AIAA 2001-4057

A UNIQUE SOFTWARE SYSTEM FOR SIMULATION-TO-FLIGHT RESEARCH

Victoria I. Chung*, Brian K. Hutchinson*
NASA Langley Research Center
MS 125B
Hampton, VA 23681

Abstract

"Simulation-to-Flight" is a research development concept to reduce costs and increase testing efficiency of future major aeronautical research efforts at NASA. The simulation-to-flight concept is achieved by using common software and hardware, procedures, and processes for both piloted-simulation and flight testing. This concept was applied to the design and development of two full-size transport simulators, a research system installed on a NASA B-757 airplane, and two supporting laboratories. This paper describes the software system that supports the simulation-to-flight facilities. Examples of various simulation-to-flight experimental applications were also provided.

Introduction

In 1974, NASA Langley Research Center (LaRC) obtained a research airplane, a Boeing 737-100 series aircraft, to conduct aviation systems and operational research for the Advanced Transport Operating Systems (ATOPS) program. The standard airplane systems were modified and interfaced to a separate research system. The airplane was named the Transport Systems Research Vehicle (TSRV) and could be flown from take-off through landing from the conventional forward flight deck or from a full-size, two-crewmember, research aft flight deck (AFD). The AFD was located in the passenger cabin. Thrust and flight control inputs were made by the AFD crew through control inceptors or through a research autopilot system via a fly-by-wire computer system. During flight research operations, the AFD was generally flown by test subject crews while the forward flight deck was manned by safety pilots to monitor, engage, and disengage the aft flight deck as an in-flight safety measure.

Besides the aft flight deck, the research system installed in the airplane consisted of four other major experimental subsystems: three flight control computers, a navigation and guidance system, an electronic display system, and a data acquisition system[1]. Navigation, guidance, and flight control computer software were hosted on a MicroVAX computer, while the display software was hosted on another MicroVAX computer. An integrated Air Data Inertial Reference System (ADIRS) unit was used to provide accelerations, velocities, airplane position, and standard air data information to the research system. An experimental digital data bus called the Digital Autonomous Terminal Access Communications (DATAC) was developed to provide transport data from various sensors and research pallets throughout the airplane. The DATAC was a copper-based physical communication media which was later commercially recognized as a prototype of the ARINC 629 standard and was used as a computing network between research subsystems.

A development, verification, and validation laboratory, known as the Experimental Avionics Systems Integration Laboratory (EASILY), was used to develop, integrate, and preflight-validate the hardware and software systems for a TSRV flight test. The EASILY was equipped with VAX 4000-200 computers to provide simulation modeling, flight management system software, data acquisition functions, and graphic display generation. Interface units for testing signals associated with the ADIRS unit and the DATAC unit were also installed in the EASILY to mimic the architecture onboard the TSRV airplane.[2]

A ground-based simulator, similar to the AFD in the TSRV airplane, was built and used to conduct simulation-only or simulation and flight test research sponsored by the ATOPS program. The TSRV simulator was equipped with eight electronic monitors,

---

* *Computer Engineer, Systems Development Branch*

each with 6.5-in square viewing area. Two monitors were mounted in front of each pilot to display flight guidance and situation information and four touch-screen equipped displays were mounted in the center panel for engine information and multi-function tasks related to flight research operations. The TSRV simulator cab had an out-the-window visual scene capability, two two-axis programmable side stick controllers, two spring-loaded rudder pedals with toe brake systems, a voice actuated command system (VOTAN), and two Control Display Units (CDUs) installed. Simulation modeling was hosted on two Convex computers, and the flight instrumentation displays were generated by a Eagle 1000 Calligraphic Raster Display System (CRDS) from Terabit Computer Engineering. Communications between the Convex computers, the CRDS, and the simulator cabs were accomplished through a Computer Automated Measurement and Control (CAMAC) network using a high-speed fiber optic link configured in a star network between computers and simulator sites.[3]

Due to the differences between host computers, graphics systems, and communication architecture in the TSRV simulation and in the flight research system on the TSRV airplane, FORTRAN programmed software created at one facility could not be directly reused at another facility. This resulted in two separate developments, one for the simulation and one for the airplane, to be done to for the same set of research requirements and testing. These differences in facilities also caused difficulties for correlation of simulation data with flight test data. Langley Research Center decided to make major upgrades to both their ground-based simulation facilities and the flight research systems to be able to support the research requirements for current and future major NASA aviation programs. A simulation-to-flight concept was developed to improve the efficiencies of simulation and flight-testing. The simulation-to-flight concept used the rapid advances in computer and display technology and was based on using common hardware, software, and procedures for both simulation and flight-testing. This report will describe the system level perspectives of the simulation and flight facilities as background information followed by detailed descriptions of the common software for these facilities. The descriptions focus on the key element, the commonality between these facilities, which makes the simulation-to-flight concept successful.

Transport Research Facilities

Overview

The simulation-to-flight concept was the basis used during the design, development, and implementation of the Transport Research Facilities (TRF). The TRF are comprised of the Airborne Research Integrated Experiments System (ARIES) B-757 airplane, the Research System Integration Laboratory (RSIL), the Flight System Integration Laboratory (FSIL), the Cockpit Motion Facilities (CMF), and two simulators--the Integration Flight Deck (IFD) and the Research Flight Deck (RFD).

Inside the ARIES B-757 airplane is the Transport Research System (TRS) which is the electronic system, located in the cabin of the B-757, that drives the Flight Deck Research Station (FDRS) located on the left-hand side of the cockpit. The electronic flight displays in the FDRS are driven by the TRS.

The Cockpit Motion Facility (CMF) is a building that contains spaces for four simulator cabs including the RFD and IFD simulators, a motion platform, a lifting crane to move the simulator cabs on and off of the motion platform, the RSIL, and the interface between the simulator cabs, the RSIL, and the computers. The RSIL is a laboratory that contains a duplicate of the Transport Research System contained on the ARIES B-757 airplane. The IFD and RFD simulators and the RSIL are used to support research program requirements and to develop and validate hardware and software needed for flight test purposes onboard the ARIES NASA B-757 airplane.

The Flight Systems Integration Laboratory is located in the hanger adjacent to the ARIES B-757 airplane. The FSIL is used for integration and maintenance of the TRS on the airplane.

The simulation-to-flight concept used to design and develop the TRF was based on past experience and the research community's desire for a streamlined simulation-to-flight process. One of the major design goals of the simulation-to-flight concept was for the transport research system onboard the ARIES B-757 to readily accept software from the simulation environment to increase the efficiency and reduce the cost of implementing and conducting research tests. To satisfy this goal, common hardware and software were implemented in the research systems onboard the ARIES B-757 airplane, in the test labs, and in the simulators. This common system is the TRS. The simulation-to-flight vision of conducting research from concept formulation on a workstation to ground-based testing with a simulator, then to flight test onboard a research aircraft is illustrated in Figure 1.

The simulation and flight software in the TRS was developed using an Object-oriented (OO) methodology for designing and programming modular and robust

Simulation-to-Flight Vision

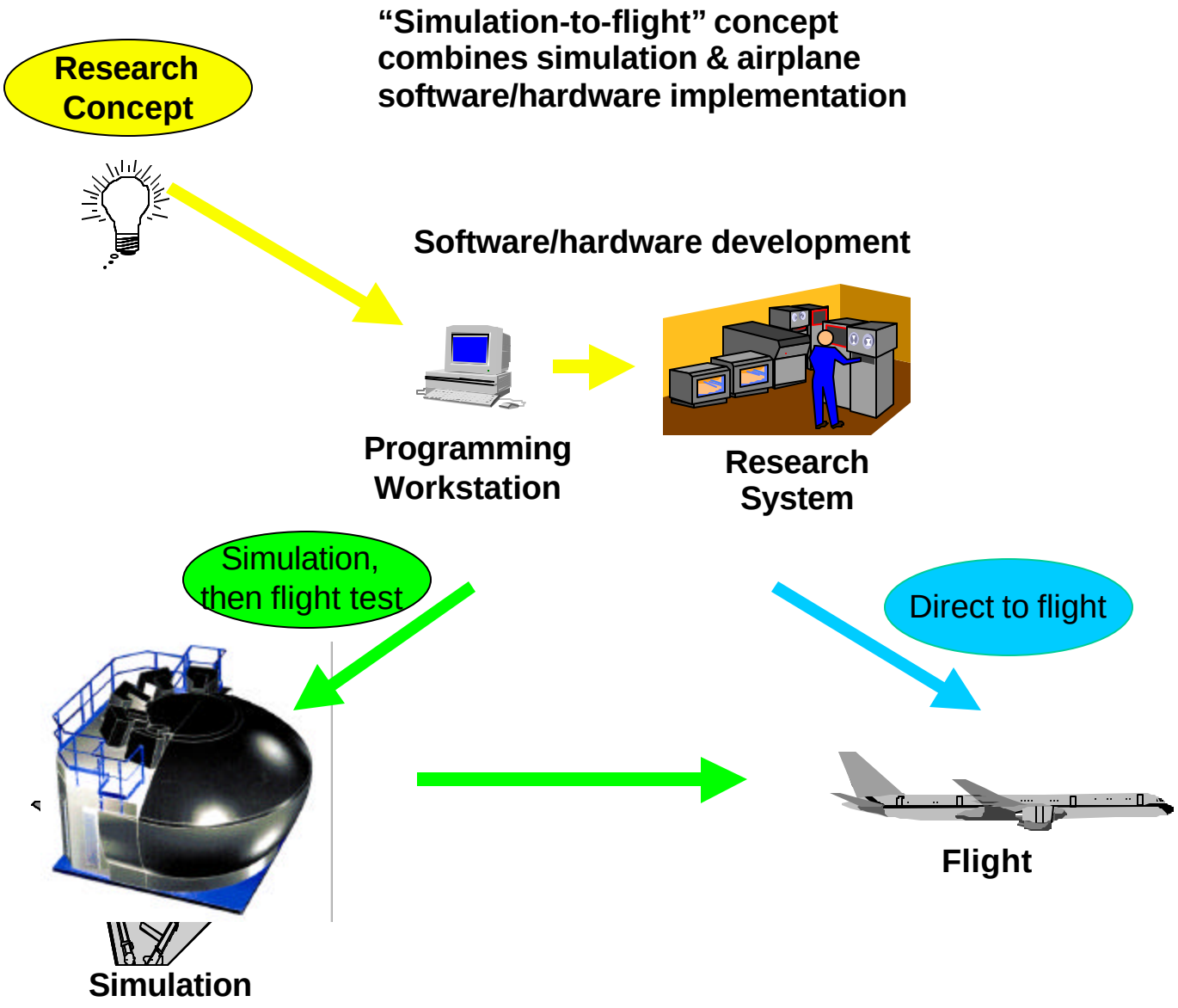


Figure 1. Simulation-to-Flight Vision

software. This OO methodology produced software that was easy to maintain and reuse, resulting in an increase in the reliability and development productivity.

Airborne Research Integrated Experiments System (ARIES) B-757 Airplane

Figure 2 is a recent picture of the NASA ARIES B-757 airplane. The ARIES B-757 airplane was the second B-757 airplane built. Its first flight was in March 1983. The Boeing Company used this airplane for the initial certification of the 757's. The airplane was then flown by Eastern Airlines as a revenue-flight aircraft. The airplane was purchased by NASA from the Eastern Airlines Bankruptcy estate in 1994.

After acquiring the airplane, NASA designed and implemented a research system in the cockpit and cabin of the airplane to support flight testing for the next 20 years. The type of testing envisioned included concept developments for increasing the safety of commercial jet transportation, increasing airport and airway system capacity, and to increase the United States' economic growth relating to the air transport system.

Flight Deck Research Station (FDRS)

The FDRS has two dedicated flight instrument CRTs (an Attitude Director Indicator--ADI and a Navigation Display--ND), a Head-up display (HUD), control panels for the dedicated flight displays, and a Control Display Unit (CDU) for the research system flight management computer (FMC).

Other instrumentation and controls used in the FDRS are unmodified, conventional B-757 flight deck equipment. This equipment includes all other flight instruments and controls (including the airspeed, altimeter, vertical speed indicator, standby instruments), the engine indication and crew alerting system (EICAS), mode control panel (MCP) controls and logic, manual- and auto-throttles, autopilot, and manual flight controls. The test subject sits in the left-hand seat and the pilot-in-command (PIC) sits in right-hand seat. A second safety pilot will normally sit in a flight deck jump seat to assist and monitor flight operations.[4]

Transport Research System (TRS)

Figure 3 shows the high level view of the FDRS and TRS onboard the ARIES B-757. The TRS is composed of various subsystems including a Silicon Graphics Incorporated (SGI) Application Onyx computer, an input and output (I/O) concentrator, a flight management computer subsystem, a data acquisition subsystem, a Global Positioning System (GPS) and Differential GPS subsystem, a data link subsystem, and a Shared Common Random Access Memory Network (SCRAMNet+). The SCRAMNet+ is a high-speed fiber optic replicated shared memory communication network manufactured by the Systran Corporation. Each TRS subsystem is installed on a pallet in the ARIES cabin. Different subsystems of the TRS interface with the SCRAMNet+ through VME-based chassises. Each VME chassis consists of a Motorola PowerPC MVME-1604 embedded processor board, which utilizes a 133 MHz MPC604 PowerPC processor, running VxWorks 5.3 as the real-time operating system.

The Application Onyx computer is used to process research applications and to generate research displays in the FDRS. The I/O concentrator is the interface between the aircraft sensors and other research subsystems of the TRS. The I/O concentrator is a VME chassis consisting of a basic Motorola Power PC processor board, a Datum time and frequency module, a SCRAMNet+ module, analog input and output cards, discrete input and output cards, Condor ARINC 429 cards, and a VME bus analyzer. The Datum time and frequency module is used to provide an accurate time source derived from the Ashtech GPS receiver for TRS. The flight management computer subsystem also consists of a VME chassis, a Honeywell flight management computer Product Improvement Package (FMC PIP), a Smiths Industry CDU, an Ashtech GPS receiver, and two Fieldworks laptop PCs. The FMC PIP is used to provide navigation and guidance in the FDRS. The data acquisition system is used for recording and limited post processing flight-test data. The data acquisition system consists of an Intel Pentium III personal computer with a 133 MHz CPU.[5]

Research System Integration Laboratory (RSIL) and Flight System Integration Laboratory (FSIL)

Clones of the TRS were implemented in two ground-based laboratories, the Research System Integration Lab (RSIL) and the Flight System Integration Lab (FSIL), to provide flight software and hardware development, integration, validation, and maintenance capabilities under the same research system architecture of the ARIES B-757. A Simulation VME chassis exists to provide the TRS an interface with either the RFD or the IFD simulator, to facilitate an arena for end-to-end simulation and testing of applications migrating to the ARIES. A flight control computer (FCC) VME chassis is equipped with triplicate FCC's to accommodate ground testing of research applications with actual flight-rated FCC's or with a simulated version of FCC's from the simulation software. The RSIL and FSIL could also operate without any actual simulator cab by using a RSIL Operator's Station. The Operator's Station consists of an attitude display and a map

Figure 2. ARIES B-757 Airplane

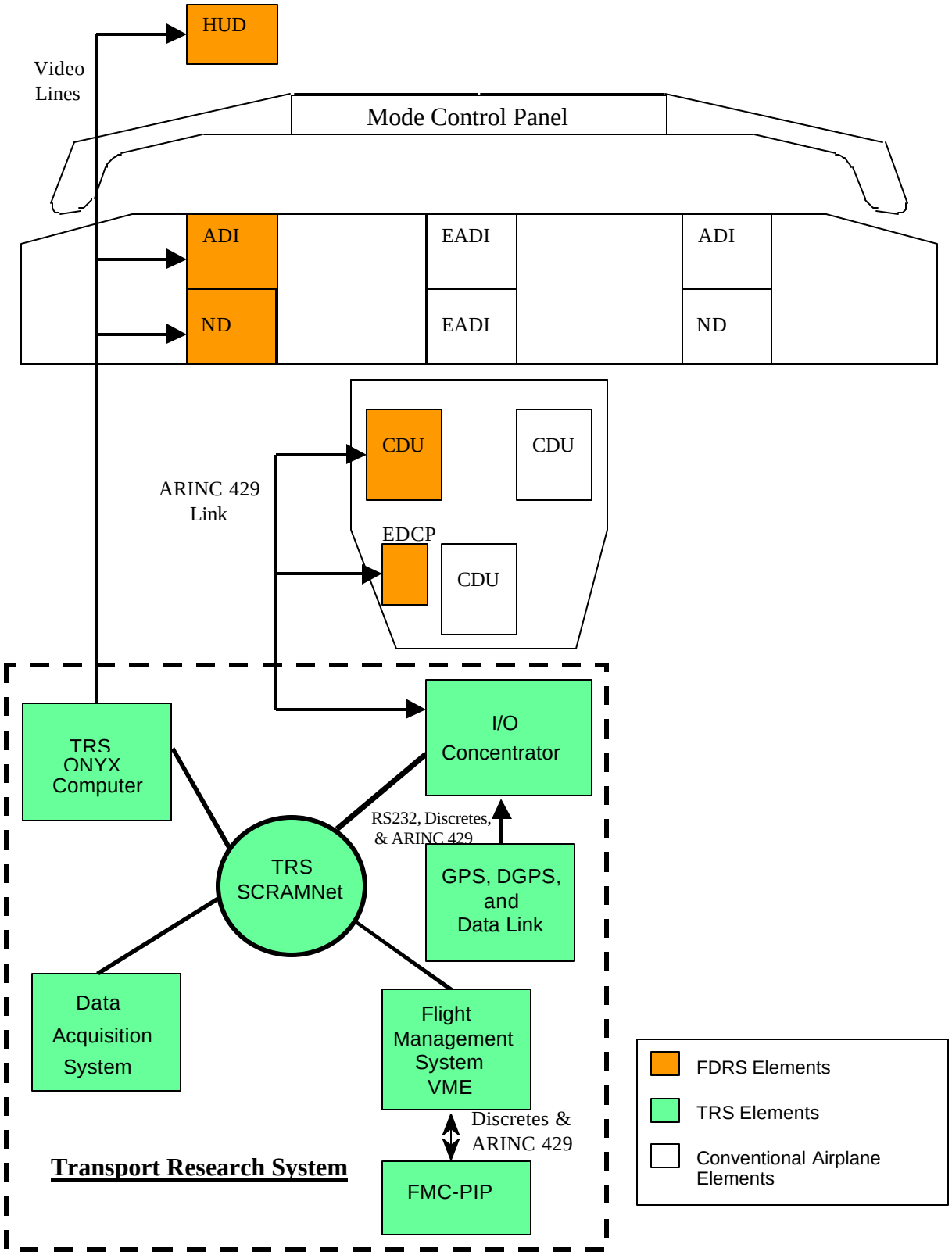


Figure 3. Flight Deck Research Station and Transport Research System Onboard ARIES

display, a Mode Control Panel to select the autopilot and flight guidance options, and an electronic display control panel to perform an engineering checkout. Figure 4 illustrates the configuration of the RSIL with the IFD or RFD simulator in the loop.

Research Flight Deck (RFD) and Integration Flight Deck (IFD) Simulators

The RFD and the IFD simulators are used to provide simulated flight deck environment and functionality for researchers to conduct pilot-in-the-loop aviation experiments. The Research Flight Deck (RFD) simulator cab was designed to represent an advanced, two-crewmember, subsonic jet transport airplane. The cab layout was based on the "best" components contained in the Boeing 777, 747-400, and MD-11, the Airbus series of airplanes, and the NASA B-737 Transport Research System Vehicle airplane.

The RFD cockpit contains eight "D-size" CRT displays, hydraulic side-stick controllers that are back-driven by the autopilot system, and a flight management system with two control display units. Subsystems and control panels are based on those of a B-757 airplane. The cockpit is designed to be modular so that cockpit configuration may be changed appropriately to satisfy research requirements.[6] Figure 5 is a recent picture of the RFD simulator.

The Integration Flight Deck (IFD) simulator cab is an engineering cab designed to represent the flight deck of the NASA ARIES B-757 airplane. The cab is populated with flight instrumentation, including the overhead subsystem panels, to replicate the B-757. Simulation software is hosted on a Simulation Onyx which communicates simulated aircraft responses and pilot inputs with the TRS in RSIL or FSIL via the SCRAMNet+. The IFD cab is used for flight testing development for the ARIES B-757 and for aircraft system integration studies. Figure 6 is a recent picture of the IFD simulator.

Both the IFD and RFD cockpits contain a "Panorama" visual out-the-window display system. This system provides a 200-degree by 40-degree visual out-the-window display to add realism to piloted experiments. Databases for nine different airports that contain other airplanes, trucks, and service vehicles found at airports can be projected dynamically on the out-the-window scene. Other airports may be added if required to satisfy research needs.

Software System

Before 1995, the simulation software was programmed in the FORTRAN language. The structure of this software architecture was procedural in nature. Software that dealt with multiple vehicle simulations were hard coded with a fixed number of vehicles and vehicle types to reduce the complexity in developing and maintaining a flexible multiple vehicle simulation. The desire to improve the efficiency in simulation development, productivity of the software developers, and the flexibility of operating multiple vehicle simulation in a real-time computing environment led to the evaluation and development of a simulation software framework that was designed in OO fashion and programmed in C++ language. This software framework was named the NASA Langley Standard Real-Time Simulation in C++ (LaSRS++).[7]

The simulation-to-flight concept requirement of using common software and hardware for all TRF called for the need to produce software that was flexible and that could be adapted or reused for multiple purposes. The successful evolution of LaSRS++ simulation software was timely for implementing this design approach. OO designed software could be used to hide the hardware details behind a common class interface. Different hardware interfaces could be easily substituted and used with the same software interface. This is the familiar OO approach of 'programming to an interface'. This approach involves abstracting implementation details and isolating them from the application. Also, with this approach, software components could move easily from the unit test stage to simulation, and then to a flight test.

A rigorous software development process has been utilized to ensure that software is thoroughly designed, developed, and tested before using it for a flight test operation. The objective of this process is to provide optimum mission and safety assurance. Figure 7 shows the software development process that includes the verification and validation process. Several commercial-off-the-shelf software products are used as development tools throughout the software development lifecycle. The software design tool, Rational Rose, from the Rational Software Corporation, is used to visualize and construct software design artifacts through Unified Modeling Language (UML). Other tools from the Rational Software Corporation include Rational ClearCase for software configuration management, Rational Purify for debugging run-time errors and memory management issues, and Rational Quantify for highlighting performance and identifying bottlenecks and untested code in applications. SGI's ProDev Workshop is also used as development tool for similar purposes. Code documentation is implemented by the shareware Doc++ software tool. The TRS software system consists of common and facility specific software. Facility specific software provides

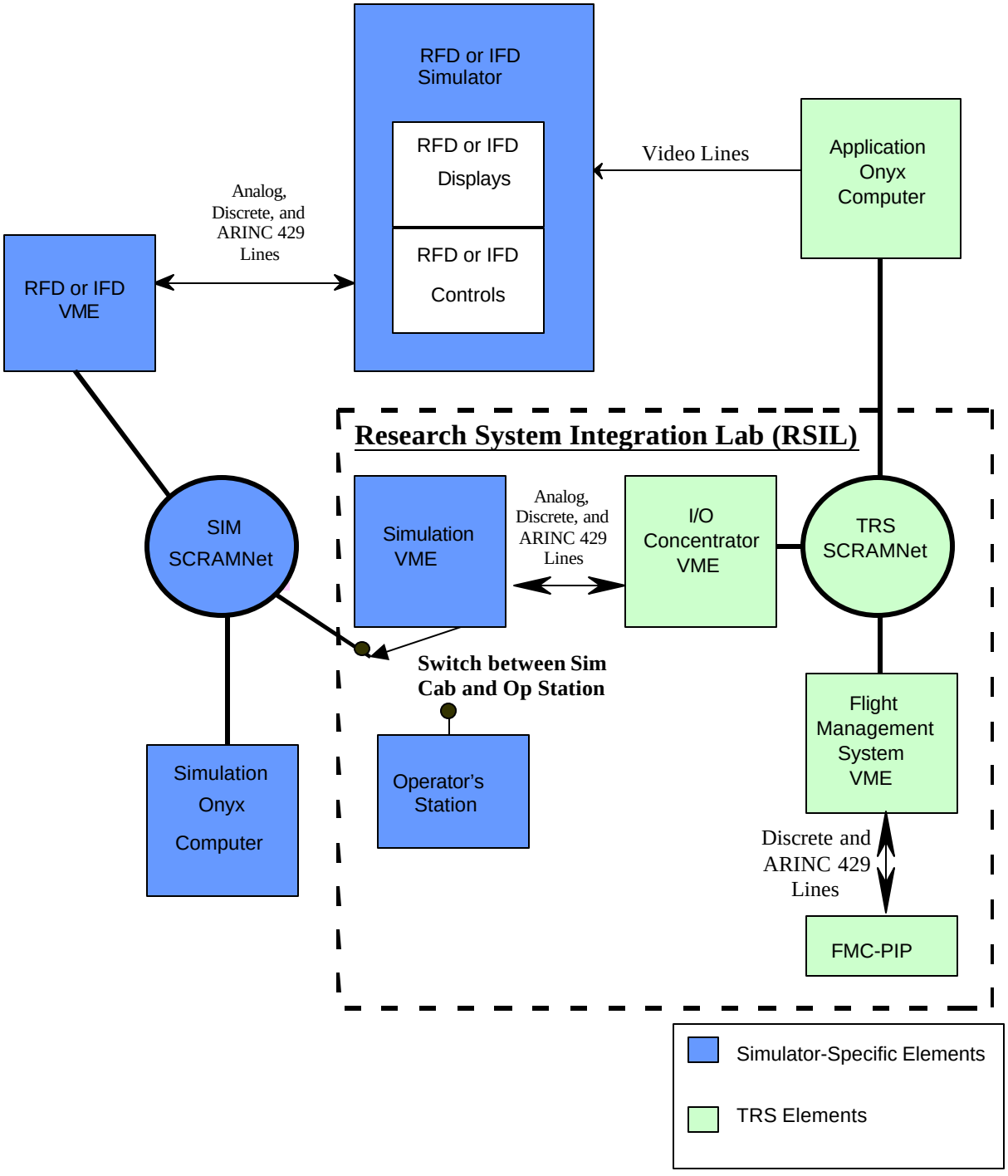


Figure 4. TRS Residing in RSIL with RFD or IFD for End to End Testing

Figure 5. Research Flight Deck Simulator

Figure 6. Integration Flight Deck Simulator

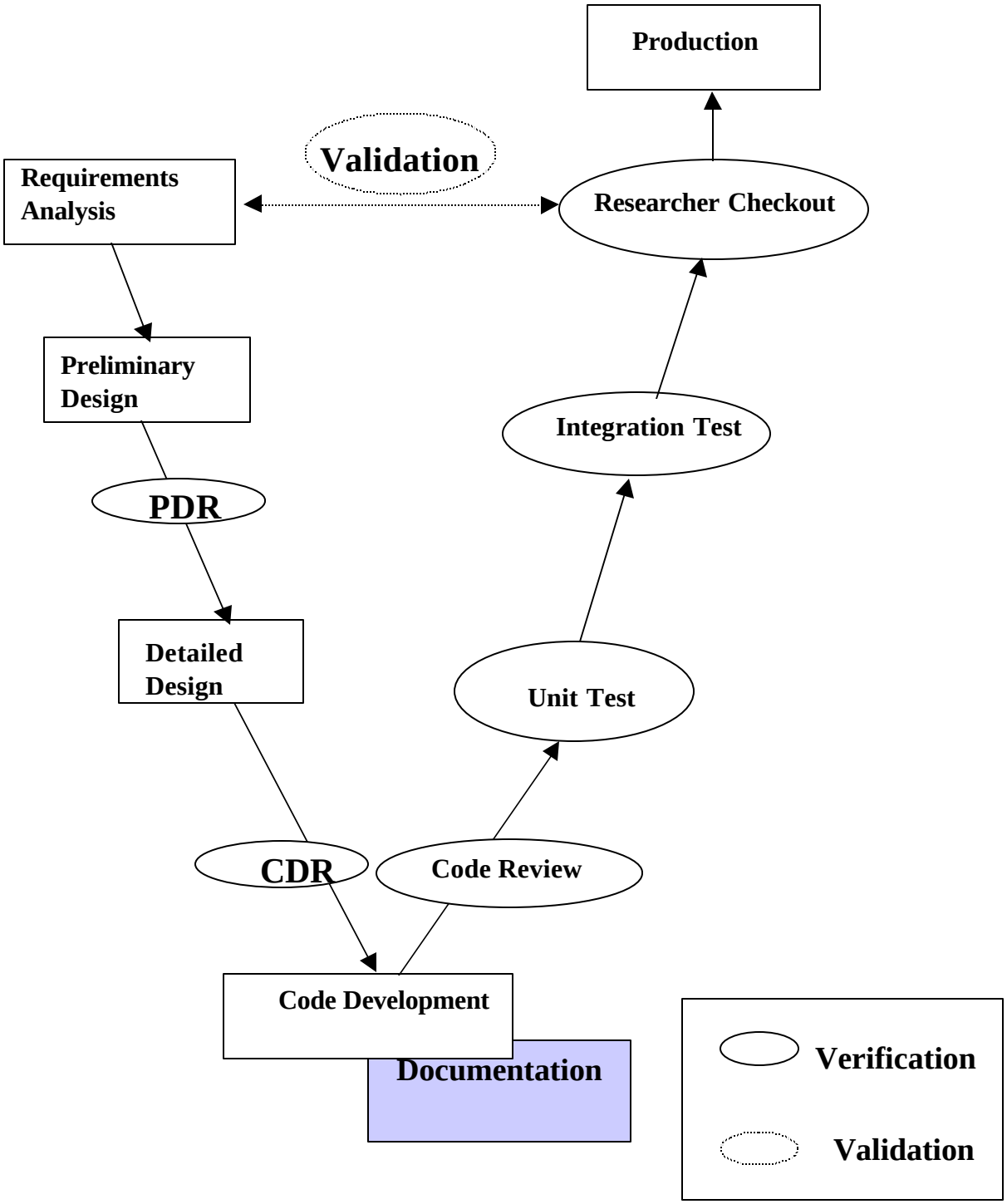


Figure 7. Software Development Process

simulation functionality exclusively. Figure 8 shows the Work Breakdown Structure of the software system.

TRS Software Design

The following designs discussed and depicted have many details eliminated to emphasize the prominent features. They demonstrate that a solid but flexible underlying communication framework has resulted in somewhat consistent designs. Object oriented purists would point out many procedural characteristics in the designs, a fact that is undeniable. However, throughout the lifecycle, the TRS has met hard implementation schedules resulting in a more pragmatic approach. Certain particular design characteristics are prevalent because they naturally de-coupled class relationships making them easier to migrate from one environment to another. The containment by aggregation rather than derivation and the use of mediation also makes incremental testing easier. Other design patterns used are factory, builder, and singleton. A formalized approach to these patterns may be found in the book *Design Patterns* by Gamma, et al[8].

LaSRS++ Framework Communication

The simulation systems and the TRS high speed I/O are built on Systran Corporation's SCRAMNet+ high speed network and Motorola Power PCs operating in VME back-planes using the WindRiver VxWorks real-time operating system. Each SCRAMNet+ node is configured with two megabytes of memory and fiber optics media adapters. The baseline TRS consists of two VME crates with Power PC MV1604 processors and a Silicon Graphics Onyx computer connected in a network using a Systran Quad Switch. The VMEs are populated with various I/O cards and each serves a specific function. The VME on the IO Concentrator receives ARINC digital signals from the various basic B-757 systems buses (Inertial Reference System/Air Data Computer/EICAS/Radar Altimeter) and is the primary source of data for ARIES B-757. A second VME provides communications with a Honeywell FMC which is separate from the basic B-757 systems FMCs. The Onyx contains eight 195 MHz MIPS R4400 CPUs with three graphics pipes connected to a Multi Channel Option that provide three video outputs from each pipe.

To avoid hard-wired addresses for variables and the conflicts that arise in connecting arbitrary combinations of SCRAMNet+ nodes with different configurations, the SCRAMNet+ memory is managed through an allocation scheme that supports initialization from configuration files and runtime allocation. The mechanism is a simple table in memory that associates block names (ASCII) with addresses for data blocks in SCRAMNet+ memory. A special system state area contains information about the state of the system. When the controlling computer (system master) has created the table, the system state area is changed from *start* to *real-time.* Server and client SCRAMNet+ nodes are programmed to monitor the system area, notice the state change, and proceed to search the table for blocks containing information for them.

Synchronous Communication

The framework system software is designed to operate the VME processors as I/O subsystems that are configurable at runtime with the Onyx as the system master. An example of this configurability would be the addition of an analog-to-digital converter. A properly jumpered card would be installed in a VME subsystem and the data block added to the configuration file read by the main host. The only software requiring modification would be the main application on the host. This architecture, used successfully throughout the Langley simulation facilities, is also used in the TRF environment.

An important characteristic of this system is that the synchronous I/O data (such as analog and discrete) is cycled at the rate and time demanded by the main application. The main program on the SGI iterates at a stable 50 Hz and provides frame start times within one millisecond and thus, is suitable for numerical integration and digital filter implementations. The SynchronousChannel software hierarchy used by the main application is shown in Figure 9. A SynchronousChannel derived class such as AnalogToDigitalChannel supports analog input for one or more physical modules on a particular I/O subsystem. Each module has sixteen inputs. The base class provides common services while derived classes provide details of processing for a specific type of module. For an installation with two modules, input lines are accessed individually via the derived class using integers from 0 to 31. The ChannelBuilder is a factory class that may optionally be used to create SynchronousChannel objects and performs the details of acquiring the SCRAMNet+ blocks, constructing objects and processing error conditions.

Asynchronous Communication

Asynchronous communication types such as RS-232 and ARINC 429 protocols are often required in simulations and the aircraft environment. Communication channels for asynchronous type data are also available using software implemented specifically for that purpose. Figure 10 is the design of the framework for an asynchronous communication hierarchy that is used extensively. These template classes provide a read/write interface (similar to UNIX system I/O) to queues established in SCRAMNet+. The ScramnetMemoryBlock, which is a specialized

1.0 Common Software

- 1.1 Systems
  - 1.1.1 Hybrid GPS
  - 1.1.2 Synthetic Instrument Landing System
- 1.2 Interfaces
  - 1.2.1 Synchronous Communications
  - 1.2.2 Asynchronous Communications
  - 1.2.3 SCRAMNet Communications
  - 1.2.4 Experimental Display Control Panel Interface
- 1.3 IFD Displays
  - 1.3.1 Attitude Director Indicator Display for Left Side
  - 1.3.2 Navigation Display for Left Side

2.0 Facility (Simulation) Specific Software

- 2.1 Systems
  - 2.1.1 Aerodynamics Model
  - 2.1.2 Propulsion Model
  - 2.1.3 Fuel
  - 2.1.4 Engine Model
  - 2.1.5 Landing Gear Model
  - 2.1.6 Effectors/Actuators/Servo Models
  - 2.1.7 Hydraulics Model
  - 2.1.8 Mass Model
  - 2.1.9 B757 Mode Control Panel
  - 2.1.10 Thrust Management System
  - 2.1.11 Auto-pilot/Flight Director Model
  - 2.1.12 Flight Controls System
  - 2.1.13 GPS
  - 2.1.14 Radar Altimeter
  - 2.1.15 Air Data Computer
  - 2.1.16 DME
  - 2.1.17 IRS
  - 2.1.18 VOR
- 2.2 Interfaces
  - 2.2.1 RFD Cockpit Interface
  - 2.2.2 IFD Cockpit Interface
  - 2.2.3 Visual Scene Interface
  - 2.2.4 Sound system Interface
  - 2.2.5 Head-down Display Interface
- 2.3 IFD Displays
  - 2.3.1 Attitude Director Indicator Display for Right Side
  - 2.3.2 Navigation Display for Right Side
  - 2.3.3 EICAS Displays (Upper & Lower)
- 2.4 RFD Displays
  - 2.4.1 Primary Flight Displays (Left & Right)
  - 2.4.2 Navigation Displays (Left & Right)
  - 2.4.3 EICAS Displays (Upper & Lower)
  - 2.4.4 Multi-functional Displays (Left & Right)
- 2.5 Onyx System Software
  - 2.5.1 Supervisor
  - 2.5.2 Device Driver

Figure 8. Work Breakdown Structure for B-757 Software System

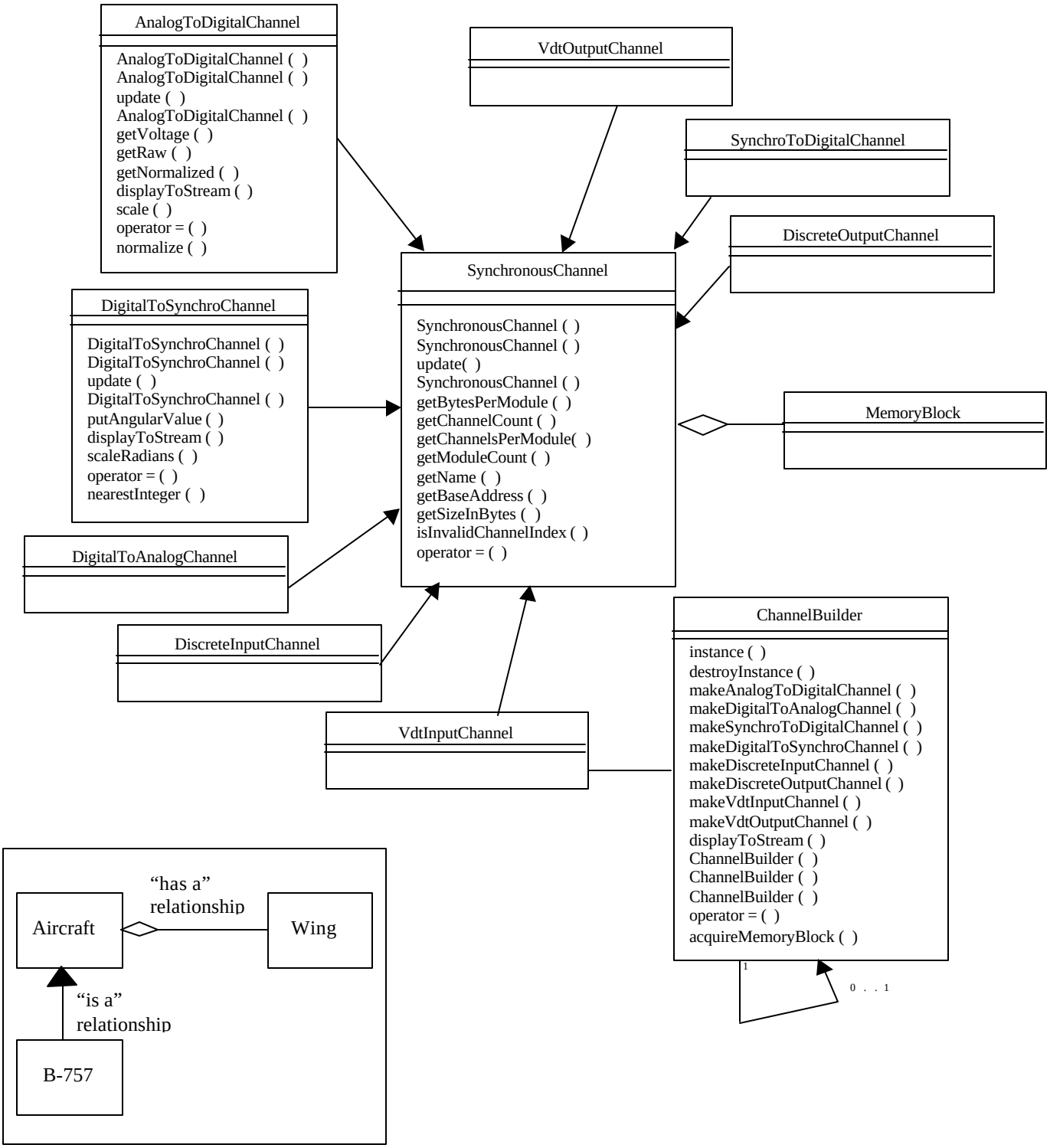
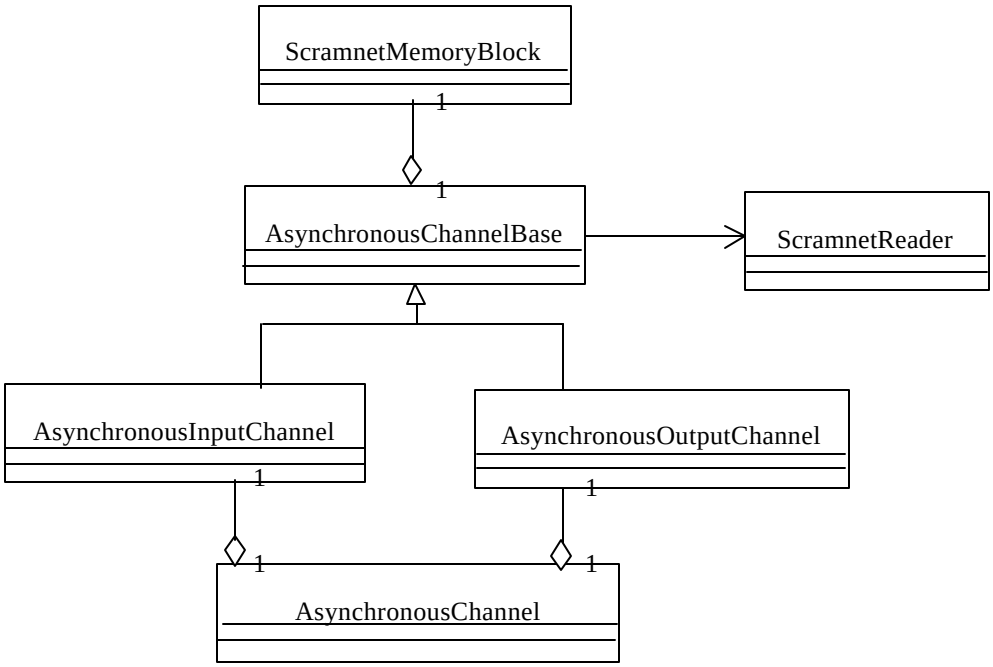


Figure 9. Synchronous Communications



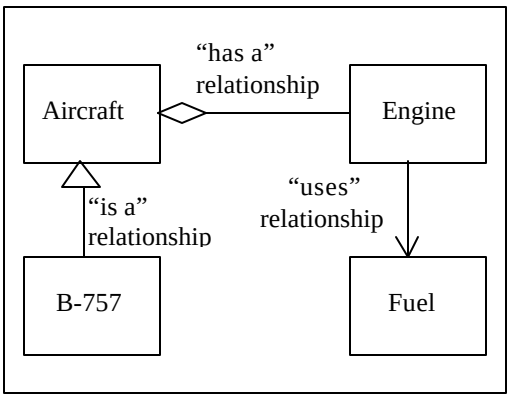


Figure 10. Asynchronous Communication Hierarchy Design

MemoryBlock, is used because it provides methods to transmit interrupt and receiver interrupt enable SCRAMNet+ memory locations.

A good application example would be the output half of a simulated Aircraft Communication and Reporting System (ACARS). The main application would construct an AsynchronousOutputChannel to an ARINC transmitter and invoke the ::write() method to transmit encoded data-link messages. During initialization, the I/O subsystem would acquire the ScramnetMemoryBlock from the SCRAMNet+ table, construct an AsynchronousOutputChannel, and spawn a task that sleeps until an interrupt is generated by the main application ::write(). The interrupt causes the task to wakeup, read the data in the channel, and transmit the data on the selected transmitter. AsynchronousChannels are often associated with, but not limited, to hardware I/O.

Communication using the ARINC 429 protocol[9] is an important capability for the TRS and for the integration of flight hardware into the simulation environment. The majority of data available on the 757 is from the digital system buses from the various basic B-757 systems. The LaRC simulators incorporate flight management computers, flight control computers, mode control panels, and various other devices that require an ARINC 429 protocol. Figure 11 depicts a group of classes built-up from framework communication that provides main applications the ability to establish a channel of ARINC 429 communication to an arbitrary node and device. The notion of a client/server relationship resulted in two primary components of the architecture being the ArincAllocator and the ArincServer. The ArincAllocator provides services to applications that convey the communication configuration to the I/O subsystem. Details for the communication are passed to the server through a SCRAMNet+ memory block specific to that purpose. The server task is spawned on the PowerPC I/O subsystem during system initialization and services requests for input and output by constructing and linking objects to accomplish the communication.

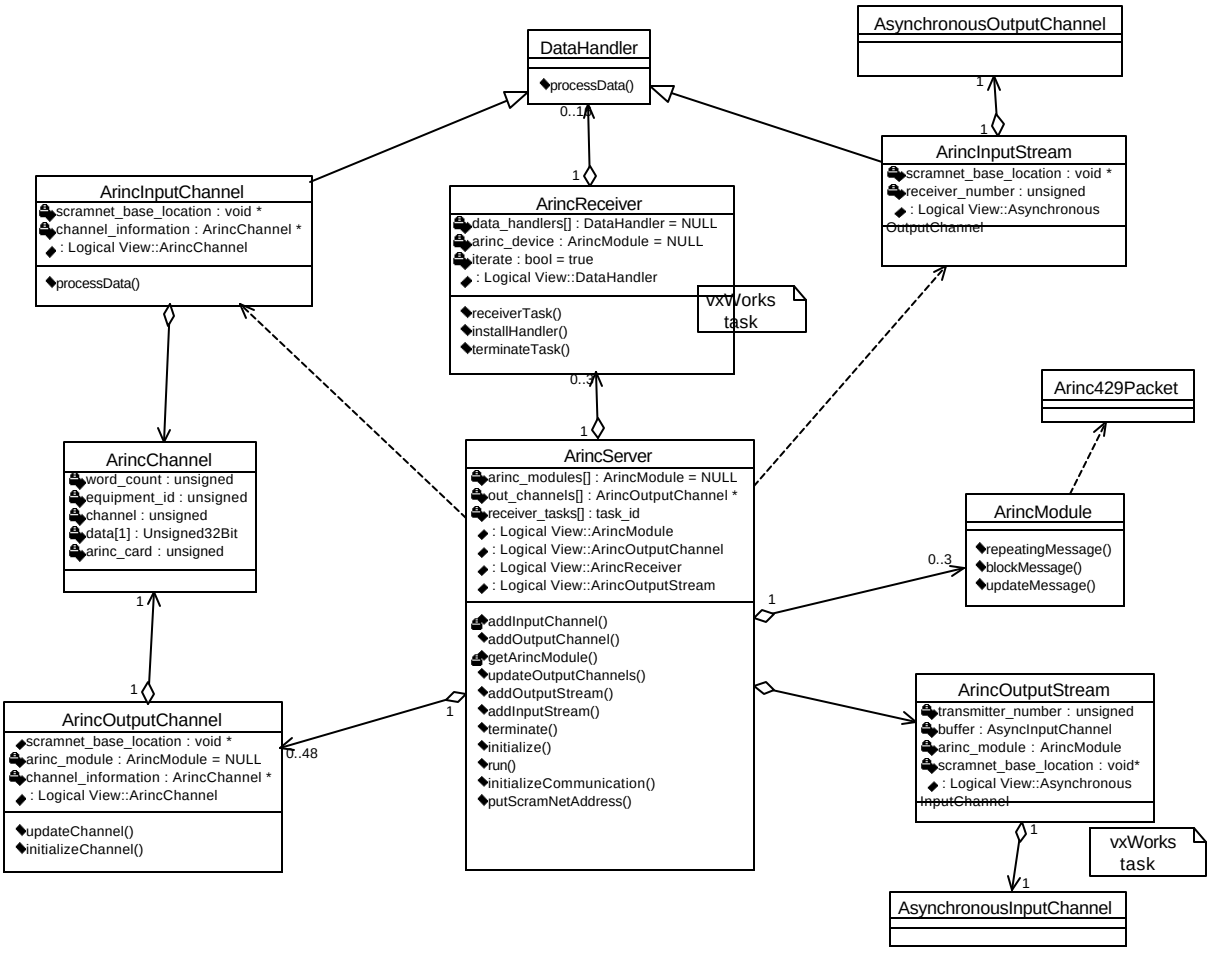
The design also attempts to keep the content of the communication separate from the mechanism. Other challenges for the design are the wide variety of ARINC specifications that use the ARINC 429 protocol for the underlying communication. Conversions of data to and from binary ARINC format are provided by conversion objects created with data from an ArincDatabase class. These objects have overloaded methods with float type arguments that return ARINC format words, and ARINC format arguments returning float types. The diagram in Figure 12 shows the ILSOutput class that is a typical ARINC bus implementation class. ILSOutput contains an ArincChannel by aggregation and uses ConversionFactory to obtain references to Convert objects during construction. Multiple ILSOutput objects may be created to receive and transmit within the same application.

TRS Application Design and Flight Test Projects

The primary baseline functions provided by the ARIES TRS software include primary and navigation flight deck displays, integration with a flight management computer, and a GPS based instrument landing system or GLS. The GLS is capable of providing guidance signals to the flight control computers during coupled approaches flown by the basic B-757 autopilot system. A key part of the GLS system is the implementation of a third order complimentary filter combining GPS position and inertial accelerations from the ships inertial reference system providing high rate position updates.

Joint Runway Friction Project[10]--The Joint Runway Friction Project was the first deployment using the new TRS on the ARIES B-757 and followed shortly after the TRS baseline instrument check flight. The role of the TRS on ARIES was to compute coefficients of friction, stopping distances, and kinematics for the accumulation of brake energy for two aerodynamic configurations. This first project encapsulated the various computations in a class with an interface for controlling inputs and outputs shown in Figure 13. An object of this type was created in AircraftBuses to which methods were added to supply inputs, as well as retrieve, display and record outputs. The aircraftBuses::update() method invoked the ::computeRunwayFriction() method as another internal helper method. The B-757 software simulation package and the RSIL lab were used to test the computation of the kinematics prior to local flight tests.

Airborne Information for Lateral Spacing (AILS)[11]--The Airborne Information for Lateral Spacing Project flight test followed and was the culmination a multi-year project jointly involving NASA Langley and Honeywell Incorporated. The premise for the flight test was alerting-algorithms operating in a red-label Traffic Collision Avoidance System (TCAS) computer programmed to receive Automatic Dependence Surveillance-Broadcast (ADS-B) aircraft state information from another aircraft on a closely-spaced parallel approach. A full simulation using the IFD simulator provided pre-recorded data of an aircraft on a parallel approach to the RSIL in addition to ARINC bus data simulating a Mode-S transponder and GPS



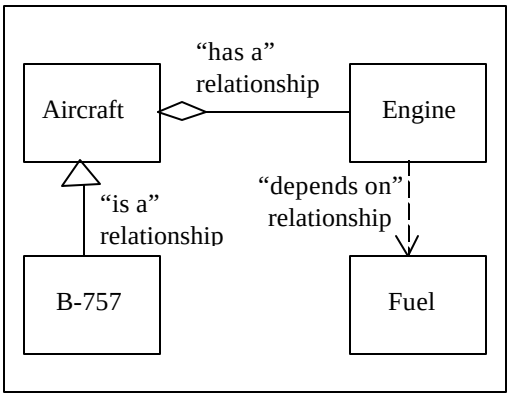


Figure 11. Subsystem ARINC Communication Components

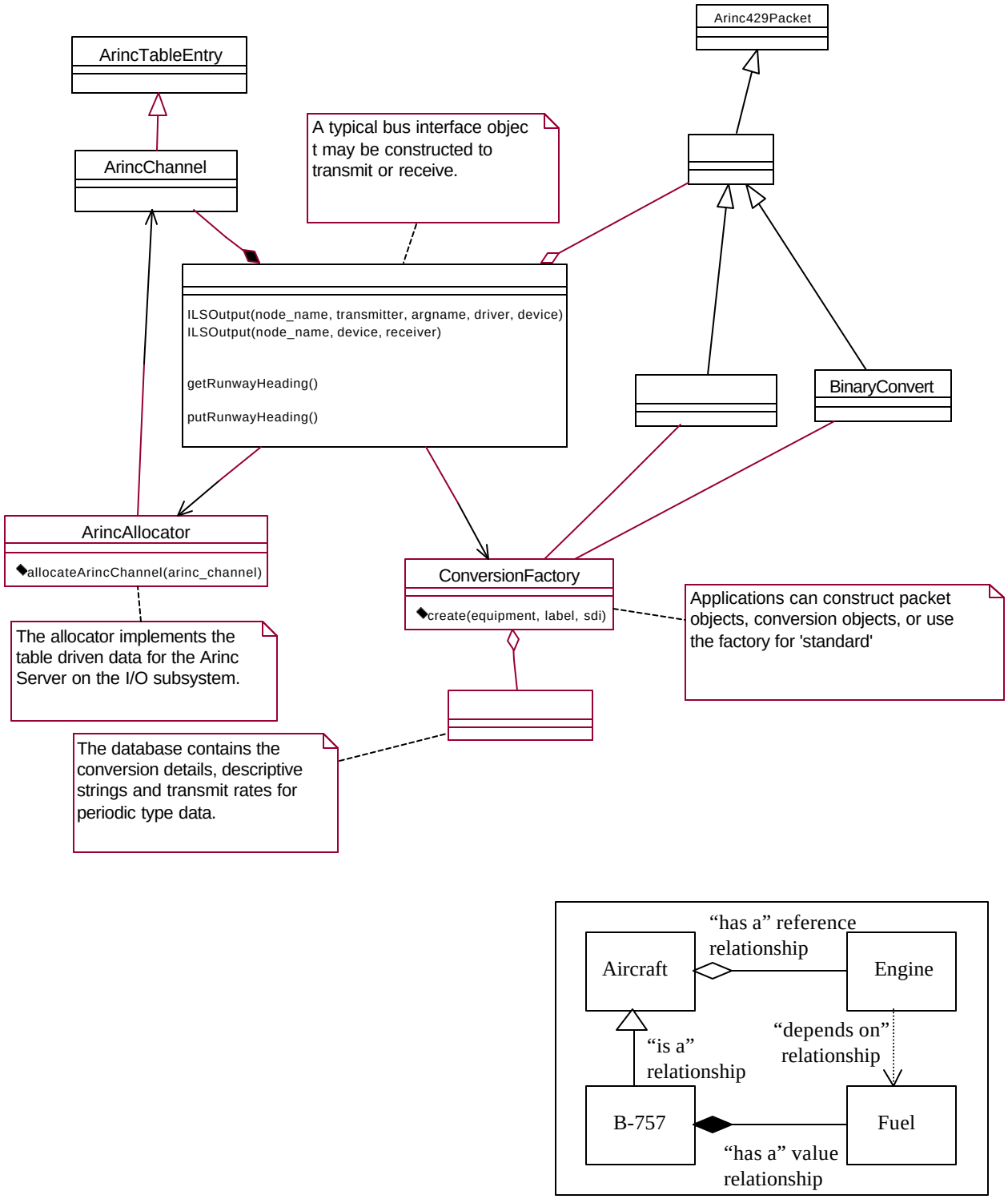


Figure 12. ILS Output Class, Another ARINC Communications Design

FlightMain Application

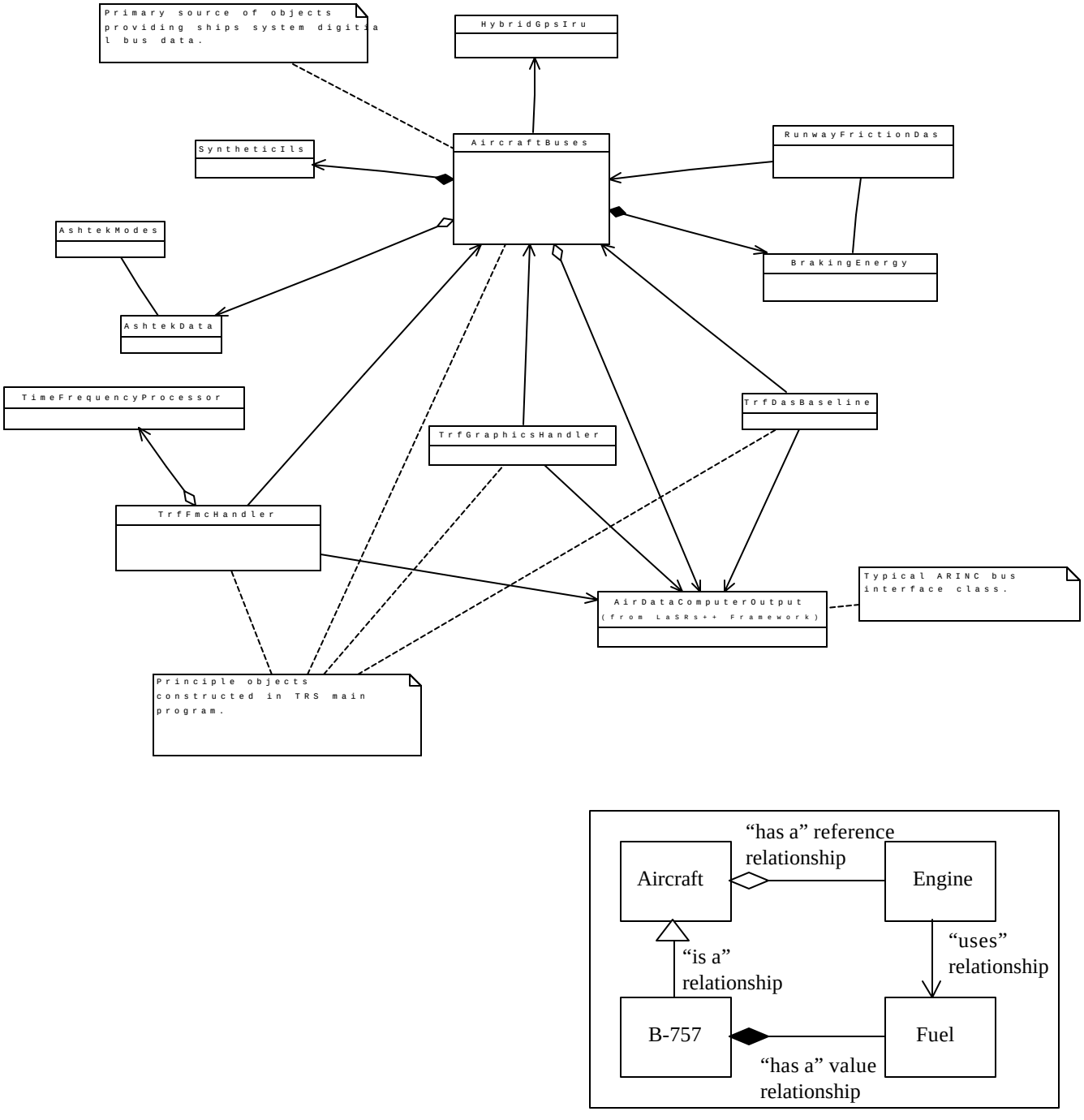


Figure 13. First TRS Baseline Design

Navigation Satellite System (GNSS). The simulation side of the RSIL was outfitted to transmit ADS-B messages to the TCAS line replaceable unit (LRU) computer using a radio frequency (RF) signal generator.

The TCAS inputs and outputs required for the flight test were connected to the TRS side of the RSIL. Figure 14 is the design used in the TRS and shows the AilsHandler containing four objects required for the periodic ARINC I/O with the TCAS each containing an ArincChannel. The TcasVapsInterface is a specialized class providing input processing of asynchronous 'track file' data from the TCAS for delivery to the Virtual Prototypes Incorporated's Visual APplicationS builder (VAPS) graphics process which in this case was a modified navigation display. The AilsHandler object was created in the main program at the same level as AircraftBuses for accessibility to the graphics and data recording components.

Year 2000 Flight Experiments

The level of activities for Year 2000 for the ARIES B-757 airplane was elevated and included multiple projects. Each of these projects were developed separately and then integrated and tested in the RSIL environment. There were two software releases operated during flight tests on ARIES in 2000. Adding several new projects, eliminating old projects and managing a release configuration became complex. The first effort to partition research projects from the baseline research system was the introduction of a new class, ResearchSystems, with the purpose of constructing and containing the project classes. Figure 15 illustrates the TRS design for Year 2000.

Community Noise Abatement Program--The B757 Community Noise Abatement Program or the Aircraft Noise Prediction Program (ANOPP) flight test was a rapidly developed project for flight test activities supporting ground-based sound instrumentation. Project development occurred early in the year parallel with another flight test project. The ANOPP flight test requirement was for the ARIES B-757 to fly accurate ground tracks, with specific vertical path trajectories over microphone stations in approach and take-off configurations. Additionally it was desired to control the take-off/go-around engine power to derated-thrust settings not available from the basic B-757 system thrust management computer. The challenge in this circumstance was completing the checkout of the simulated thrust management computer while testing the TRS implementation of the concepts to be used for lateral path and thrust management during the flight test. Figure 16 illustrates the ANOPP Design.

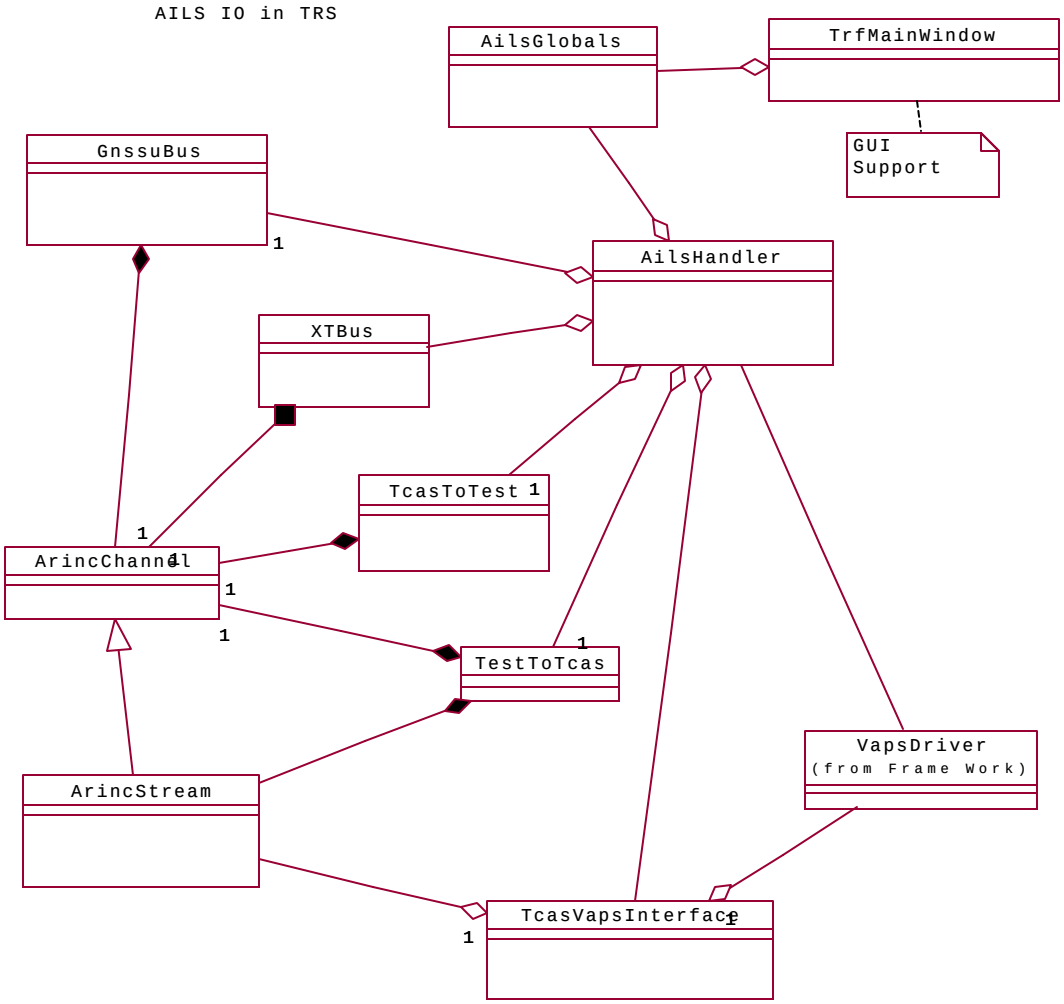
Runway Incursion Prevention System (RIPS) and *Hold Short Advisory Landing Technology (HSALT)*--RIPS and HSALT were sister projects that consisted of a head-down airport diagram and a multi-mode head-up display running as graphics processes on the Application Onyx computer and driven by the TRS from a common shared memory segment. The RIPS airport diagram depicted real-time up-linked airport traffic, ADS-B traffic, Controller Pilot Data Link Communication messages, and runway incursion alerts from two airborne sources and a ground source. The implementation relied heavily on the baseline communication mechanisms. The project was developed as a simulation-only project after which the software was transferred verbatim to the TRS.

Synthetic Vision Display Concepts (SVDC)--The SVDC project demonstrated photo-realistic and synthetic out-the-window imagery on a large format flat screen mounted on the instrument panel in front of the test subject pilot and a head-up display. The imagery, which was a separate software effort, was produced by PCs using high-end graphics cards. The TRS supplied data to the PCs that used SCRAMNet+ cards and a version of the framework SCRAMNet+ hierarchy ported to the Windows NT environment.

Weather Accident Prevention (WxAP)--The WxAP program uses data from the basic B-757 body-mounted accelerometers and digital system sensors as inputs to algorithms that compute in-situ turbulence metrics for correlation with experimental weather radar. The project was planned from the start to undergo initial implementation and testing in the simulation and then transferred to the RSIL/TRS environment. The design approach was to have the WxAP hierarchy mediated by a simulation-specific class and a TRS-specific class. Final testing prior to flight tests included recording data in the simulation and in the RSIL/TRS during the same run and co-plotting the results.

Year 2001 Flight Experiments

The activities for Year 2001 will include additional flight tests for SVDC and WxAP projects. The software development is largely the expansion of the requirements for the existing projects. However, during the interlude, a design was added to the TRS that assists in managing the flux of projects and reduces the workload when shifting the mix of projects for a software release. The design as shown in Figure 17 modifies the existing ResearchSystems class to include and Standard Template Library (STL) vector of references to ResearchProject objects. Projects are derived from ResearchProject and inherit the common interface on which ResarchSystems operates as it iterates on the STL vector. The details for the



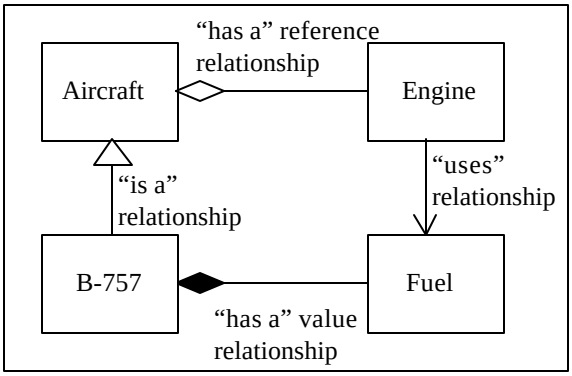


Figure 14. TRS Software Design for AILS

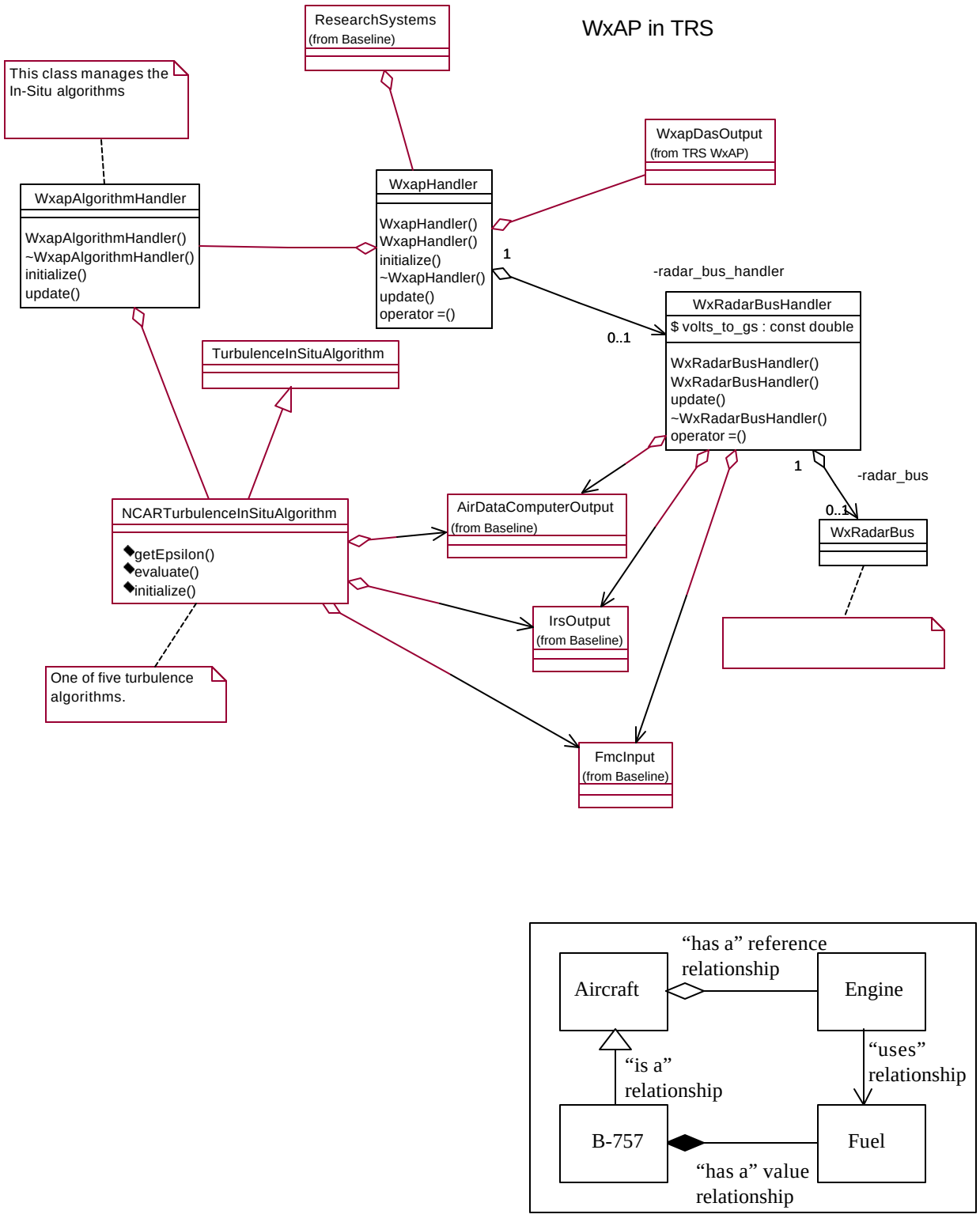


Figure 15. Year 2000 TRS Software Design for RIPS/HSALT/SVDC/WxAP

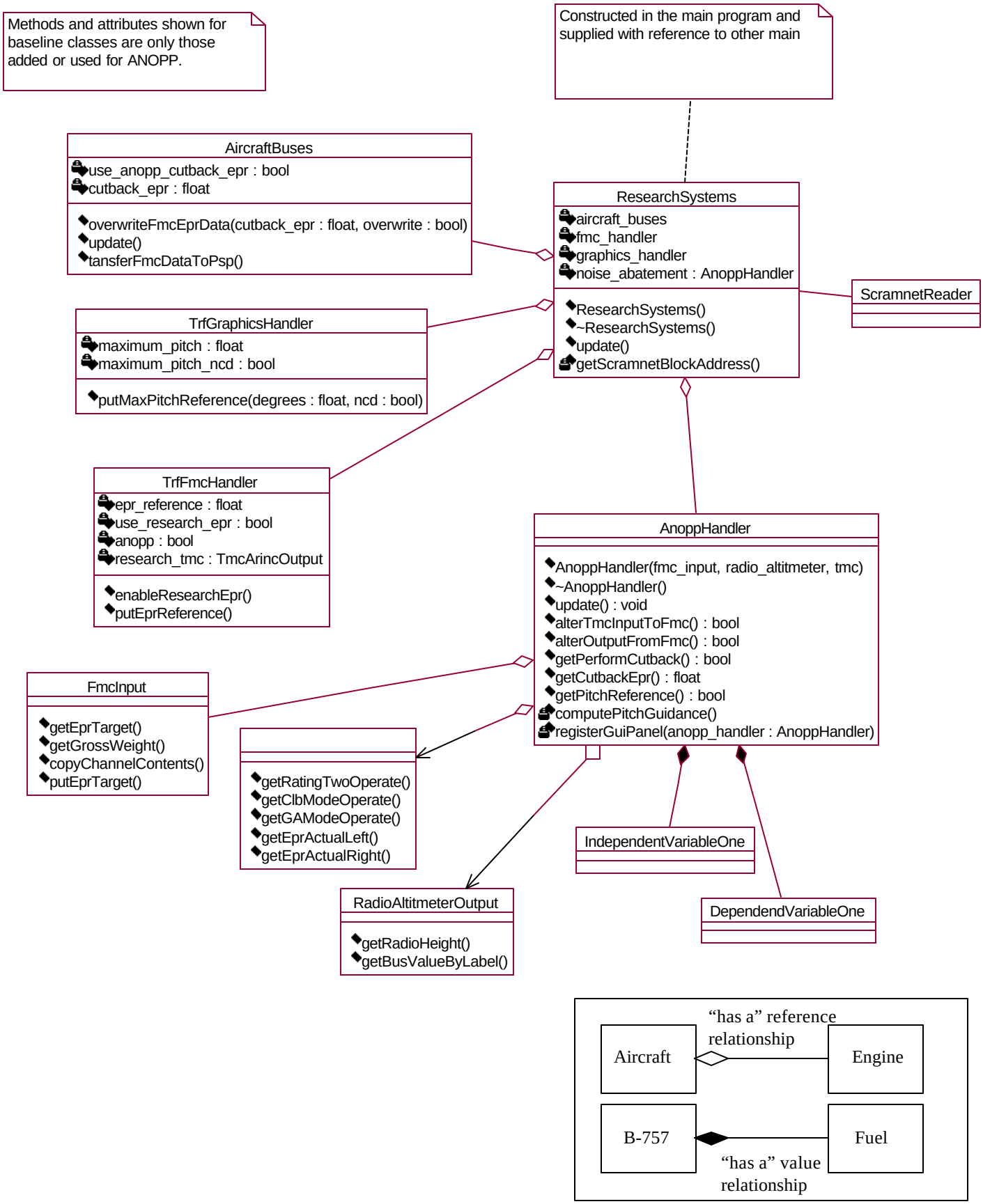


Figure 16. TRS Software Design for ANOPP

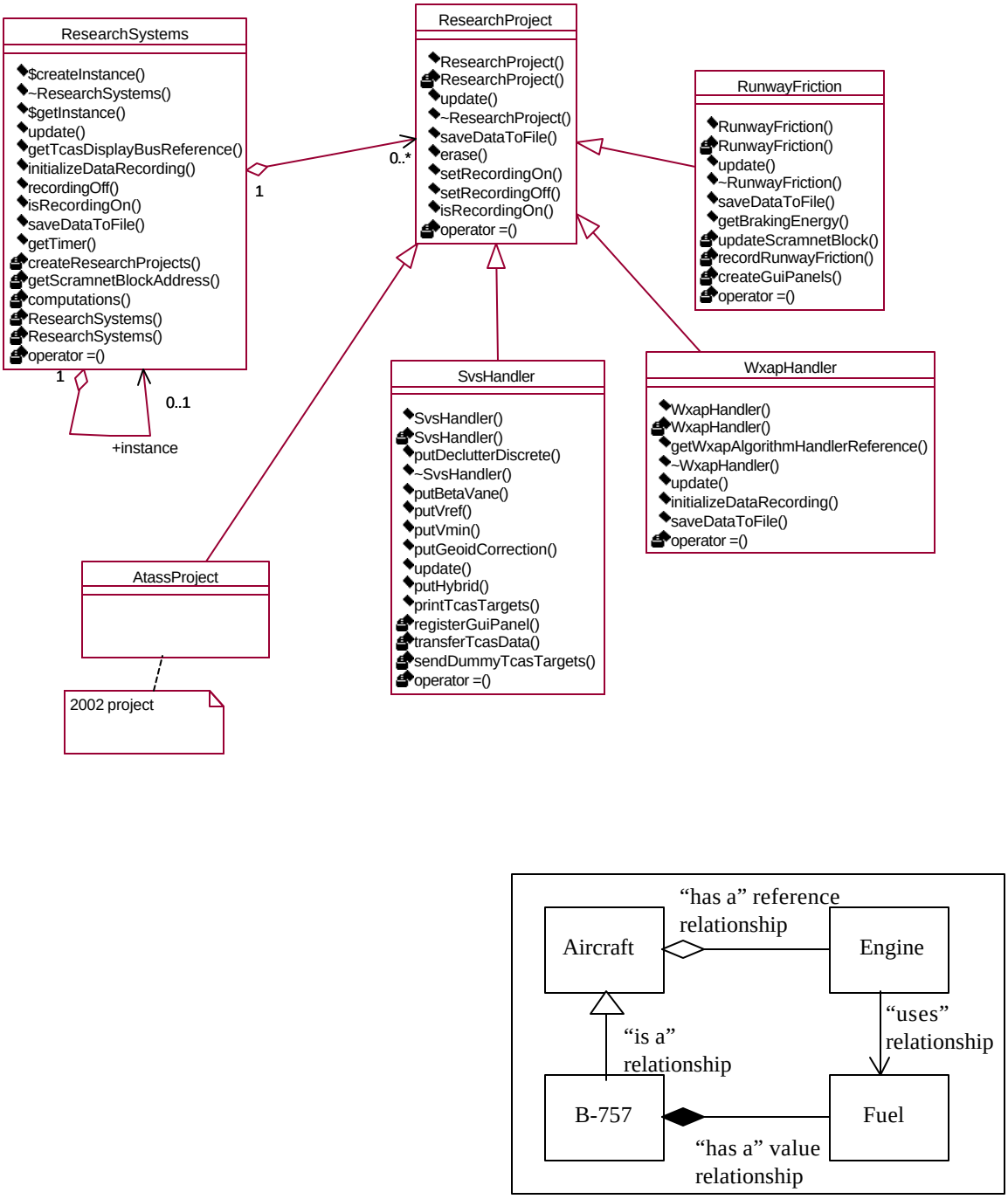


Figure 17. TRS Design with SVDC/WxAP Projects for Year 2001

construction of projects are isolated to the ResearchSystems::createResearchProjects which has implementation in a separate compilation unit. Projects are maintained in separate directories, with individual build scripts for project libraries. The TRS build script controls which projects are linked and constructed. The potential exists to configure runtime project instantiation from an initialization file or from command line options.

Conclusion

The process to satisfy the simulation-to-flight concept has been implemented effectively as evidenced by research results from various past experiments. The TRS software design has also gone through an evolution process to increase the flexibility and efficiency to support multiple projects during a single integrated flight experiment. The simulation-to-flight concept research processes will be continuously improved for long term success.

Acknowledgements

The authors gratefully acknowledge the significant contributions of Charlie Knox, Carey Buttrill, and Jacob Houck from the NASA Langley Research Center Airborne Systems Competency for providing valuable reviews and critics for this paper. The efficacious guidance provided by Simon Chung from the NASA Scientific and Technical Information (STI) Program Office in navigating through the Agency's STI systems is also greatly appreciated.

References

[1] Terminal Configured Vehicle Program, *Test Facilities Guide*. NASA SP-435

[2] Outlaw, Bruce K. E.: *Description of the Experimental Avionics Systems Integration Laboratory (EASILY)*. NASA TM 109072, 1994

[3] Cleveland, Jeff I.; Herndon, Sonia S.; Houck, Jacob A.; Kibler, Kemper S.; Meetze, Lemuel E.; and Simmons, Harold I.: *Real-Time Simulation User's Guide*. 1997

[4] Knox, C.E.: *The Requirements Document for the Transport Research System, Version 3.4.* NASA Langley Research Center, 1997.

[5] Fisher, Bruce D.; and White, John J. II: New NASA Transport Research Facilities to Support Research Flight Operation in Present and Future ATC Environments. *1997 World Aviation Congress*, October 13-16, 1997

[6] Smith, R. Marshall: A Description of the Cockpit Motion Facility and the Research Flight Deck Simulator. *AIAA 2000-4174, Modeling and Simulation Technologies Conference and Exhibit*, Aug. 14-17, 2000

[7] Leslie, Richard A.; Geyer, David W.; Cunningham, Kevin; Glaab, Patricia C.; Kenney, P.S.; and Madden, Michael M.: LaSRS++ -- An Object-oriented Framework for Real-time Simulation of Aircraft. *AIAA 1998-4529, Modeling and Simulation Technologies Conference and Exhibit*, Aug. 10-12, 1998

[8] Gamma, Erich; Helm, Richard; Johnson, Ralph; and Glissades, John: *Design Patterns Elements of Reusable Object-Oriented Software*. Reading Massachusetts: Addison-Wesley, 1995, pp. 81-127

[9] *ARINC Mark 33 Digital Information Transfer System (DITS) Part 1*. ARINC Specification 429P1-15, Sep. 1, 1995

[10] Yager, Thomas J.: Aircraft and Ground Vehicle Winter Runway Friction Assessment. NASA TM 1999-209142, 1999

[11] Rine, Laura L.; Abbott, Terence S.; Lohr, Gary W.; Elliott, Dawn M.; Waller, Marvin C.; and Perry, R. Brad: The Flight Deck Perspective of the NASA Langley AILS Concept. NASA TM 2000-209841, 2000