Using a Formal Description Technique to Model Aspects of a Global Air Traffic Telecommunications Network

—

James H. Andrews
Dept. of Computer Science
University of Western Ontario
London, Ont., Canada
andrews@csd.uwo.ca
(formerly of UBC CS)

—

Nancy A. Day
Dept. of Computer Science
University of British Columbia
Vancouver, BC, Canada
day@cs.ubc.ca

—

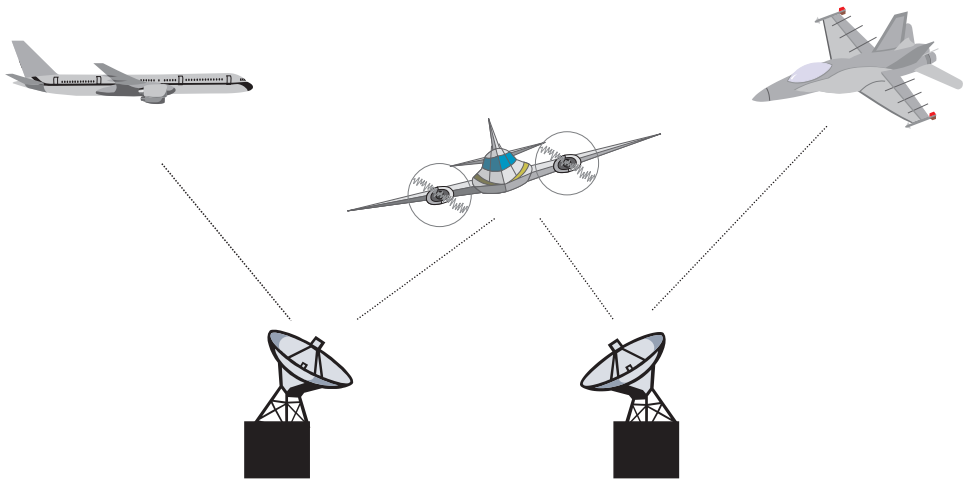
Jeffrey J. Joyce
Hughes Aircraft of Canada Limited
Richmond, BC, Canada
jjoyce@ccgate.hac.com

The FormalWARE Project

`http://www.cs.ubc.ca/nest/isd/FormalWare/`

- 2-year project (1996-1998)
- Investigates industrial use of formal methods in the development of critical systems
- Principal Investigator: Jeff Joyce, Hughes Aircraft of Canada Limited
- Funded by:
  - Hughes Aircraft of Canada Limited (HACL)
  - MacDonald Dettwiler (MDA)
  - British Columbia Advanced Systems Institute (BCASI)
- Personnel at: HACL, MDA, UBC Computer Science, UBC Electrical Engineering, University of Victoria Computer Science

Aeronautical Telecommunications Network (ATN)



- Proposed software system supporting air traffic control
- Software on aircraft, in ground stations communicate
- Protocol requirements stated in "SARPs" documents
- Requirements developed by ICAO

The Formalization Work

Purpose:

- Develop expertise in ATN at Hughes
- Identify ambiguous and unclear passages of SARPs
- Test viability of formal methods
- Model-checking for safety, liveness (Day, future work)

Tools:

- **Statecharts** formalism
- **S** formal description notation
- **Fuss** typechecker

Results:

- Broad cross-section of SARPs formalized
- Some problems found in SARPs
- Ongoing work on model-checking
- Benefitted from using general logic-based notation

Structure of the ATN

ATN (Aeronautical Telecommunications Network):

- Based on OSI model
- Concerns Application Entities (AEs)
- Each AE =

  Association Control Service Element (ACSE) +
  Application Service Element (ASE) +
  Control Function (CF)

- Four types of ASEs
- Each type has "air" and "ground" variant

(page with “structure of an AE” diagram)

Standards and Recommended Practices (SARPs)

ATN specification documents

- About 1000 pages of specification text
- Mostly concerns specifying ASEs
- State machine-based
- Contain state transition tables
- Conditions on transitions sometimes complex

(page with SARPs text)

(page with SARPs state table)

Statecharts

Background:

- Developed 1987 by David Harel
- Communicating hierarchical state machines
- Broadcast communication

Purpose of our use:

- Gaining popularity in aerospace sector
- SARPs already state machine-based
- Had expertise within our group
- Had machine-readable formal semantics

S and Fuss

S:

- General-purpose formal description notation
- Developed 1994 by Joyce, Day, Michael Donat
- Based on typed higher order logic

Purpose of our use:

- Need to express complex transitions
- Formal semantics of statecharts expressed in S
- Expertise within group

Fuss:

- Typechecking program for S specifications
- Use analogous to `lint` or "clean compiling"
- Allows types to be inferred a la ML

Statechart Model of ATN

(picture of statechart model)

Modelling effort:

- Carried out by 5 grad students and 1 research associate
- Model typechecked and integrated
- Model-checking work ongoing (Day)
- CCS, customized Prolog models developed for comparison

(excerpts from S text, to be described verbally)

Problems in SARPs

Minor problems:

- Mostly to do with minor lack of clarity
- To be dealt with in interpretation notes

Major problem (identified Dec 1996):

- Usually ACSE sends 1 message per input message
- On protocol errors, sends 2
- Not clear how CF (Control Function) handles this
- Reported to relevant ICAO committee, acted on

Lessons Learned

Formalization exercise:

- Formalization helps identify ambiguities

Classification of assumptions:

- Simplifying assumptions: made to "abstract away" details of no interest
- Disambiguating assumptions: made to clear up ambiguities in specifications

Use of general-purpose logic-based formal description notation:

- Allows links to other formally-defined components
- Allows natural, "programming language-style" definitions of auxiliary functions, customizing declarations
- Enables use of *any* chosen formalism (e.g. statecharts) with a minimum of overhead

Current and Future Work

- Continue with model checking
- More extensive validation by modelling team
- Maintain model to track changes in SARPs