability

Icing Probability

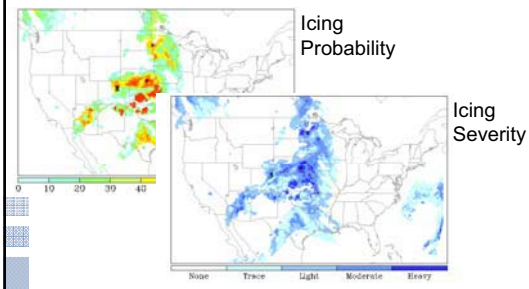
- CONUS Only
- Calibrated likelihood of icing
  - Based on comparisons with icing reports in high traffic areas
  - Decreases with increasing lead time



Icing Potential

- Likelihood of icing
  - Not a percentage (i.e. 50 does not mean a 50% chance, just that it's more likely than 30 and less likely than 70)
  - Probability calibration requires high density observations of icing
- Seasonal interest maps
  - Tropical (May to October) vs. Mid-latitude (November to April) flow patterns
  - Temperature map changes
    - Lower temperatures allowed in mid-latitude regime

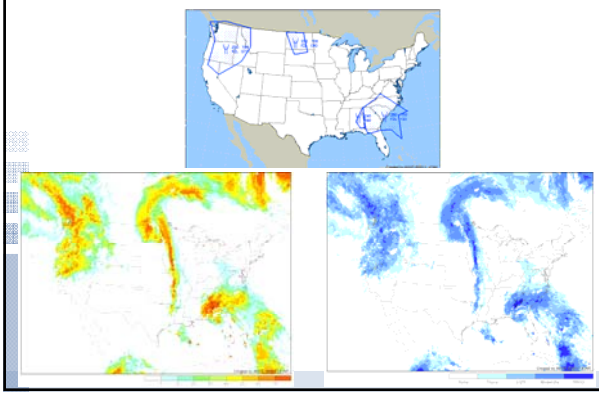
FIP



Icing Severity

- Attempting to forecast amount of liquid water present
  - Production (numerator) vs. depletion (denominator)
- Scenario based
  - Terms differ for production and depletion
  - Freezing rain vs. snow vs. no precipitation
- Output is a 0 – 1 value
  - Thresholds applied ➔ Trace, Light, Moderate, or Heavy icing

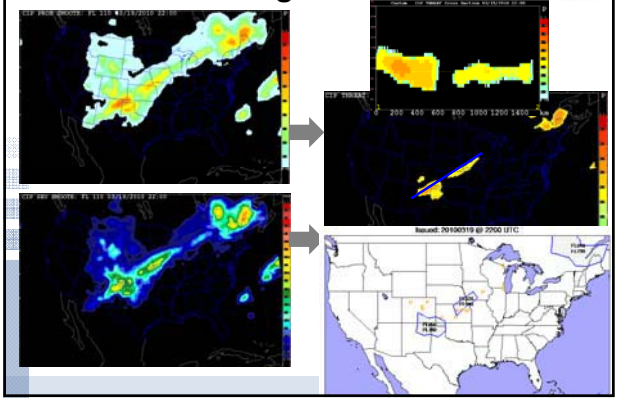
Operational Use



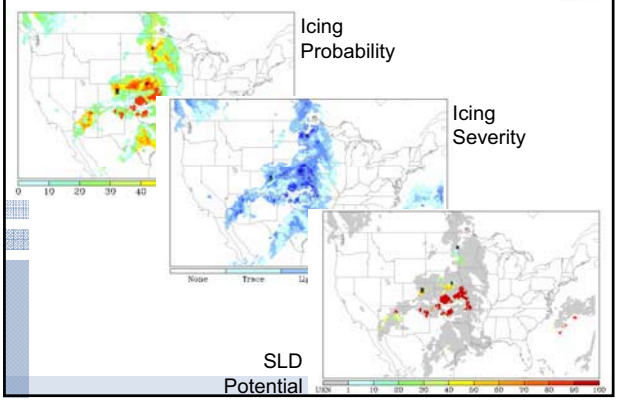
IceThreat

- Outlines areas of potential icing threats
  - Three dimensions
- Highly configurable
  - Parameters set by user
- Smoothes fields (probability and severity)
- Calculates icing threat by combining fields
- Clumps volumes of high threat together
- Creates shapes with a floor and ceiling

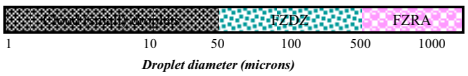
Icing Threat



FIP

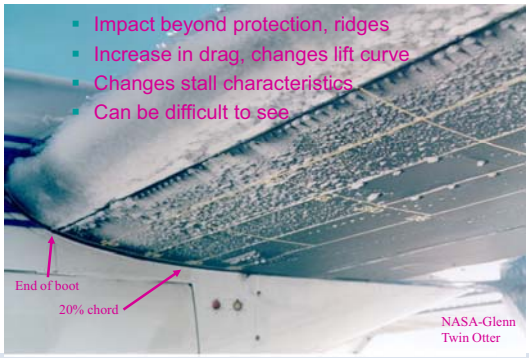


SLD

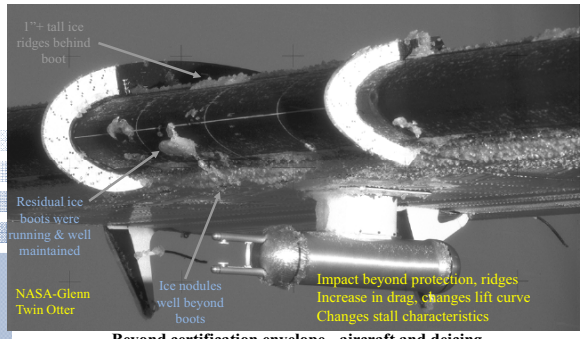


- 1 micron = .001 mm
- Dangers
  - Icing beyond protection
    - Drops go through air stream or run back
  - Odd shapes
    - Especially when combined with small drops

SLD Icing - Freezing Drizzle



SLD Icing - Freezing Rain



Beyond certification envelope - aircraft and deicing equipment are not certified for flight in FZDZ or FZRA

SLD in FIP

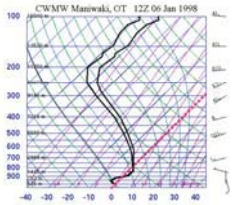
Collision Coalesence

- Drops form and grow to DZ size by colliding
- No ice process - warm cloud top (> -14 ° C)
- Boundary layer
  - Attached to the surface
  - High drop concentrations
  - Requires more liquid to form SLD
- Non-boundary layer
  - Clean and isolated from the boundary layer
  - Low drop concentrations
  - Requires less liquid to form SLD

SLD in FIP

Freezing Rain

- Requirements
  - Deep continuous moist layer
  - CTT < -12 ° C
  - Surface precipitation
  - Elevated melting layer above a subfreezing layer (warm nose)



- Function of temperature and amount of condensate

SLD in FIP

Convection

- Strong updrafts and warm cloud bases cause high supersaturations and a better chance for large drops
- Fields
  - Convective QPF
  - Temperature (down to -30 ° C)
  - CAPE
  - CIN
  - LI
  - TT
  - KI