/N
/2720
P-46

NASA Contractor Report 189699

Formal Representation of the Requirements for an Advanced Subsonic Civil Transport (ASCT) Flight Control System

Deborah Frincke
Dave Wolber
Gene Fisher
California Polytechnic State University
San Luis Obispo, California

G. C. Cohen
Boeing Defense & Space Group
Seattle, Washington

NASA Contract NAS1-18586
November 1992

NASA
National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia 23665-5525

[illegible]

Preface

This document was generated in support of NASA contract NAS1-18586, Design and Validation of Digital Flight Control Systems Suitable for Fly-By-Wire Applications, Task Assignment 7. Task 7 is associated with formal representation of requirements.

This document describes a partial requirements specification for an Advanced Subsonic Civil Transport (ASCT) Flight Control System. The example has been adopted from requirements given in NASA Contractor report NAS1-187526, October 1991, G. C. Cohen and R. E. McLees, authors. The language used to describe the requirements, Requirements Specification Language (RSL), is described in a companion document.

The NASA technical monitor for this work is Sally C. Johnson of the NASA Langley Research Center, Hampton, Virginia.

The work was accomplished at Boeing Defense & Space Group, Seattle, Washington, and at the California Polytechnic State University, San Luis Obispo, California. Personnel responsible for the work include:

Boeing Military Airplanes:
D. Gangsaas, Responsible Manager
T. M. Richardson, Program Manager

Boeing High Technology Center:
Gerald C. Cohen, Principal Investigator

California Polytechnic State University
Dr. Gene Fisher
Deborah Frincke, Ph.D. candidate
Dave Wolber, Ph.D. candidate

TABLE OF CONTENTS

1. Introduction

This document is the second in a two-part collection that describes a general-purpose Requirements Specification Language, RSL. The RSL language and supporting toolset are described in detail in Reference 1. Presented here is an extended example of RSL use. The example is a partial requirements specification for an Advanced Subsonic Civil transport (ASCT) Fight Control System. The example has been adapted from requirements given in a NASA Contractor report (Reference 2).

It is important to note that large examples such as this should be viewed using the computer-based RSL browsing tools. These tools are described in a companion document. The sequential text form of the Example presented here is the formal base specification that is input to the RSL browsers. While it is possible to read the specification sequentially, it is intended to be viewed online via the browsers.

When using the RSL browsers, a number of indices are available to facilitate "navigation" through a large specification. The browsers also support hypertext linking and graphical views of a specification. The current implementation of the browsing tools does not provide the means to generate the indices in a textual form suitable for inclusion in a hard-copy document. Hence, the textual form of the example that follows does not include the online browser indices, or any of the hypertext or graphical information that is available online.

2. RSL Definition ASCT

The text of the requirements specification is given in Appendix A.

Appendix A: Formal Requirments Specification

```
(*************************************************************************
 *************************************************************************
 *                                                                       *
 *                 Requirements Specification for an                    *
 *             Advanced Subsonic Civil Transport (ASCT)                 *
 *                       Fight Control System                           *
 *                                                                       *
 *     Adapted from requirments given in NASA Contract report NAS1-18586,  *
 *             August 1989, G. C. Cohen and R E. McLees authors.        *
 *                 (referred to subsequently to as "ASCT1")             *
 *                                                                       *
 *                                                                       *
 * The specification is organized into the following modules:           *
 *                                                                       *
 *     FlyMission, Crew, Aircraft, Navigate, ControlMissionFlight,       *
 *        ControlAerodynamicBraking, ControlLiftConfiguration,          *
 *          ControlPitch, FlightControlSystemPitchFunctions,            *
 *           ControlRoll, FlightControlSystemRollFunctions,             *
 *            ControlYaw, FlightControlSystemYawFunctions,              *
 *                       FlightControlSystem                            *
 *                                                                       *
 *                                                                       *
 *************************************************************************
 *************************************************************************)


(*************************************************************************
 * FlyMission is the top-level module of the system.  It is derived from the
 * material on ASCT1 pages 9-15.
 *************************************************************************)
module FlyMission; (* from pg 13 *)

  import Navigate, ControlMissionFlight;
  export Mission, TargetFlightPath, ActualFlightPath, ExternalForcesOnActuator;

  (*** These three field definitions are from Table 1 on page 15. ***)
  define object attribute control_action, driver, control_system_requirement;

  object Mission is
    components: TaxiInOut and TakeOff and Climb and Cruise
            and Descent and Approach and Landing and AltitudeRange
            and MissionState;
    operations:
      Navigate: (Mission) -> (TargetFlightPlan);
      (* There should be additional operations that are not explicitly
      specified in ASCT1.*)
    description: (*
      Definition of particular flight mission from which the target flight path
      can be generated (ASCT1 pg. 13).  A Mission is the main object of the
      ASCT1 Flight Control System.  Its first seven components represent each
      of the main segments of a controlled flight (ASCT1 pg. 15).  The last two
      components represent the altitudes that may be attained during a mission
      (from 0 to MaxAltitude) and the global states of the mission.*)
```

```
end Mission;

object MissionState is
  components:
    OP: OperatingProcedures,
    FP: FlightPlan,
    FE: FlightEnvelope;
  operations: ;
  description: (* *) ;
end MissionState;

(***** Mission Segments (pg. 15) *****)

object class MissionSegment is
  components: AltitudeRange;
  description: (*
    A generic mission phase.  Only identified component from ASCT1 is
    altitude range, but presumably there should be more. *);
end MissionSegment;

object TaxiInTaxiOut instance of MissionSegment is
  components:  MoveFromTerminalPhase and AltitudeRange;
  operations: ;
  control_action: (* Move from passenger terminal to runway. *);
  {driver:}  (* Terrain and obstacle avoidance. *)
  {control_system_requirement:} (*Speed control, nosewheel steering.*)
end TaxiInTaxiOut;

object TakeOff instance of MissionSegment is
  components: RunwayAcceleration and RunwayDeparture and AltitudeRange;
  operations:
      AccelerateToTakeOff:  (AircraftState, SpeedControls) -> (AircraftState),
      DepartRunway: (AircraftState, LiftOffControls) -> (AircraftState);
  control_action:
    (* Accelerate to takeoff speed and depart runway. *)
  {driver:}
    (* Runway length, thrust limits, crosswind conditions *)
  {control_system_requirement:}
    (* Set height lift, set takeoff trim, thrust setting, nosewheel steering,
    engine out augmentation, on ground braking, stall angle of atack warning,
    manual trajectory control *)
end TakeOff;

object ClimbOutAndClimb instance of MissionSegment is
  components: ClimbOut and ClimbToAltitude;
  control_action:
    (* Ascend to cruise altitude< and speed.*)
  {driver:}
    (* Time constraint, fuel consumption, ease pilot workload, ride quality *)
  {control_system_requirement:}
    (* Thrust setting, manual trajectory control, auto trajectory control,
    manual and auto trim envelope protection, auto control limiting, lift
    config. *)
end ClimbOutAndClimb;

object Cruise instance of MissionSegment is
```

```
  components: ;
  operations: ;
  control_action:
    (* Cruise. *);
  driver:
    (* Ease pilot workload, fuel consumption, minimize drag, ride quality. *);
  control_system_requirement:
    (* Speed control, manual trajectory control, auto trajectory control,
    manual an auto trim, envelope protection, auto control limiting, lift
    control. *);
end Cruise;

object DescentAndApproach instance of MissionSegment is
  components: Descent and Approach;
  control_action:
    (* *);
  driver:
    (* Ease pilot workload, ride quality, crosswind conditions, all weather
    approaches, tight path following. *);
  control_system_requirement: (* *);
    (* Speed control, manual trajectory control, auto trajectory control,
    manual and auto trim, envelope protection, auto control limiting, lift
    control. *);
end DescentAndApproach;

object Landing instance of MissionSegment is
  components: Deceleration and Touchdown ;
  control_action:
    (* Flare, touchdown and decelerate to taxi speed. *);
  driver:
    (* Runway length, crosswind conditions, rapid speed change, tight path
    following all weather landings, ease pilot workload. *);
  control_system_requirement:
    (* Speed control, manual trajectory control, auto trajectory control,
    envelope protection, auto control limiting, lift control, stall angle of
    atack warning. *);
end Landing;

object MissedApproach instance of MissionSegment is
  {control_action: }
  {driver:}
    (* Rapid thrust change; quick, hard maneuvers. *)
  {control_system_requirement:}
    (* Terrain and obstacle avoidance, wind shears, ride quality. *)
  description: (*
    Thrust control, manual trajectory control, envelope protection, lift
    control, engine out augmentation, stall angle of attack *);
end MissedApproach;


object class FlightPath is
  components: Direction, Angle, (*...*) ;
  operations: ;
  description: (* *);
end FlightPath;
```

```
object ActualFlightPath instance of FlightPath is
  components: (*Inherited from FlightPath.*);
  description: (*
    The sensed 4 dimensional flight path and attitudes of the aircraft as
    well as any other sensed values necessary to satisfy the control
    requirements. (See page 13.) *);
end ActualFlightPath;

object TargetFlightPath instance of FlightPath is
  components: ;
  description: (*
    The desired 4 dimensional flight path and attitudes generated by some
    navigation function.  (See page 13.) *);
end TargetFlightPath;

object AircraftAttitudes is
  components: Pitch, Roll, Heading;
  description: (*
    Aircraft pitch, roll and heading attitudes.  (See page 13.) *);
end AircraftAttitudes;

 (*** External forces object.  Referenced in ASCT1, but not thoroughly
     defined there.  See translation notes for further discussion. ***)
 object class ExternalForcesOnActuator is
   components: ;
   operations: ;
   description: (* ;
    All forces (in particular environmental forces) other than the actuation
    forces acting on the aerodynamic braking and roll actuation system.*);
 end ExternalForcesOnActuator;

(* C.M.F.2 *)
operation EvaluateHandlingQualities is
  components: ;
  inputs: a: Aircraft, m: Mission;
  outputs: PilotRating;
  {agent:} (* Pilot *)
  precond: m.State.FlightEnvelope = Normal;
  postcond: ;
  description: (* *);
end EvaluateHandlingQualities;

operation MinimumAugmentation is
  inputs: A: Aircraft;
  outputs: ;
  description: (* *);
end MinimumAugmentation;

var t: Time;
    m: Mission;
    a: Aircraft;
    fl: FailureLevel;
    ef: ExternalForce;

axiom
  if (exists (t: Time, m: Mission, a: Aircrart)
        (m.Time = t and a.State.HandlingQuality = Degraded))
    then m.State = IsDegraded(m);

  axiom
    if (forall (fl: FailureLevel, a: Aircraft)
        Probability(fl) < 1.0*10^-9 and a.State.Mode = CoreControl)
    then MinimumAugmentation(a);

  (* An "external forces" axiom that states that external forces exist that
    cause anomalous conditions to arise, e.g., degraded handling quality. *)
      exists (ef: ExternalForce)
         exists (t: Time)
            if m.Time = t then a.State.HandlingQuality = Degraded;

end FlyMission;


(*******************************************************************************
 * Module Crew contains material gleaned from throughout the ASCT1
 * specification.  Pp. 196-196 contain very brief object descriptions of the
 * Crew, but no details.
 *******************************************************************************)
module Crew;

  object class CrewMember is
    components: Name, SkillLevel, StrengthLevel;
    description: (*
      Class of crew members *);
  end CrewMember;

  object Pilot instance of CrewMember is
    components: PilotClassification;
    description: (*
      The pilot of the mission *);
  end Pilot;

  object CoPilot instance of CrewMember is
    components: PilotClassification;
    description: (*
      The copilot of the mission *);
  end CoPilot;

  object SkillLevel is number;
  object StrengthLevel is number;

  object MissionControlSystem is
    components: (*...*) ;
    description: (*
      The onboard computer support system.  Used as agent of operation where
      appropriate (i.e., in operations that are performed automatically versus
      manually). *);
  end MissionControlSystem;

  operation PerformPilotFunctions is
    components: ;
    inputs: PFPCFF: PilotFlightPathCommandFeelForce;
    outputs: PLTF: PilotLongitudinalTrimForce,
      PFPCF: PilotFlightPathCommandForce;
    description: (*
      The functions performed by the pilot. *);
  end PerformPilotFunctions;

  operation PerformCopilotFunctions is
     components: ;
     inputs: CopilotFlightPathCommandFeelForce (* ... *);
     outputs: CopilotLongitudinalTrimForce, CopilotFlightPathCommandForce
              (* ... *);
      (* The functions performed by the copilot. *);
  end PerformCopilotFunctions;

end Crew;


(******************************************************************************
 * Module Aircraft contains material gleaned from throughout the ASCT1
 * specification.  Pp. 194-197 contain brief object descriptions of the
 * Aircraft, but few details.
 ******************************************************************************)
module Aircraft;

  object Aircraft is
    components: State, Structure, MajorSystems, Attitudes, (* ... *);
    operations: ;
    description: (* *);
  end Aircraft;

  object AircraftState is
    components:
      MCM: ManualControlMode, (* The two modes bof aircraft control, q.v. *)
      HQ: HandlingQuality,
      NWP: NoseWheelPosition,
      LEWP: LeadingEdgeWingPosition,
      TEWP: TrailingEdgeWingPosition;
        (* . . . *)
    operations: (* Many *);
    description: (*
      The top-level repository for all aircraft state information.  Note that
      any explicit definition of this object is conspicuously missing from
      ASCT1. *)
  end AircraftState;

  (* object AircraftState is Aircraft.State; *)
    (* Simple naming macro for ASCT1 consistency -- DISCONTINUED SYNTAX *)

  object NosewheelPosition is  (* pg. 88 *)
    components: ;
    operations: ;
    description: (*
      Angular position of the nosewheel used for on ground low speed heading
      control. *);
  end NosewheelPosition;
```

```
object AircraftStructure is
  components: Engine*, EngineSupport*, PropellerShaft*, HighLiftDevices
              (* ... *);
  description: (*
    The structural components of the aircraft.  Note that only those
    components that appear in the requirements are listed here.  A full
    structural decomposition of the aircraft should be done in a complete
    structures module, and would of course be very detailed. *);
end AircraftStructure;

object class StructuralElement is
  components: HowMounted, WhereMounted;
  operations: ;
  description: (* *);
end StructuralElement;

object Engine instance of StructuralElement is
  components: EngineThrust (* ... *) ;
  description: (*
    The aircraft engine *);
end Engine;

object EngineThrust is  (*** See ASCT1 pg. 88. ***)
  components: ;
  operations: ;
  description: (*
    Thrust measurement; *)
end EngineThrust;

(* obj EnginesThrust is Engine.Thrust; *)
  (* Simple naming macro for ASCT1 consistency -- DISCONTINUED SYNTAX *)

object HighLiftDevices is
  components: LeadingEdgeFlap*, TrailingEdgeFlap*;
  description: (* *);
end HighLiftDevices;


object HowMounted is
  components: Location (* ... *);
  operations: ;
  description: (* *);
end HowMounted;

object HowMountedLocation is
  components: External or Internal;
end HowMountedLocation;

object External = "External";
object Internal = "Internal";


(* Major Aircraft Systems  (ASCT1 pg. 196 and elsewhere).  In a full spec,
 each component here should undoubtedly be represented in a separate module.*)

 object MajorSystems is
```

```
  components: SensorSystem, PilotControlSystem, PropulsionSystem,
    AirframeSystem, AutoFlightSystem;
  operations: ;
  description: (* *);
end MajorSystems;

(*Note that this should certainly be integrated with as a MajorSystems
 component, but it appears as an isolated object in ASCT1.*)
object Autopilot is
  components: ;
  description: (*
    The autopilot control unit. *)
end Autopilot;

(* Flight Modes and Commands *)

object class Mode is
  components: ;
  operations: ;
  description: (*
    A generic flight mode; specializations follow. *);
end Mode;

object class Command is
  components: ;
  operations: ;
  description: (*
    A generic flight command.  Note that the component structure of a command
    is is not precisely clear from the various appearances of the term
    ''command'' throughout ASCT1.  This should be corrected. *)
end Command;


(* Mode Specializations *)
object ManualFlightMode instance of Mode is
  components: Angle;
  operations: ;
  description: (*
    Appears on ASCT1 pg. 129; no textual description given. *)
end ManualFlightMode;

object AutoFlightMode is
  components: ;
  operations: ;
  description: (* ;
    Appears on ASCT1 pg. 129; no textual description given. *)
end AutoFlightMode;


(* Command Specializations *)
object ManualFlightPathCommand instance of Command is
  components: Angle;
  operations: ProvideLongitudinalEnvelopeProtection,
    GenerateFlightPathCommand, GenerateFlightPathCmdManual;
  description: (*
    Flight path angle command generated manually (i.e., by the crew) *);
  end ManualFlightPathCommand;

  object AutoFlightPathCommand instance of Command is
    components: Angle;
    operations: ;
    description: (*
      Flight path command generated in an automated fashion (i.e., by a
      computer system) *);
  end AutoFlightPathCommand;

end Aircraft;


(*******************************************************************************
 * Module Navigate is largely a place holder for information that is outside of
 * the specific focus of this document, but which should be represented
 * formally in some form in a complete document.  Pages 12 and 13 are the only
 * explicit mention of a Navigate function in ASCT1.
 ******************************************************************************)
module Navigate;

  (* Evidently outside of the scope of this spec *)
  operation Navigate is
    components: ;
    inputs: Mission;
    outputs: TargetFlightPath;
    description: (*
      Generates the target flight path based on the particular mission
      requirements and anticipated and sensed environmental conditions. *);
  end Navigate;

end Navigate;


(*******************************************************************************
 * Module ControlMissionFlight is contains the top-level functional components
 * of the system, defined on pp. 17-89 of ASCT1.
 ******************************************************************************)
module ControlMissionFlight;

  from FlyMission import ActualFlightPath, TargetFlightPath,
    ExternalForcesOnActuator, AircraftAttitudes;
  from Mission import MissionState;
  from Aircraft import AircraftState, EnginesThrust;
  from ControlYaw import ExternalForcesOnYawActuator,
    DisplayedDirectionalTrimPos;
  from ControlAerodynamicBraking import DisplayedInflightBrakePos;
  from ControlPitch import DisplayedLongitudinalTrimPosition,
    ExternalForcesOnPitchActuator, StallAngleOfAttackWarning;
  from ControlRoll import DisplayedRollTrimPosition;
  from ControlLiftConfiguration import DisplayedConfigAndFailureStatus;

  define attribute CMF1;  (* General Control Requirements *)
  define attribute CMF2;  (* Handling Qualities *)
  define attribute CMF3;  (* Operational Flight Envelope *)
  define attribute CMF4;  (* Manual and Automatic Trim Functions *)
define attribute CMF5;  (* Envelope protection *)
define attribute CMF6;  (* Autopilot Limiting and Actuation *)
define attribute CMF7;  (* Maneuver Control Lags *)
define attribute CMF8;  (* Requirements in Icing Conditions *)
define attribute CMF9;  (* Control System Stability Requirement *)
define attribute CMF10; (* Residual Oscillations *)
define attribute CMF11; (* Longitudinal Control Power Requirements *)
define attribute CMF12; (* Longitudinal Trim Authority *)
define attribute CMF13; (* Enhanced Longitudinal Control Maneuver Response *)
define attribute CMF14; (* Roll Mode Time Constant *)
define attribute CMF15; (* Pilot-Induced Oscillations *)
define attribute CMF16; (* Stall Characteristics *)
define attribute CMF17; (* Lateral Control Power Requirements *)
define attribute CMF18; (* Roll Response Linearity *)
define attribute CMF19; (* Roll Control Cross Coupling *)
define attribute CMF20; (* Lateral Trim Authority *)
define attribute CMF21; (* Enhanced Roll Maneuver Control *)
define attribute CMF22; (* Dynamic Stability *)
define attribute CMF23; (* Turn Coordination *)
define attribute CMF24; (* Directional Control Power Requirements *)
define attribute CMF25; (* Directional Trim Authority *)
define attribute CMF26; (* Flutter Prevention Requirements *)

operation ControlMissionFlight is  (* pp. 13, 87 *)
  components: ControlThrust, ControlPitch, ControlRoll, ControlYaw,
    ControlHeadingOnGround, ControlAerodynamicBraking,
    ControlBrakingOnGround, ControlLiftConfig, UpdateAircraftState;
  inputs: TargetFlightPath, ExternalForcesOnActuator,
    ExternalForcesOnYawActuator, ExternalForcesOnPitchActuator,
    EnginesThrust;
  outputs:  DisplayedLongitudinalTrimPosition, StallAngleOfAttackWarning,
    DisplayedRollTrimPospition, DisplayedDirectionalTrimPos,
    AircraftAttitudes, ActualFlightPath, DisplayedInflightBrakePos,
    DisplayedConfigAndFailureStatus;
  description: (*
    Receives a target flight path (generated by navigation) and generates
    control signals for the actuation systems which generate the forces and
    moments to control the aircraft attitudes to generate a flight path which
    matches the target flight path. *)
end ControlMissionFlight;

 (* General Remarks:
DSBP: 1a
 Means shall be provided to indicate to the flight crew the position of the
 speed brake system.

DSBP: 1b
 Annunciation of failures or system operation which could result in an
 unsafe condition if the crew were not aware of the condition shall be
 provided (FAR 25..672a)

DSBP: 1c
 Annunciation to the crew (in the form of an aural warning) shall be
 provided for speedbrake deployment or the following condition: take-off
 power and airplane on ground. (FAR 25.703a)
 *
```

```
*)

(** The following are atomic component operations of ControlMissionFlight.
    The remaining component operations are at the head of their respective
    modules. ***)
operation ControlThrust is
  inputs: (* Note that there should probably be inputs here. *);
  outputs: ThrustVectorActuatorConfiguration;
  description: (*
    No description in ASCT1 *);
end ControlThrust;

operation ControlHeadingOnGround is
  components: ;
  inputs: (* Ibid. *);
  outputs: NosewheelPosition;
  description: (*
    No description in ASCT1.  Note also that lack of inputs is suspicious *);
end ControlHeadingOnGround;

operation ControlBrakingOnGround is
  components: ;
  inputs: (* Ibid.*);
  outputs: WhellBrakingPosition;
  description: (*
    No description in ASCT1.  Note also that lack of inputs is suspicious *);
end ControlBrakingOnGround;

(*** Organizationally, this operation would probably better be included in
    the Aircraft module.  It is here to maintain some lexical correspondence
    with ASCT1. ***)
operation UpdateAircraftState is
  components: ;
  inputs: ThrustVectorActuatorConfiguration, PitchActuatorPosition,
      RollActuatorPosition, YawActuatorPosition, NosewheelPosition,
      DragActuatorPosition, WheelBrakePosition, LiftConfig, AircraftState;
   outputs: AircraftAttitudes, ActualFlightPath, AircraftState;
  description: (*
    Includes the airframe and the flight environment and outputs the aircraft
    flight state as a result of the flight state and the configuration of the
    flight control system.  Note: this appears to be a rather imprecise
    description; furthermore, inputs and outputs are not clearly specified.
    See translation notes for further discussion. *);
end UpdateAircraftState;

(*** The following are atomic operations of Control Mission Flight, from pp.
    87-88 of ASCT1.  Global non-atomic operations, such as AircraftState,
    ActualFlightPath, and TargetFlightPath are defined in appropriate major
    object modules.  Local objects that belong to functions defined in other
    modules, such as PitchActuatorPosition, are defined in the appropriate
    operation modules.  Use the browser to find their definitions. ***)

object ThrustVectorActuatorConfiguration is
  components: ;
  operations: ;
  description: (*
      Configuration of the system which controls the magnitude and direction of
      the thrust vector.*);
  end ThrustVectorActuatorConfiguration;


  (*** General Control Requirements (C.M.F.1), Pg. 18: Two modes of manual
      control shall be provided: core control and enhanced control.  ***)

  object ManualControlMode is
    components: CoreControl or EnhancedControl;
    description: (*
      The core control mode provides the minimum level of augmentation (e.g.,
      yaw damper, Mach trim, etc.) required by FAA certification at all failure
      levels not extremely improbable (probability < 1.0E-9). *)
  end ManualControlMode;

  (*** Pg. 18: Transfer between core and enhanced control modes. ***)
  operation TransferControlMode is
    inputs: AS: AircraftState;
    outputs: AS': AircraftState;
    postcond: if AS.ManualControlMode = CoreControl
              then AS'.ManualControlMode = EnhancedControl;
    {agent: Crew or AutoControlUnit}
  end TransferControlMode;

  object CoreControl = "CoreControl";
  object EnhancedControl = "EnhancedControl";

  object HandlingQuality is
    components: Normal or Degraded;
  end HandlingQuality;

  object Normal = "Normal";
  object Degraded = "Degraded";

end ControlMissionFlight;


(*******************************************************************************
 * Module ControlAerodynamicBraking from pp. 90-107
 ******************************************************************************)
module ControlAerodynamicBraking;

  from FlyMission import TargetFlightPath, ActualFlightPath,
    ExternalForcesOnActuator;

  define operation attribute CAB1, CAB1a, CAB1b, CAB1c, CAB1d,
    CAB2, CAB2a, CAB2b;

  operation ControlAerodynamicBraking is (*** pg. 88, 105-107 ***)
    components:
      GenerateManualBrakeCommand, GenerateAutoBrakeCommand,
      DisplaySpeedBrakePos, MoveDragActuator, ProvideCrewBrakingInterface,
      GenerateDragActuatorCommand;
    inputs: TargetFlightPath, ExternalForcessOnActuator, ActualFlightPath;
    (* NOTE: Inconsistency in outputs from ControlAerodynamicBraking between
     pages 87 versus 105. *)
  outputs: DragActuatorPosition, DisplayedInflightBrakePos,
    DragActuatorDisplacement;
  description: (*
    Controls drag and lift dumping to provide and aerodynamic braking
    capability. *);

  CAB1: (* (pg. 90)
   Manual and automatic control of aerodynamic braking shall be available.
   Manual control shall be able to override the automatic control function.
   Aerodynamic speed brake control function shall be available
      for on-ground and in-flight operation. *);

  CAB1a: (* 1.0 Ground Speed Brake Control
   Ground speedbrake control shall provide ground deceleration capability
      consistent with operational field landing length requirements.*)
    (* See predicates on DecelerateOnGround operation. *);

  CAB1b: (* 1.2.0a Inflight Speed Brake Control
   The inflight speed brake actuators shall be sized to give adequate
      inflight deflection at Vmo/Mmo for emergency descent.*);
  (*
    SpeedBrakeAcutators.Size = AdequateInflightDeflection(Vmo, Mno)*)

  CAB1c: (* 1.2.0b Inflight Speed Brake Control
   Normal descent speed brake requirements shall not cause objectionable
      horizontal tail buffet of engine flow distortion (FAR 25.251b) *);
    (* See predicates on Descend and OperateEngine operations. *)

  CAB1d: (*1.2.0c CAB 2.0c Inflight Speed Brake Control
   Control forces to trim the pitching moment change shall be less
      than or equal to those required by FAR 25.143(b)*);
  (* axiom
    Pitching.Moment.ControlForces <= FAR25_143_b; *)
      (* Note that reference to FAR functions assumes there definition
       * elsewhere. *)

  CAB2a: (* Aerodynamic Braking Function Availability Requirements *);

  CAB2a: (*
   Each individual speed brake device shall provide fail-passive control for
   failure modes more probable than 10-7/flt hour. *);

  CAB2b: (*
   Loss of all speedbrake control shall be less than 10-7/flt hour. *);

end ControlAerodynamicBraking;

operation GenerateManualBrakeCommand is
  components: ;
  inputs: TargetFlightPath, ActualFlightPath;
  outputs: CrewBrakeForce;
  {agent: Crew;}
  description: (*
    Generates the speedbrake command manually (i.e., by the crew). *)
end GenerateManualBrakeCommand;
```

```
operation GenerateAutoBrakeCommand is
  components: ;
  inputs: TargetFlightPath, ActualFlightPath;
  outputs: AutoBrakeCommand;
  {agent: FlightControlSystem;}
  description: (*
    Involves generation of the speedbrake command in an automated fashion *);
end GenerateAutoBrakeCommand;

operation DisplaySpeedBrakePos is
  components: ;
  inputs: DragActuatorDisplacement;
  outputs: DisplayedInflightBrakePos;
  description: (* *);
    (* Indicates to the flight crew the position of the speedbrake system and
    annunciates unsafe speedbrake positions and unsafe failures. *)
end DisplaySpeedBrakePos;

operation MoveDragActuator is
  components: ;
  inputs: DesiredDragActuatorPosition, ExternalForcesOnActuator;
  outputs: DragActuatorDisplacement;
  description: (*
    Moves the position of the system which provides the aerodynamic braking
    and lift dumping capability (spoiler/speedbrakes) *);
end MoveDragActuator;

operation ProvideCrewBrakingInterface is
  components: ;
  inputs: CrewBrakeForce;
  outputs: ManualBrakeCommand;
  description: (*
    Converts the force exerted by the crew into an aerodynamic braking
    command. *);
end ProvideCrewBrakingInterface;

operation GenerateDragActuatorCommand is
  components: ;
  inputs: ManualBrakeCommand, AutoBrakeCommand;
  outputs: DesiredDragActuatorPosition;
  description: (*
    Generates a drag actuator command based on the manual and auto braking
    commands. *);
end GenerateDragActuatorCommand;

object DragActuatorPosition is  (* pg. 88 *)
  operations: ControlAerodynamicBraking, UpdateAircraftState;
  description: (*
    Position of the system used to generate drag used for in air and on
    ground aerodynamic braking. *);
end DragActuatorPosition;

object DisplayedInflightBrakePos is  (* pg 106 *)
  operations: ControlMissionFlight, ControlAerodynamicBraking,
    DisplaySpeedBrakePos;
  description: (*
      Indication to the crew of the speedbreak position and status. *);
  end DisplayedInflightBrakePos;

  object DragActuatorDisplacement is  (* pg. 106 *)
    operations: ControlAerodynamicBraking, DisplaySpeedBrakePos,
      MoveDragActuator;
    description: (*
      Displacement of the drag actuators (i.e., the speedbrakes). *);
  end DragActuatorDisplacement;

  object AutoBrakeCommand is  (* pg. 106 *)
    operations: GenerateAutoBrakeCommand, GenerateDragActuatorCommand;
    description: (*
      The automatically (non-manual) generated aerodynamic braking command. *);
  end AutoBrakeCommand;

  object CrewBrakeForce is  (* pg. 106 *)
    operations: GenerateManualBrakeCommand, ProvideCrewBrakingInterface;
    description: (*
      Force exerted by crew (pilot or copilot) on the aerodynamic braking
      controller. *);
    (* NOTE: some reconciliation with the Crew module should be
      made for this and other crew-related objects. *);
  end CrewBrakeForce;

  object DesiredDragActuatorPosition is  (* pg. 106 *)
    operations: MoveDragActuator, GenerateDragActuatorCommand;
    description: (*
      The commanded rage actuator position. *);
  end DesiredDragActuatorPosition;

  object ManualBrakeCommand is  (* pg. 106 *)
    operations: ProvideCrewBrakingInterface, GenerateDragActuatorCommand;
    description: (*
      The speedbrake command generated as a result of the crew exerting a force
      on the controller. *);
  end ManualBrakeCommand;

end ControlAerodynamicBraking;


(******************************************************************************
 * Module ControlLiftConfiguration from pp. 92-119
 ******************************************************************************)
module ControlLiftConfiguration;

  from FlyMission import TargetFlightPath, ActualFlightPath;

  define operation attribute CLC;

  operation ControlLiftConfig is
    components: ;
    inputs: TargetFlightPath, ActualFlightPath;
    outputs: DisplayedConfigAndFailureStatus, LiftConfig;
    description:
      (* Configures the wing for different lift properties such that required
    lift and control is achieved at low speed (takeoff and landing) and low
    drag an be achieved at high speeds. *);

 CLC: (* 1
  The wing high lift design (both leading edge and trailing edge devices)
  shall be adjustable to provide a variable lift capability to ensure the
  achievement of low speeds performance requirements coupled with certifiable
  handling characteristics. Manual and automatic system operation shall be
  provided. High lift device position indication and failure status shall be
  available.*);

 CLC: (*2  p. 93. Lift configuration control function availability
  requirements. The high lift system shall provide the following functional
  availability ( function, probability of loss of function
  (LE and TE Control, 10-7)
  (LE Control, 10-6)
  (TE Control, 10-6)
  (Autoslat, 10-5)
  (Flap load relief, 10-5)
  (LE and TE Failure annunciation, 10-5)
  (LE Control and LE Failure annunciation, 10-9)
  (TE Control and TE Failure annunciation, 10-9)*);

end ControlLiftConfig;

operation GenerateMaualConfigCmd is
  components: ;
  inputs: TargetFlightPath, ActualFlightPath;
  outputs: CrewConfigCmdForce;
    (* Note inconsistent names pp. 88, 112, 113. *)
  {agent: Crew;}
  description: (*
    Involves the generation of the high lift configuration command in the
    manual fashion (i.e., by the crew). Note that name spelling (...Cmd) is
    not consistent with spellings of comparable operations (i.e.,
    ...Command). *);
end GenerateMaualConfigCmd;

operation GenerateAutoConfigCommand is
  components: ;
  inputs: TargetFlightPath, ActualFlightPath;
  outputs: AutoConfigCommand;
  {agent: MissionFlightSystem;}
  description: (*
    Generates the high lift configuration command in an automated fashion
    (i.e., by the computer system). *);
end GenerateAutoConfigCommand;

operation DisplayConfigAndFailStatus is
  components: ;
  inputs: HighLiftConfigAndFailureStatus;
  outputs: DisplayedConfigAndFailureStatus;
  description: (*
    Displays to the crew the position of the high lift devices and
    annunciates any height lift device failure conditions. *);
end DisplayConfigAndFailStatus;
```

```
operation MoveLiftConfigActuator is
  components: ;
  inputs: HighLiftActuatorCommands;
  outputs: HighLiftDevicePositions, HighLiftConfigAndFailureStatus,
    LiftConfig;
  description: (*
    Involves the actuation of the high lift devices (i.e., the leading edge
    and trailing edge flaps). *);
end MoveLiftConfigActuator;

operation ProvideCrewConfigInterface is
  components: ;
  inputs: CrewHLConfigCmdForce;
  outputs: ManualConfigCmd;
  description: (*
    Provides the interface which allows the crew to input commands to the
    high lift system.  See notes in analysis section about ad hoc user
    interface specification in the original ASCT1. *);
end ProvideCrewConfigInterface;

operation GenerateConfigActuatorCmd is
  components: ;
  inputs: AutoConfigCommand, HighLiftDevicePositions, ManualConfigCmd;
  outputs: HighLiftActuatorCommands;
  description: (*
    Involves the actuation of the high lift devices (i.e., the leading edge
    and trailing edge flaps). *);
end GenerateConfigActuatorCmd;

object AutoConfigCommand is  (* pg. 113 *)
  operations: GenerateAutoConfigCommand, GenerateConfigActuatorCmd;
  description: (*
    The automatically generated commands for the leading edge and trailing
    edge high lift devices. *);
end AutoConfigCommand;

object CrewConfigCmdForce is  (* pg. 113 *)
  operations: GenerateMaualConfigCmd, ProvideCrewConfigInterface;
  description: (*
    This is the force exerted by the crew to generate the manual high lift
    configureation command. *);
end CrewConfigCmdForce;

object DisplayedConfigAndFailureStatus is  (* pg. 113 *)
  operations: ControlMissionFlight, ControlLiftConfig,
    DisplayConfigAndFailStatus;
  description: (*
    *);
end DisplayedConfigAndFailureStatus;

object HighLiftActuatorCommands is  (* pg. 113 *)
  operations: MoveLiftConfigActuator, GenerateConfigActuatorCmd;
  description: (*
    Commands to the various actuators which move the leadin edge and trailing
    edge flaps. *);
end HighLiftActuatorCommands;
```

```
  object HighLiftDevicePositions is  (* pg. 113 *)
    operations: MoveLiftConfigActuator, GenerateConfigActuatorCmd;
    description: (*
      Sensed positins of the leading edge and trailin ede high lift positions.
      *);
  end HighLiftDevicePositions;

  object HighLiftConfigAndFailureStatus is  (* pg. 113 *)
    operations: DisplayConfigAndFailStatus, MoveLiftConfigActuator;
    description: (*
      Position of leading edge and trailing edge high lift devices and failure
      status of the high lift devices. *);
  end HighLiftConfigAndFailureStatus;

  object LiftConfig is
    components: (* Surmised from prose description on pg. 113. *)
      LeadingEdgeWingPos, TrailingEdgeWingPos;
    operations: UpdateAircraftState, ControlLiftConfig;
    description: (*
      Configuration of the lift system to achieve necessary lift to support
      desired flight path angle at all mission phases (speeds and altitudes).
      The record consists of the leading edge and trailing edge wing positions.
      *);
  end LiftConfig;

end ControlLiftConfiguration;


(********************************************************************************
 * Module ControlPitch from pp. 120-144
 ********************************************************************************)
module ControlPitch;
  from FlyMission import TargetFlightPath, ActualFlightPath,
    ExternalForcesOnActuator;
  from AirCraft import AutolFlightPathCommand, ManualFlightPathCommand;

  define operation attribute LAPC;
  define operation attribute PLEP;
  define operation attribute PSAW;

  operation ControlPitch is (* pp. 87, 120*)
    components: GenerateLongitudinalTrimCommand, DisplayLongitudinalTrimStatus,
      GeneratePitchActuatorCommand, MovePitchActuators,
      ProvideStallAngleOfAttackWarning, ProvideLongitudinalEnvelopeProtection,
      GenerateFlightPathCommand, LimitAutoPitchCommand;
    inputs: ActualFlightPath, ExternalForcesOnPitchActuator, TargetFlightPath;
    outputs: StallAngleOfAttackWarning, DisplayedLongitudinalTrimPosition,
        PitchActuatorPosition;
    description:  (*
      Performs all functions required to control the lateral axis by
      controlling the pitch angle. *);
     LAPC: (* *);
     PLEP: (* *);
     PLEP: (* *);
     PSAW: (* *);
  end ControlPitch;
```

```
operation GenerateLongitudinalTrimCommand is
  components: ;
  inputs: ;
  outputs: AutoLongitudinalTrimCommand, ManualLongitudinalTrimCommand;
  description:
    (* Generates trim commands to offload steady state pitch commands from
       the elevator to the stabilizer. *)
end GenerateLongitudinalTrimCommand;

operation DisplayLongitudinalTrimStatus is
  components: ;
  inputs: LongitudinalTrimPosition;
  outputs: DisplayedLongitudinalTrimPosition;
  description: (*
    Displays the longitudinal trim status to the crew.  NOTE: inconsistency
    in this function name on pp. 120 and 121.) *)
end DisplayLongitudinalTrimStatus;

operation GeneratePitchActuatorCommand is
  components: ;
  inputs: LimitedFlightPathCommand, ManualLongitudinalTrimCommand,
    AutoLongitudinalTrimCommand, ActualFlightPath;
  outputs: DesiredPitchActuatorPosition, LongitudinalTrimPosition;
  description: (*
    Generates the pitch actuator (elevator and stabilizer) position command
    based on the flight path angle and longitudinal trim commands *)
end GeneratePitchActuatorCommand;

operation MovePitchActuators is
  components: ;
  inputs: DesiredPitchActuators, ExternalForcesOnActuator;
  outputs: PitchActuatorPosition;
  description: (*
    Receives the desired pitch actuators positions and attempts to
    move the actuators to thoses positions. *)
end MovePitchActuators;

operation ProvideStallAngleOfAttackWarning is
  components: ;
  inputs: ActualFlightPath;
  outputs: StallAngleOfAttackWarning;
  description: (*
    Monitors the aircraft flight path state vector and attitudes and
    generates a warning for the crew when approaching the aircraft stall
    angle of attack.  NOTE naming inconsistency on pp. 120 and 121. *)
end ProvideStallAngleOfAttackWarning;

operation ProvideLongitudinalEnvelopeProtection is
  components: ;
  inputs: ActualFlightPath, ManualFlightPathCommand, LimitedFlightPathCommand;
  outputs: LimitedFlightPathCommand;
  description: (*
    Monitors the aircraft states and modifies the flight path angle command
    as necessary to satisfy the longitudinal envelope protection
    requirements. *);
end ProvideLongitudinalEnvelopeProtection;
```

```
 operation GenerateFlightPathCommand is
   components: GenerateFlightPathCommandManual,
     MakeManualVsAutoFlightModeDecision, EngageManOrAutoOperation,
     GenerateFlightPathCommandAuto;
   inputs: TargetFlightPath, ActualFlightPath;
   outputs: ManualFlightPathCommand, AutoFlightPathCommand;
   description: (*
     Compares the actual flight path angle to the desired flight path angle
     and generates the necessary flight path angle command. *);
 end GenerateFlightPathCommand;

 operation LimitAutoPitchCommand is
   components: ;
   inputs: AutoFlightPathCommand;
   outputs: LimitedFlightPathCommand;
   description: (*
     Limits the autopilot control authority and protects against failures (in
     particular hardover and oscillatory failures) in the autopilot. *);
 end LimitAutoPitchCommand;

 (* Pp. 127 - 128 *)
 operation GenerateFlightPathCmdManual is
   inputs: ActualFlightPath, TargetFlightPath;
   outputs: ManualFlightPathCommand;
   description: (*
     Involves the generation of a flight path command manually (i.e., by the
     crew) as a result of comparing the target and actual flight paths. *);
 end GenerateFlightPathCmdManual;

 operation GenerateFlightPathCmdAuto is
   inputs: ;
   outputs: ;
   description: (*
     Generates a flight path angle command automatically (i.e., by the a
     computer) as a result of the difference between the actual and target
     flight paths. *);
 end GenerateFlightPathCmdAuto;

 operation MakeManualVsAutoFlightModeDecision is
   inputs: ManualFlightMode;
   outputs: AutoFlightMode;
   description:
     (* Decides whether to generate flight path commands manually or
     automatically. *);
 end MakeManualVsAutoFlightModeDecision;

{Evidently already defined -- FIX
 operation EngageManOrAutoOperation is
   inputs: ManualFlightMode;
   outputs: AutoFlightMode;
   description: (*
     Activates one of the flight path command generation processes depending
     on the mode engaged. *);
 end EngageManOrAutoOperation;
}
```

```
object ExternalForcesOnPitchActuator instance of ExternalForcesOnActuator is
  operations: ControlMissionFlight, ControlPitch;
  description: (*
    All forces (in particular environmental forces) other than the actuation
    forces acting on the pitch actuator. *);
end ExternalForcesOnPitchActuator;

object DisplayedLongitudinalTrimPosition is
  operations: ControlMissionFlight, ControlPitch,
    DisplayLongitudinalTrimStatus;
  description:
    (* The longitudinal trim position displayed to the crew *)
end DisplayedLongitudinalTrimPosition;

object AutoLongitudinalTrimCommand is
  operations: GenerateLongitudinalTrimCommand, GeneratePitchActuatorCommand;
  description: (*
    The Longitudinal trim command generated automatically during enhanced
    manual control and autoflight control *)
end AutoLongitudinalTrimCommand;

object ManualLongitudinalTrimCommand is
  operations: GenerateLongitudinalTrimCommand, GeneratePitchActuatorCommand;
  description: (*
    The longitudinal trim command generated by the crew for use during normal
    and backup control *);
end ManualLongitudinalTrimCommand;

object AutoFlightPathCommand is
  operations: GenerateFlightPathCommand, LimitAutoPitchCommand;
  description: (*
    The flight path command generated automatically during enhanced manual
    control and autoflight control *);
end AutoFlightPathCommand;

object LongitudinalTrimPosition is
  operations: DisplayLongitudinalTrimStatus, GeneratePitchActuatorCommand,
    GeneratePitchActuatorCommand;
  description:
    (* Position of the longitudinal trim actuator *)
end LongitudinalTrimPosition;

object ActualFlightPath is
  operations: ;
  description: (*
    The sensed 4 dimensional flight path & attitudes of the aircraft as well
    as other sensed values necessary to satisfy the control requirements.*);
end ActualFlightPath;

object LimitedFlightPathCommand is
  operations: GeneratePitchActuatorCommand,
    ProvideLongitudinalEnvelopeProtection, LimitAutoPitchCommand;
  description: (*
    The flight path command limited such that envelope protection is not
    violated. *);
end LimitedFlightPathCommand;
```

```
  object StallAngleOfAttackWarning is
    operations: ControlMissionFlight, ControlPitch,
      ProvideStallAngleOfAttackWarning;
    description: (*
      The audible and visual indication to the crew that the aircraft is
      approaching the stall angle of attack. *);
  end StallAngleOfAttackWarning;

  object PitchActuatorPosition is
    operations: ;
    description: (*
      The Position of the actuator(s) which provide(s) aircraft pitch maneuver
      and trim control. *);
  end PitchActuatorPosition;

  (* object TargetFlightPath is FlyMission.TargetFlightPath; *)

  object DesiredPitchActuatorPosition is
    operations: GeneratePitchActuatorCommand, MovePitchActuators;
    description: (*
      The desired pitch actuator (elevator) position such that the limited
      flight path angle command is achieved *);
  end DesiredPitchActuatorPosition;

  (* object ExternalForcesOnActuator = FlyMission.ExternalForcesOnActuator; *)

end ControlPitch;


(*****************************************************************************
 * Module FlightControlSystemPitchFunctions from pp. 129-144
 *****************************************************************************)
module FlightControlSystemPitchFunctions;

  from ControlPitch import PitchActuatorPosition;
  from Aircraft import AutoFlightPathCommand;

  operation FlightControlSystemPitchContext is
    components: PerformPilotFunctions, PerformCopilotFunctions,
      FlightControlSystemPitchFunctions, PerformAutoFlightSystemFunctions;
    inputs:
      (* Unclear -- see pg. 129. *);
    outputs: PitchActuatorPosition;
    description: (* Unclear -- see pp. 129-130. *);
  end FlightControlSystemPitchContext;

  operation FlightControlSystemPitchFunctions is
    components: ProvidePilotPitchInterface, ProvideCopilotPitchInterface,
      DisplayLongitudinalTrimStatus, ResolvePitchControlContention,
      GeneratePitchActuatorCommand, MovePitchActuators,
      ProvideStallAngleOfAttackWarning,
      DisplayLongitudinalEnvelopeProtectStatus,
      ProvideLongitudinalEnvelopeProtection, LimitAutoPitchCommands;
    inputs: PilotLongitudinalTrimForce, PilotFlightPathCommandForce,
      CopilotLongitudinalTrimForce, CopilotFlightPathCmdForce,
      AutoLongitudinalTrimCommand, AutoFlightPathCommand
    (* PLUS MAYBE THE FOLLOWING DUE TO AMBIGUITY ON PP. 129 VS 133: *)
    , ActualFlightPath, ExternalForcesOnActuator;
  outputs: PilotFlightPathCmdFeelForce, CopilotFlightPathCmdFeelForce,
    PitchActuatorPosition;
  description: (*
    Note that the following description is taken from ASCT1 page 130, but it
    is not a fully accurate description of this operation as it is defined in
    WSRSL.  See the remarks in the first-year report.

    Contains all the flight control functions assigned to the FCS.  As a
    result of this assignment several new processes are created.  some of
    these are interface functions and others are as a result of how functions
    were allocated to the AEs.  (I.e., Envelope Protection was assigned to
    the FCS with a probability of failure < 10E-6.  However this function
    requires <10E-9.  Therefore the pilot and copilot must perform envelope
    protection when not being performed by the FCS.  Thus a pilot indication
    function of the status of envelope protection is generated.)  Pilot and
    copilot can command roll reate, thus there is a function requirement to
    resolve control contention.*)
end FlightControlSystemPitchFunctions;

object CopilotFlightPathCommandFeelForce is
  components: ;
  description: (*
    A resistance force exerted by the controller which is a feedback to the
    copilot indicative of the flight path angle. *);
end CopilotFlightPathCommandFeelForce;

object CopilotFlightPathCommandForce is
  components: ;
  description: (*
    The physical force generated by the copilot to control the aircraft
    flight path angle. It is in the form of a force exerted by the pilot's
    hand. *);
end CopilotFlightPathCommandForce;

object CopilotLongitudinalTrimForce is
  components: ;
  description: (*
    The physical force exerted by the copilot's hand to generate the desired
    longitudinal trim command. *);
end CopilotLongitudinalTrimForce;

object PilotFlightPathCommandForce is
  components: ;
  description: (*
    The physical signal created by the pilot to control the aircraft flight
    path. It is in the form exerted by the pilot.*);
end PilotFlightPathCommandForce;

object PilotFlightPathFeelForce is
  components: ;
  description: (*
    A resistance force exerted by the controller which is a feedback to the
    pilot indicative of the flight path command. *);
end PilotFlightPathFeelForce ;
```

```
 object PilotLongitudinalTrimForce is
   description: (*
     This flow is the physical force exerted by the pilot's hand to generate
     the desired longitudinal trim command. *);
 end PilotLongitudinalTrimForce;

 (*Pp. 133 - 135*)
 operation ProvidePilotPitchInterface is
   components: ;
   inputs: PilotLongitudinalTrimForce, PilotFlightPathCmdForce;
   outputs: PilotFlightPathCmdFeelForce, PilotLongitudinalTrimCommand,
     PilotFlightPathCommand;
   description: (*
     Converts the signal received from the pilot in the form of a force
     exerted by the pilot into a flight path angle command signal to be used
     by the FCS.  It also provides the pilot with a feedback feel force
     indicative of the command. *);
 end ProvidePilotPitchInterface;

 operation ProvideCopilotPitchInterface is
   components: ConvertForceToDisplacement, GenerateLongitudinalFeelForce,
     TranslateFlightPathDisplacementToCommand,
     TranslateTrimForceToTrimCommand;
   inputs: CopilotFlightPathCmdForce, CopilotLongitudinalTrimForce;
   outputs: CopilotFlightPathCmdFeelForce, CopilotFlightPathCommand,
     CopilotLongitudinalTrimCommand;
   description: (*
     Provides the same capability for the copilot as the
     ProvidePilotPitchInterface does for the pilot. *);
 end ProvideCopilotPitchInterface;

{Evidently already defined -- FIX
 operation DisplayLongitudinalTrimStatus is
   components: ;
   inputs: ;
   outputs: LongitudinalTrimPosition;
   description: (*
     Displays the longitudinal trim status to the crew. *);
 end DisplayLongitudinalTrimStatus;
}

 operation ResolvePitchControlContention is
   components: ;
   inputs: CopilotFlightPathCommand, CopilotLongitudinalTrimCommand,
     PilotFlightPathCommand, PilotLongitudinalTrimCommand;
   outputs: ManualFlightPathCommand;
   description: (*
     Generated by the assignment of the GenerateFlightPathCommandManual to
     both the pilot and copilot.  NOTE: this description is unclear. *);
 end ResolvePitchControlContention;

{Evidently already defined -- FIX
 operation GeneratePitchActuatorCommand is
   components: ;
   inputs: ManualFlightPathCommand, ActualFlightPath,
     AutoLongitudinalTrimCommand;
```

```
  outputs: LongitudinalTrimPosition, DesiredPitchActuatorPosition;
  description: (*
    Generates the pitch actuator (elevator and stabilizer) position commands
    based on the flight path angle and longitudinal trim commands.
     *);
 end GeneratePitchActuatorCommand;
}

{Evidently already defined -- FIX
 operation MovePitchActuators is
   components: ;
   inputs: DesiredPitchActuatorPosition, ExternalForcesOnActuator;
   outputs: PitchActuatorPosition;
   description: (*
     Receives the desired pitch actuators positions and attempts to move the
     actuators to those positions. *);
 end MovePitchActuators;
}

{Evidently already defined -- FIX
 operation ProvideStallAngleOfAttackWarning is
   components: ;
   inputs: ActualFlightPath;
   outputs: StallAngleOfAttackWarning;
   description: (*
     Monitors the aircraft flight path state vector and attitudes and
     generates a warning for the crew when approaching the aircraft stall
     angle of atack. *);
 end ProvideStallAngleOfAttackWarning;
}

 (*NOTE: Inconsistent Names Pp. 133, 134*)
 operation DisplayLongitudinalEnvelopeProtectStatus is
   components: ;
   inputs: LongitudinalEnvelopeProtectStatus;
   outputs: DisplayedLongitudinalEnvelopeProtectStatus;
   description: (*
     Results from the allocation of ProvideLongitudinalEnvelopeProtection to
     the FCS with a probability of loss of function of <10E-6.  Pitch envelope
     protection has a req for probability of loss of function <10E-9, and thus
     the crew has responsibility for pitch envelope protection when not
     performed by the FCS.  Thus the crew must be aware of envelope protect
     status, hence the functional requirement to
     DisplayLongitudinalEnvelopeProtectStatus *);
 end DisplayLongitudinalEnvelopeProtectStatus;

{Evidently already defined -- FIX
 operation ProvideLongitudinalEnvelopeProtection is
   components: ;
   inputs: ActualFlightPath, LimitedFlightPathCommand,
     ManualFlightPathCommand;
   outputs: LimitedFlightPathCommand, LongitudinalEnvelopeProtectStatus;
   description: (*
     Monitors the aircraft states and modifies the flight path angle command
     as necessary to satisfy the longitudinal envelope protection
     requirements.
```

```
      *);
  end ProvideLongitudinalEnvelopeProtection;
 }

  operation LimitAutoPitchCommands is
    components: ;
    inputs: AutoFlightPathCommand;
    outputs: LimitedFlightPathCommand;
    description: (*
      Limits the autopilot control authority and protects against failures (in
      particular hardover and oscillatory failures in the autopilot. *);
  end LimitAutoPitchCommands;

  (**** Pilot Pitch Interface, pp. 137-138 ****)
  operation ConvertForcesToDisplacement is
    components: ;
    inputs: FlightPathCommandForce, FlightPathCommandFeelForce;
    outputs: FlightPathCommandDisplacement;
    description: (*
      Receives the pilot force and feedback feel force and generates a
      displacement.  Note name inconsistency on pp. 137, 138. *)
  end ConvertForcesToDisplacement;

  operation GenerateLongitudinalFeelForce is
    components: ;
    inputs: FlightPathAngleCommand;
    outputs: FlightPathCommandFeelForce;
    description: (*
      Generates a force to feedback to the pilot which is indicative of the
      pitch maneuver and trim commands.  Note name inconsistency pp. 137,138.
      *);
  end GenerateLongitudinalFeelForce;

  operation TranslateFlightPathDisplacementToCommand is
    components: ;
    inputs: FlightPathCommandDisplacement;
    outputs: FlightPathAngleCommand;
    description: (*
      Translates the physical displacement of the pitch controller into a
      flight path command.  Note name inconsistency pp. 137,138. *);
  end TranslateFlightPathDisplacementToCommand;

  operation TranslateTrimForceToTrimCommand is
    components: ;
    inputs: LongitudinalTrimForce;
    outputs: LongitudinalTrimCommand;
    description: (*
      Converts the physical displacement generated by the physical force
      exerted by the pilot into a trim command for use by the FCS. Note name
      inconsistency pp. 137,138. *);
  end TranslateTrimForceToTrimCommand;

end FlightControlSystemPitchFunctions;


(******************************************************************************
 * Module ControlRoll from pp. 145-168
 ***************************************************************************)
module ControlRoll;

  from FlyMission import TargetFlightPath, ActualFlightPath,
    ExternalForcesOnActuator;

  operation ControlRoll is (*pp. 87, 145*)
    components: GenerateRollTrimCommand, DisplayRollTrimPosition,
      GenerateRollActuatorCommand, MoveRollActuator,
      ProvideRollEnvelopeProtection, GenerateRollRateCommand,
      LimitAutoRollCommands;
    inputs: TargetFlightPath, ActualFlightPath, ExternalForcesOnRollActuator;
    outputs: DisplayedRollTrimPosition, RollActuatorPosition;
    description: (*
      Performs all functions required to control the lateral axis by
      controlling the roll angle. *);
  end ControlRoll;

  operation GenerateRollTrimCommand is
    components: ;
    inputs: (*Note that no inputs is suspicious here*);
    outputs: AutoRollTrimCommand, ManualRollTrimCommand;
    description: (*
      Generates roll trim commands to offset asymmetries such as engine out,
      engine loss and lateral winds. *);
  end GenerateRollTrimCommand;

  operation DisplayRollTrimPosition is
    components: ;
    inputs: RollTrimPosition;
    outputs: DisplayedRollTrimPosition;
    description: (*
      Displays roll trim position to the crew. *);
  end DisplayRollTrimPosition;

  operation GenerateRollActuatorCommand is
    components: ;
    inputs: ManualRollTrimCommand, AutoRollTrimCommand, ActualFlightPath,
      LimitedRollRateCommand;
    outputs: RollTrimPosition, DesiredRollActuatorPosition;
    description: (*
      Generates the roll actuator (aileron / spoiler) position commands based
      on roll rate and trim commands. *);
  end GenerateRollActuatorCommand;

  operation MoveRollActuator is
    components: ;
    inputs: DesiredRollActuatorPosition, ExternalForcesOnRollActuator;
    outputs: RollActuatorPosition;
    description: (*
      Receives the desired roll actuator position and attempts to move the roll
      actuator to that position. *);
  end MoveRollActuator;

  operation ProvideRollEnvelopeProtection is
  components: ;
  inputs: ManualRollRateCommand, LimitedAutoRollCommand, RollAngle;
  outputs: LimitedRollRateCommand;
  description: (*
    Monitors actual roll angle and commanded roll rate and modifies the roll
    rate command as necessary to prevent the roll angle from exceeding
    certain limits. *);
end ProvideRollEnvelopeProtection;

operation GenerateRollRateCommand is
  components: GenerateRollRateCommandManual, EngageManOrAutoOperation,
    GenerateRollRateCommandAuto, MakeManualVsAutoFlightModeDecision;
  inputs: TargetFlightPath, ActualFlightPath;
  outputs: AutoRollRateCommand, ManualRollRateCommand;
  description: (*
    Compares the target flight path and actual flight path and generates
    necessary roll rate command to drive the actual to the target. *);
end GenerateRollRateCommand;

operation LimitAutoRollCommands is
  components: ;
  inputs: AutoRollRateCommand;
  outputs: LimitedAutoRollCommand;
  description: (*
    Limits the autopilot control authority and protects against failures (in
    particular hardover or oscillatory failures) in the autopilot. *);
end LimitAutoRollCommands;

(* Pp. 151-152 *)
operation GenerateRollRateCommandManual is
  components: ;
  inputs: ActualFlightPath, TargetFlightPath, ManualModeEngaged;
  outputs: ManualRollRateCommand;
  description: (*
    Involves the generation of ta roll rate command manually (i.e., by the
    crew) as a result of comparing the target and actual flight paths. *);
end GenerateRollRateCommandManual;

operation EngageManOrAutoOperation is
  components: ;
  inputs: ManualFlightMode, AutoFlightMode;
  outputs: ManualModeEngaged, AutoModeEngaged;
  description: (*
    Activates one of the roll rate generation processes depending on the mode
    engaged. *)
end EngageManOrAutoOperation;

operation GenerateRollRateCommandAuto is
  components: ;
  inputs: AutoModeEngaged, TargetFlightPath, ActualFlightPath;
  outputs: AutoRollRateCommand;
  description: (*
    Involves the generation of a roll rate command automatically (i.e., by
    the computer) as a result of the difference between the actual and target
    flight path. *);
end GenerateRollRateCommandAuto;
```

```
operation MakeManualVsAutoFlightModeDecision is
  components: ;
  inputs: (* Note no inputs -- seem reasonable here. *);
  outputs: ManualFlightMode, AutoFlightMode;
  description: (*
    Decides whether to generate flight path commands manually or
    automatically.  Note -- not clear if this should be the same as operation
    of the same in ControlPitch module. *);
end MakeManualVsAutoFlightModeDecision;

object AutoRollRateCommand is
  operations: GenerateRollRateCommand, LimitAutoRollCommands,
    GenerateRollRateCommandAuto;
  description: (*
    Roll rate command generated in an automated fashion (i.e., by an
    autoflight computer). *);
end AutoRollRateCommand;

object AutoRollTrimCommand is
  operations: GenerateRollTrimCommand, GenerateRollActuatorCommand;
  description: (*
    Roll trim command generated automatically for use during enhanced manual
    control and autoflight control. *);
end AutoRollTrimCommand;

object DesiredRollActuatorPosition is
  operations: GenerateRollActuatorCommand, MoveRollActuator;
  description: (*
    The desired roll actuator position such that the limited roll rate
    command is achieved. *);
end DesiredRollActuatorPosition;

object DisplayedRollTrimPosition is
  operations: ControlMissionFlight, ControlRoll, DisplayRollTrimPosition,
    FlightControlSystemRollFunctions, DisplayRollTrimStatus;
  description: (*
    The roll trim position displayed to the crew. *);
end DisplayedRollTrimPosition;

object ExternalForcesOnRollActuator instance of ExternalForcesOnActuator is
  operations: ControlRoll, MoveRollActuator,
    FlightControlSystemRollFunctions;
  description: (*
    All forces (in particular environmental forces) other than the actuation
    forces acting on the aerodynamic braking and roll actuation system. *);
end ExternalForcesOnRollActuator;

object LimitedAutoRollCommand is
  operations: ProvideRollEnvelopeProtection, LimitAutoRollCommands;
  description: (*
    The auto roll rate command limited to the autoflight roll authority. *);
end LimitedAutoRollCommand;

object LimitedRollRateCommand is
  operations: GenerateRollActuatorCommand, ProvideRollEnvelopeProtection;
  description: (*
      The roll rate command limited such that the envrlope protection criteria
      are not violated. *);
  end LimitedRollRateCommand;

  object ManualRollRateCommand is
    operations: ProvideRollEnvelopeProtection, GenerateRollRateCommand,
      GenerateRollRateCommandManual, ResolveRollControlContention;
    description: (*
      Roll rate command generated manually (i.e., by the crew). *);
  end ManualRollRateCommand;

  object ManualRollTrimCommand is
    operations: GenerateRollTrimCommand, GenerateRollActuatorCommand;
    description: (*
      The roll trim command as generated by the crew for normal control.  The
      trim provides a steady state roll angle to offset asymmetries. *);
  end ManualRollTrimCommand;

  object RollActuatorPosition is
    operations: UpdateAircraftState, ControlRoll, MoveRollActuator,
      FlightControlSystemRollContext, FlightControlSystemRollFunctions;
    description: (*
      Position of the system which makdes the aircraft roll. *);
  end RollActuatorPosition;

  object RollAngle is
    operations: ProvideRollEnvelopeProtection,
      FlightControlSystemRollFunctions;
    description: (*
      Airplane roll angle. *);
  end RollAngle;

  object RollTrimPosition is
    operations: DisplayRollTrimPosition, GenerateRollActuatorCommand;
    description: (*
      Position of the roll trim actuator. *);
  end RollTrimPosition;

end ControlRoll;


(*****************************************************************************
 * Module FlightControlSystemRollFunctions from pp. 153-168
 *****************************************************************************)
module FlightControlSystemRollFunctions;

  operation FlightControlSystemRollContext is
    components: PerformPilotFunctions, PerformCopilotFunctions,
      FlightControlSystemRollFunctions, PerformAutoFlightSystemFunctions;
    inputs: (* Unclear -- see pg. 153. *);
    outputs: RollActuatorPosition;
    description: (* Unclear -- see pp. 153-154.  Also cf.
      FlightControlSystemPitchContext in module
      FlightControlSystemRollFunctions above. *);
  end FlightControlSystemRollContext;
```

```
operation FlightControlSystemRollFunctions is
  components: ProvidePilotRollInterface, ProvideCopilotRollInterface,
    DisplayRollTrimStatus, ResolveRollControlContention,
    GenerateRollActuatorcommand, MoveRollActuator,
    DisplayRollEnvelopeProtectStatus, ProvideRollEnvelopeProtection,
    LimitAutoRollCommands;
  inputs: PilotRollTrimForce, PilotRollRateForce, CopilotRollRateForce,
    CopilotRollTrimForce, AutoRollTrimCmd, AutoRollRateCmd
    (* plus maybe the following due to ambiguity on pp. 153 versus 157: *)
    , ActualFlightPath, ExternalForcesOnActuator, RollAngle (*Note that the
    RollAngle input here is seemingly inconsistent with the ActualFlightPath
    input in the comparable position in the FlightContorlSystemPitchFunctions
    on pg. 133.*);
  outputs: PilotRRCmdFeelForce, CopilotRRCmdFeelForce,
    DisplayedRollTrimPosition, RollActuatorPosition,
    DisplayedRollEnvelopeProtectStatus (*Note that as with inputs, these are
    inconsistent on pp. 153 versus 157.*);
  description: (*
    Note that the following description is taken from ASCT1 page 154, but it
    is not a fully accurate description of this operation as it is defined in
    WSRSL.  See the remarks in the first-year of the report.  Cf. also
    description of operation FlightControlSystemPitchFunctions above.

    Contains all the flight control functions assigned to the FCS.  As a
    result of this assignment several new processes are created.  Some of
    these are interface functions and others are as a result of how functions
    were allocated to the AEs.  (I.e., Envelope protection was assigned to
    the FCS with a probability of failure <10E-6.  However this function
    requires <10E-09.  Therefore the pilot and copilot must perform envelope
    protection when not being performed by the FCS.  Thus a pilot indication
    function of the status of envelope protection is generated.)  Pilot and
    copilot can command roll rate, thus there is a functional requirement to
    resolve control contention. *);
end FlightControlSystemRollFunctions;

operation ProvidePilotRollInterface is
  components: ConvertForcesToDisplacement, GenerateRollFeelForce,
    TranslateRRDisplToRRCommand, TranslateTrimForceToTrimCommand;
  inputs: PilotRollTrimForce, PilotRollRateForce;
  outputs: PilotRRCmdFeelForce, PilotRollTrimCommand, PilotRollRateCommand;
  description: (*
    Converts the signal received from the pilot in the form of a force
    exerted by the pilots hand into a roll rate signal to be used by the FCS.
    It also provides the pilot with a feedback feel force proportional to the
    commanded roll rate. *);
end ProvidePilotRollInterface;

operation ProvideCopilotRollInterface is
  components: ;
  inputs: CopilotRollRateForce, CopilotRollTrimForce;
  outputs: CopilotRRCmdFeelForce, CopilotRollRateCommand,
    CopilotRollTrimCommand;
  description: (*
    Provides the same function for the copilot as the
    ProvidePilotRollInterface does for the pilot. *);
end ProvideCopilotRollInterface;
```

```
operation DisplayRollTrimStatus is
  components: ;
  inputs: RollTrimPosition;
  outputs: DisplayedRollTrimPosition;
  description: (*
    Displays roll trim position to the crew.  Note naming inconsistency on
    pp. 157 vs. 158. *);
end DisplayRollTrimStatus;

operation ResolveRollControlContention is
  components: ;
  inputs: PilotRollRateCommand, PilotRollTrimCommand, CopilotRollRateCommand,
    CopilotRollTrimCommand;
  outputs: ManualRollRateCommand, ManualRollTrimCmd;
  description: (*
    Generated by the assignment of the Generate Roll Rate Cmd Manual to both
    the pilot and copilot. *)
end ResolveRollControlContention;

operation GenerateRollActuatorCommand is
  components: ;
  inputs: LimitedRollRateCommand, ManualRollTrimCmd, ActualFlightPath,
    AutoRollTrimCmd;
  outputs: RollTrimPosition, DesiredRollActuatorPos;
  description: (*
    Generates the roll actuator (aileron / spoiler) position commands based
    on roll rate and trim commands. *);
end GenerateRollActuatorCommand;

operation MoveRollActuator is
  components: ;
  inputs: DesiredRollActuatorPos, ExternalForcesOnActuator;
  outputs: RollActuatorPosition;
  description: (*
    Receives the desired roll actuator position and attempts to move the roll
    actuator to that position. *);
end MoveRollActuator;

operation DisplayRollEnvelopeProtectStatus is
  components: ;
  inputs: RollEnvelopeProtectStatus;
  outputs: DisplayedRollEnvelopeProtectStatus;
  description: (*
    Results from the allocation of Provide Roll Envelope Protection to the
    FCS with a probability of loss of function of <10E-6.  Provide
    RollEnvelopeProtection has a probability of loss of function of < 10E-9
    and thus the crew has responsibility for roll envelope protection when
    not performed by the FCS.  Thus the crew must be aware of envelope
    protect status, hence the function requirement to Display Roll Envelope
    Protect Status. *);
end DisplayRollEnvelopeProtectStatus;

operation ProvideRollEnvelopeProtection is
  components: ;
  inputs: RollAngle (*Note:Why not ActualFlightPath as in
    operation ProvideLongitudinalEnvelopeProtection on pg. 133*),
      LimitedAutoRollcommand, ManualRollRateCommand;
    outputs: LimitedRollRateCommand, RollEnvelopeProtectStatus;
    description: (*
      Monitors actual roll angle and commanded roll rate and modifies the roll
      rate command as necessary to prevent the roll angle from exceeding
      certain limits. *);
  end ProvideRollEnvelopeProtection;

  operation LimitAutoRollCommands is
    components: ;
    inputs: AutoRollRateCommand;
    outputs: LimitedAutoRollCommand;
    description: (*
      Limits the autopilot control authority and protects against failures (in
      particular hardover or oscillatory failures) in the autopilot. *);
  end LimitAutoRollCommands;

  (**** Pilot Roll Interface, pp. 161-162 ****)
  operation ConvertForcesToDisplacement is
    components: ;
    inputs: RollRateForce, RRCmdFeelForce;
    outputs: RollRateCmdDispl;
    description: (*
      Receives the pilot force and feedback feel force and generates a
      displacement. *);
  end ConvertForcesToDisplacement;

  operation GenerateRollFeelForce is
    components: ;
    inputs: RollRateCommand;
    outputs: RRCmdFeelForce, RRCmdFeelForce;
    description: (*
      Generates a force to feedback to the pilot which is an indication of the
      commanded roll rate. *);
  end GenerateRollFeelForce;

  operation TranslateRRDisplToRRCommand is
    components: ;
    inputs: RollRateCmdDispl;
    outputs: RollRateCommand, RollRateCommand;
    description: (*
      Translates the sidestick controller displacement to a roll rate command.
       *);
  end TranslateRRDisplToRRCommand;

  operation TranslateTrimForceToTrimCommand is
    components: ;
    inputs: RollTrimForce;
    outputs: RollTrimCommand;
    description: (*
      Converts the physical displacement generated by the physical force
      exerted by the pilot into a trim command for use by the FCS. *);
  end TranslateTrimForceToTrimCommand;

end FlightControlSystemRollFunctions;
```

```
(****************************************************************************
 * Module ControlYaw from pp. 169 - 193
 ****************************************************************************)
module ControlYaw;
  operation ControlYaw is
    components: GenerateDirectionalTrimCommand, DisplayDirectionalTrimPosition,
      GenerateYawActuatorCommand, EngineOutControlAugmentation,
      MoveYawActuator, ProvideYawEnvelopeProtection, GeneateSideslipCommand,
      LimitAutoSideslipCommands;
    inputs: TargetFlightPath, ActualFlightPath, EngineThrust,
      ExternalForcesOnYawActuator, SideslipAngle (* Inconsistent pp. 87 versus
      169 *);
    outputs: DisplayedDirectionalTrimPos, YawActuatorPosition;
    description:
      (* Controls the aircraft directional axis. *);
  end ControlYaw;

  operation GenerateDirectionalTrimCommand is
    components: ;
    inputs: (* None -- suspicious. *);
    outputs: ManualDirectionalTrimCmd, AutoDirectionalTrimCmd;
    description: (*
      Generates directional trim commands to offset asymmetries such as engine
      out and lateral winds.  Note: inconsistent names pp. 1. *);
  end GenerateDirectionalTrimCommand;

  operation DisplayDirectionalTrimPosition is
    components: ;
    inputs: DirectionalTrimPosition;
    outputs: DisplayedDirectionalTrimPos;
    description: (*
      Displays the position for the directional trim actuator to the crew. *);
  end DisplayDirectionalTrimPosition;

  operation GenerateYawActuatorCommand is
    components: ;
    inputs: LimitedSideslipCommand, ManualDirectionalTrimCmd,
      AutoDirectionalTrimCmd, ActualFlightPath, ECAYawCommand;
    outputs: DirectionalTrimPosition, DesiredYawActuatorPosition;
    description: (*
      Generates the sideslip actuator (rudder) position command based on the
      limited sideslip command, directional trim command and the engine out
      control augmentation command. *);
  end GenerateYawActuatorCommand;

  operation EngineOutControlAugmentation is
    components: ;
    inputs: EnginesThrust;
    outputs: ECAYawCommand;
    description: (*
      Monitors the engine thrust and generates a yaw command to assist the
      pilot in compensation for an engine out situation.  In particular it
      helps relieve pilot workload in takeoff and go around which are high
      pilot workload situations. *);
  end EngineOutControlAugmentation;
```

```
operation MoveYawActuator is
  components: ;
  inputs: DesiredYawActuatorPosition, ExternalForcesOnYawActuator;
  outputs: YawActuatorPosition;
  description: (*
    Receives the desired yaw actuator position and attempts to move the yaw
    actuator to that position. *);
end MoveYawActuator;

operation ProvideYawEnvelopeProtection is
  components: ;
  inputs: SideslipAngle, ManualSideslipComand, LimitedAutoSideslipCommand;
  outputs: LimitedSideslipCommand;
  description: (*
    Monitors the commanded sideslip and the actual sideslip and modifies the
    sideslip command to prevent the sideslip angle from exceeding unsafe
    limits. *);
end ProvideYawEnvelopeProtection;

operation GeneateSideslipCommand is
  components: ;
  inputs: TargetFlightPath, ActualFlightPath;
  outputs: AutoSideslipCommand, ManualSideslipCommand;
  description: (*
    Involves the generation of sideslip commands to allow for decrab for
    landings, performing coordinated turns and offsetting certain
    asymmetries. *);
end GeneateSideslipCommand;

operation LimitAutoSideslipCommands is
  components: ;
  inputs: AutoSideslipCommand;
  outputs: LimitedAutoSideslipCommand;
  description: (*
    Limits the autopilot control autority and protects against failures (in
    particular hardover or oscillatory failures) in the autopilot. *);
end LimitAutoSideslipCommands;

operation GenerateSideslipCmdManual is
  components: ;
  inputs: ActualFlightPath, TargetntFlightPathNBManualSideslipCommand,
    ManualModeEngaged;
  outputs: ManualSideslipCommand;
  description: (*
    Involves the generation of a sideslip command manually (i.e., by the
    crew) as a result of comparing the actual and desired flight path
    (including attitudes). *);
end GenerateSideslipCmdManual;

(* NOTE: Next to ops are generic and should, accordingly, appear in another
    module.  Cf. GenerateRollRateCommand (pg. 151) and
    GenerateFlightPathCommand (pg. 127). *)
operation MakeManualVsAutoFlightModeDecision is
  components: ;
  inputs: none;
  outputs: ManualFlightMode, AutoFlightMode;
```

```
    description: (* *);
  end MakeManualVsAutoFlightModeDecision;

  operation EngageManOrAutoOperation is
    components: ;
    inputs: ManualFlightMode, AutoFlightMode;
    outputs: ManualModeEngaged, AutoModeEngaged;
    description: (*
      Involves the generation of a sideslip command automatically (i.e., by a
      computer. *);
  end EngageManOrAutoOperation;

  operation GenerateSideslipCmdAuto is
    components: ;
    inputs: TargetFlightPath, ActualFlightPath, AutoModeEngaged;
    outputs: AutoSideslipCommand;
    description: (*
       *);
  end GenerateSideslipCmdAuto;

  object ExternalForcesOnYawActuator instance of ExternalForcesOnActuator is
    operations: ControlMissionFlight, ControlRoll, ControlYaw, MoveYawActuator;
    description: (*
      All forces (in particular environmental forces) other than
      actuation forces actin on the yaw actuation system. *);
  end ExternalForcesOnYawActuator;

end ControlYaw;

(*******************************************************************************
 * Module FlightControlSystemYawFunctions from pp. 179-193
 *******************************************************************************)
module FlightControlSystemYawFunctions;

  operation FlightControlSystemYawContext is
    components: PerformPilotFunctions, PerformCopilotFunctions,
      FlightControlSystemYawFunctions, PerformAutoFlightSystemFunctions;
    inputs: (* Unclear -- see pg. 179. *) ;
    outputs: YawActuatorPosition;
    description: (* *);
  end FlightControlSystemYawContext;

  operation FlightControlSystemYawFunctions is
    components: ProvidePilotYawInterface, ProvideCopilotYawInterface,
      DisplayDirectionalTrimPosition, ResolveYawControlContentions,
      GenerateYawActuatorCommand, EngineOutControlAugmentation,
      MoveYawActuator, DisplayEnvelopeProtectStatus,
      ProvideYawEnvelopeProtection, LimitAutoSideslipCommands;
    inputs: PilotDirectionalTrimForce, PilotSideslipForce,
      CopilotDirectionalTrimForce, CopilotSideslipForce,
      AutoDirectionalTrimCmd, AutoSideslipCommand
      (* plus maybe the following due to ambiguity on pp. 179 versus 183: *)
      , ActualFlightPath, EngineThrust, ExternalForcesOnActuator,
      SideslipAngle;
    outputs: ;
    description: (* *);
```

```
end FlightControlSystemYawFunctions;

operation ProvidePilotYawInterface is
  components: ConvertForceToDisplacement, GenerateSideslipFeelForce,
    TranslateSideslipDisplCmd, TranslateDirecTrimForceToCommand;
  inputs: PilotSideslipForce, PilotDirectionalTrimForce;
  outputs: PilotSideslipCmdFeelForce, PilotDirectionalTrimCmd,
    PilotSideslipCommand;
  description: (*
    Converts the signal received from the pilot in the form of a force
    exerted by the pilot's hand into a sideslip signal to be used by the FCS.
    It also provides the pilot with a feedback force proportional to the
    command sideslip angle. *);
end ProvidePilotYawInterface;

operation ProvideCopilotYawInterface is
  components: ;
  inputs: CopilotSideslipForce, CopilotDirectionalTrimForce,
    CopilotSideslipCommand, CopilotDirectionalTrimCmd;
  outputs: CopilotSideslipCmdFeelForce;
  description: (*
    Provides the same function for the copilot as the
    ProvidePilotYawInterface does for the pilot. *);
end ProvideCopilotYawInterface;

operation DisplayDirectionalTrimPosition is
  components: ;
  inputs: DirectionalTrimPosition;
  outputs: DisplayedDirectionalTrimPos;
  description: (*
    Displays the position of the directional trim actuator to the crew. *);
end DisplayDirectionalTrimPosition;

operation ResolveYawControlContentions is
  components: ;
  inputs: PilotSideslipCommand, PilotDirectionalTrimCmd,
    CopilotSideslipCommand, CopilotDirectionalTrimCmd;
  outputs: ManualSideslipCommand, ManualDirectionalTrimCmd;
  description: (*
    Generated as a result of the assignment of the GenerateSideslipCmdManual
    to both the pilot and copilot. *);
end ResolveYawControlContentions;

operation GenerateYawActuatorCommand is
  components: ;
  inputs: LimitedSideslipCommand, ManualDirectionalTrimCmd, ActualFlightPath,
    AutoDirectionalTrimCmd, ECAYawCommand;
  outputs: DirectionalTrimPosition, DesiredYawActuatorPosition;
 description: (*
    Generates the sideslip actuator (rudder) position command based on the
    limited sideslip command, directional trim command and the engine out
    control augmentation command. *);
end GenerateYawActuatorCommand;

operation EngineOutControlAugmentation is
  components: ;
  inputs: EnginesThrust;
  outputs: ECAYawCommand;
  description: (*
    Monitors the engine thrust and generates a yaw command to assist the
    pilot in compensating for an engine out situation.  In particular it
    helps relieve pilot workload in takeoff and go around which are high
    pilot workload situations. *);
end EngineOutControlAugmentation;

operation MoveYawActuator is
  components: ;
  inputs: DesiredYawActuatorPosition, ExternalForcesOnActuator;
  outputs: YawActuatorPosition;
  description: (*
    Receives the desired yaw actuator position and attempts to move the yaw
    actuator to that position. *);
end MoveYawActuator;

operation DisplayYawEnvelopeProtectStatus is
  components: ;
  inputs: YawEnvelopeProtectStatus;
  outputs: DisplayedYawEnvelopeProtectStatus;
  description: (*
    Results from the allocation of ProvideYawEnvelopeProtection to the FCS
    with a probability of loss of function < 10E-6.  YawEnvelopeProtection
    has a proability of loss of function < 109E-9 and thus the crew has
    responsibility for yaw envelope protection when not performed by the FCS,
    hence the crew must be aware of the envelope protection status which
    leads to this functional requirement. *);
end DisplayYawEnvelopeProtectStatus;

operation ProvideYawEnvelopeProtection is
  components: ;
  inputs: SideslipAngle, LimitedAutoSideslipCommand, ManualSideslipCommand;
  outputs: LimitedSideslipCommand, YawEnvelopeProtectStatus;
  description: (*
    Monitors the commanded sideslip and the actual sideslip and modifies the
    sideslip command to prevent the sideslip angle from exceeding unsafe
    limits. *);
end ProvideYawEnvelopeProtection;

operation LimitAutoSideslipCommands is
  components: ;
  inputs: AutoSideslipCommand;
  outputs: LimitedAutoSideslipCommand;
  description: (*
    Limits the autopilot control authority and protects against failures (in
    particular hardover or oscillatory failures) in the autopilot. *);
end LimitAutoSideslipCommands;

(**** Pilot Yaw Interface, pp. 187-188 ****)

operation ConvertForceToDisplacement is
  components: ;
  inputs: SideslipForce, SideslipFeelForce;
  outputs: SideslipCommandDispl;
```

```
    description: (*
      Receives the pilot force and feedback feel force and generates a
      displacement. *);
  end ConvertForceToDisplacement;

  operation GenerateSideslipFeelForce is
    components: ;
    inputs: SideslipCommand;
    outputs: SideslipFeelForce;
    description: (*
      Generates a force to feedback to the pilot which is an indication of the
      commanded sideslip angle. *);
  end GenerateSideslipFeelForce;

  operation TranslateSideslipDisplCmd is
    components: ;
    inputs: SideslipCommandDispl;
    outputs: SideslipCommand;
    description: (*
      Translates the displacement (rudder pedal) to a sideslip command. *);
  end TranslateSideslipDisplCmd;

  operation TranslateDirecTrimForceToCommand is
    components: ;
    inputs: DirectionalTrimForce;
    outputs: DirectionalTrimCommand;
    description: (*
      Converts the physical displacement generated by the physical force
      exerted by the pilot into a trim command for use by the FCS. *);
  end TranslateDirecTrimForceToCommand;

end FlightControlSystemYawFunctions;


(******************************************************************************
 * Module ControlAerodynamicBraking from pp. 198-219, and AE diagrams
 * pp. 218 - 219
 *****************************************************************************)
module FlightControlSystem;

  object FlightControlSystem is
    components: FlightControlComputer, SpeedBrakeController,
      HightLiftController, Displays, HightLIftSystem, RudderSystem,
      SpoilerSystem, AileronSystem, ElevatorStabilizerSystem,
      SidestickControllers, RudderPedals;
    description: (*
      The primary agent, along with Crew members, to execute flight control
      operations *);
  end FlightControlSystem;

  operation PerformAutoFlightSystemFunctions is
    components: ;
    inputs: ;
    outputs: ;
    description: (*
      The AEAuto-FlightSystem ''Architectural Element''. *);
  end PerformAutoFlightSystemFunctions;


  object class Computer is
    components: ;
    operations: ;
    description: (* *);
  end Computer;

  object class Sensor is
    components: ;
    operations: ;
    description: (* *);
  end Sensor;

  object class SurfaceActuator is
    components: ;
    operations: ;
    description: (* *);
  end SurfaceActuator;

  object class Command is
    components: HowActuated, AffectedAircraftComponents (* ... *) ;
    description: (*
     The high-level class of control commands that are generated by either
     the crew or flight control system. *);
  end Command;


  (***** Control System Signal Transmission (F.C.S.1), Pg. 199 *****)

  object Communicant is
    components: Computer | Sensor | SurfaceActuator;
    operations: ;
    description: (* *);
      (* One of the classes of objects between which data communications take
      place. *)
  end Communicant;

  object DataBus is
    components: ;
    operations: TransmitData: (Communicant, Communicant, DataBus) -> (boolean);
    description: (* *);
  end DataBus;

  (***** Control System Computation Requirements (F.C.C.1), Pg. 223 *****)

  var cm1, cm2: Communicant;
      db: DataBus;
  axiom
    forall (cm1, cm2:Communicant, db: DataBus)
      if TransmitData(cm1, cm2, db)
      then (db.Type = Electrical or db.Type = Optical) and
         (db.Speed > MinimumDataCommSpeed);

end FlightControlSystem;
```

References

1. Frincke, Deborah; Wolber, Dave; Fisher, Gene; Cohen Gerald: Requirements Specification Language (RSL) and Supporting Tools. NASA Contractor Report 189700, July 92.

2. McLees, R. E., and Cohen, G. C.: An Example of Requirements for Advanced Subsonic Civil Transport (ASCT Flight Control System Using Structured Techniques. NASA Contractor Report NAS1-187526, October 1991.