CS-Note* 15

Andrew Ireland & Bill Ellis
June 19, 2002

Pushing Tin†: An Air Traffic Control Simulator

1 Introduction

This note outlines a software engineering project based around the development of an Air Traffic Control (ATC) simulator. Funded on the NuSPADE grant, this project will employ Tommy Ingulfsen over the summer vacation to develop the foundations of the simulator using Ada and SPARK. This "toy safety critical software" project will provide us with:

- our own in-house demonstrator of the SPARK approach to high integrity Ada;
- an additional source of VCs for the grant;
- case study material for the High Integrity Computing module that starts in autumn 2002.

The development of an ATC simulator is too ambitious for an 8 week project. For this reason we have identified a safety critical core, essentially a model of the terminal airspace around an airport and an Automatic Conflict Detection (ACD) system. The remainder of this note presents background material together with the project requirements.

2 Background

As mentioned above, the simulator will be based on an airport's terminal airspace[1]. Moreover, the simulator will focus upon the approach control, the phase of the air traffic control process where aircraft inbound to an airport are guided to a final approach point. The simulation will take the form of a radar scope where aircraft enter from 3 directions, each entry point will correspond to a VOR[2] beacon. Aircraft will be tagged with a callsign, flight level and heading. The role of the approach controller is to issue commands (heading and flight level) that will guide aircraft onto the final approach to the airport while maintaining safe separation. The use of a holding stack is envisaged in order to support safe separation during busy points within a simulation. However, support for a command interface and a holding stack will be a longer-term goal.

†Film buffs see `http://www.foxmovies.com/pushingtin/`

*Notes in this series are for ϵ-baked ideas, for $1 \geq \epsilon \geq 0$. Only exceptionally should they be cited or distributed outwith the CLAM-SPARK project (EPSRC GR/R24081).

[1]The airport will be fictional.

[2]A Very high frequency Omnidirectional Range beacon is a key navigational aid.

3 Requirements

Within the short-term, a significantly cut down version of the simulator is required. A simulation will be driven by scripts that define both the flight plans[3] and commands issued to each aircraft by the approach controller. The script mechanism will enable both safe and unsafe flows of aircraft to be modelled. By supporting unsafe flows we will be able demonstrate the ACD system – our toy safety critical software component. The detailed functional requirements are:

R1: the simulator will model the terminal control area around a fictional airport;

R2: aircraft will enter the terminal control area at 3 distinct locations each corresponding to a VOR beacon;

R3: aircraft will exit the model once they reach the final approach point, *e.g.* 3 miles from the end of the runway;

R4: simulations will be driven by a script mechanism that will define all aircraft movements;

R5: the simulator will maintain a 2D graphical presentation of the terminal control area, highlighting the position of each aircraft together with their callsign, flight level and heading;

R6: the simulator will detect and suitably indicate aircraft collisions;

R7: the simulator will maintain a "raw radar" model of the terminal control area, *i.e.* 3D co-ordinates for each aircraft;

R8: using the raw radar model, the optional ACD subsystem will provide a suitable alert when aircraft violate the safe separate criteria;

R9: safe separation criteria, terminology etc will be as specified by the CAA [4].

R10: the simulator will be designed so as to easily support an interactive component[4] at some point in the future;

In addition to the basic functional requirements given above, there are a number of non-functional requirements:

R11: the simulator will be designed and documented using the INFORMED method [1];

R12: the implementation language will be Ada [2];

R13: GtkAda[5] will provide the basis for the GUI component;

R14: the safety critical component, ACD, will be developed using the SPARK approach [3].

4 Conclusion

In the longer-term, the ATC simulator holds the potential for a dynamic 3D software game played out via a 2D GUI. However, achieving the short-term requirements presented above will make a valuable contribution to our work in terms of providing a rich source of verification problems as well as a fun demonstrator for the NuSPADE project!

References

[1] P. Amey. *INFORMED Design Method for SPARK.* Praxis Critical Systems, 1999. Restricted Access.

[2] J. Barnes. *Programming in Ada 95.* Addison-Wesley, 1995.

[3] J. Barnes. *High Integrity Ada: The Spark Approach.* Addison-Wesley, 1997.

[4] G. Duke. *Air Traffic Control.* ian Allan Publishing, 2001.

---

[3] Time and entry point into the simulation.
[4] An interface that will support approach controller style interactions.
[5] See `http://libre.act-europe.fr/GtkAda/` and `ftp://cs.nyu.edu/pub/gnat/gtkada/`