Object-oriented Software Considerations in Airborne Systems and Equipment Certification

DO-178C and Object-oriented Technology

Mike Elliott

The Boeing Company

Michael.R.Elliott2@boeing.com

When?

- June 4, 1996

Where?

- June 4, 1996
- Kourou, French Guiana

What?

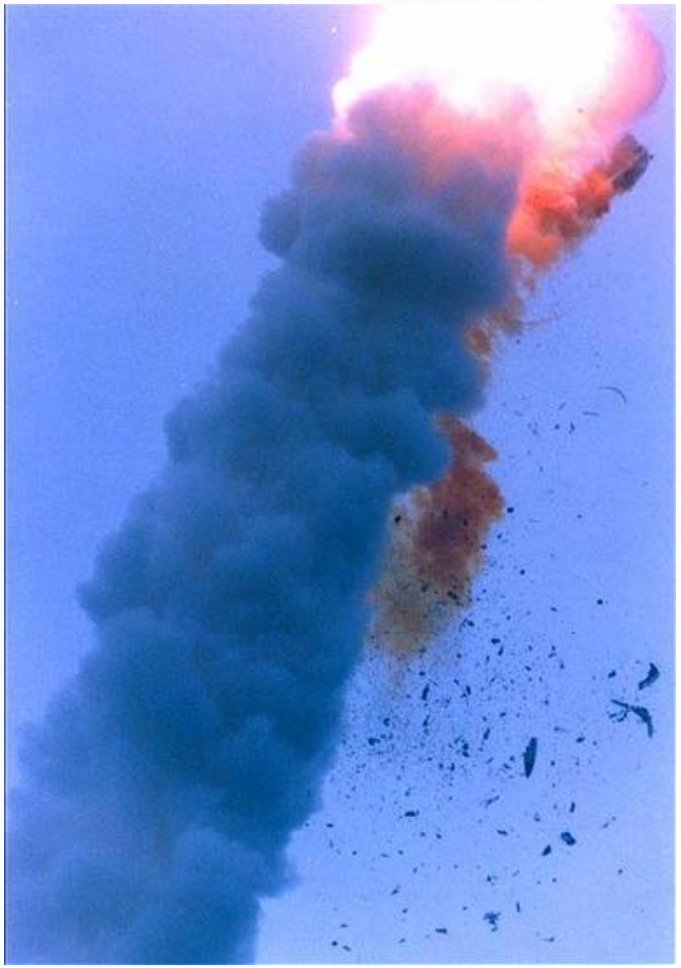
- June 4, 1996
- Kourou, French Guiana
- Ariane 5
  - Flight 501

Why?

- June 4, 1996
- Kourou, French Guiana
- Ariane 5
  - Flight 501
- Type conversion – 64 bits to 16 bits

Level A - Catastrophic

- June 4, 1996
- Kourou, French Guiana
- Ariane 5
  - Flight 501
- Type conversion – 64 bits to 16 bits
- Unhandled exception
  - What we're trying to avoid



Certification for Airworthiness

- Airworthiness is determined on a country by country basis
  - Federal Aviation Administration (FAA) – USA
  - European Aviation Safety Agency (EASA) – Western Europe
    - In conjunction with local authorities (CAA Netherlands, for example)
  - Transport Canada – Canada
  - Civil Aviation Administration of China (CAAC) - China

Standards

- *One ring to rule them all!*
  - There is only one standard
- Developed through RTCA and EUROCAE
  - Committees and working groups composed of “volunteers”
- Documents created by consensus
  - Where “consensus” changes as the effort grows longer

Currently DO-178B/ED-12B

- *Software Considerations in Airborne Systems and Equipment Certification*
  - International standard for airworthiness of systems containing software, for use in civil airspace
- Software is not certified
  - Airplanes
  - Engines
  - Propellers
  - Auxiliary Power Units (UK only)

RTCA, Inc.

- A not-for-profit corporation formed to advance the art and science of aviation and aviation electronic systems for the benefit of the public.
- Functions as a Federal Advisory Committee and develops consensus-based recommendations on contemporary aviation issues.
- Recommendations used as the basis for government decisions as well as the foundation for many FAA TSOs

EUROCAE

- *European Organization for Civil Aviation Equipment*
- Provide a European forum for resolving technical problems with electronic equipment for air transport
- Deals exclusively with aviation standardization and related documents
- Documents referred to as a means of compliance with European TSOs

DO-178 / ED-12 Series

- Used as the means by which certification authorities determine that aircraft and engines containing software can be granted airworthiness certification for civil airspace
- Specifies the means by which software is produces and verified in order to obtain airworthiness certification
- Required reading by thousands of software developers worldwide

Four versions

- DO-178/ED-12
  - 1980 – 1982
- DO-178A/ED-12A
  - 1983 – 1985
- DO-178B/ED-12B
  - 1989 – 1992
- DO-178C/ED-12C
  - 2005 – 2011 (maybe)

DO-178/ED-12

- RTCA SC-145, May 1980
  - *Digital Avionics Software*
- EUROCAE WG-12, 1980
  - ED-35 *Recommendations on Software Practice and Documentation for Airborne Systems*
- EUROCAE and RTCA developed common guidance
- DO-178 published in January 1982, shortly followed by ED-12

DO-178/ED-12 Impact

- Provide a basis for communications between applicants and certification authorities
  - Set of *best practices*
- Applicants should ***meet the intent***
  - No specific objectives to be achieved
  - No guidance as to how to achieve success
- Three tiered system
  - Critical, essential and non-essential

DO-178/ED-12 Acceptance

- It did provide a linkage between software verification efforts and Federal Aviation Regulations and Technical Standard Orders (European and American)
- Consensus was quickly reached that a revision was needed
  - No discussion of software process
  - Unclear what artifacts were needed for certification authorities

DO-178A/ED-12A

- *Software Considerations in Airborne Systems and Equipment Certification*
  - RTCA SC-152, 1983 – 1985
  - EUROCAE published identical technical content
- Quite different from DO-178/ED-12
  - Rigorous requirements
    - Software process
    - Software production
    - Process documentation
    - Process History

DO-178A/ED-12A Impact

- Many new companies began producing avionics with software
- Lack of experience and understanding of how to satisfy DO-178A
- Entire projects failed due to disconnect between certification authorities and applicants
- Widespread differences in certification authorities on a per-region basis

DO-178B/ED-12B

- SC-167 and WG-12 – summer 1989
- Review and revision of DO-178A
- Fundamental changes to DO-178A/ED-12A
  - Software criticality levels
  - Strong emphasis on requirements-based testing
  - More rigorous definition of software process
  - More documentation needed from applicants for things like SQE and process

Levels A, B, C, D, E

- Software Design Assurance Level / Software Criticality Level

A – Catastrophic

B – Hazardous

C – Major

D – Minor

E – No Effect

Level A – Catastrophic

- Failure may cause a crash
  - Fuel management system fails to deliver fuel from the reserve tank to the wing tanks causing engine flameout due to fuel exhaustion
  - AirTransat flight 206 – August 24, 2001

Level B - Hazardous

- Failure has a large negative impact on safety or performance
- Reduces the ability of the crew to operate the aircraft due to physical distress or a higher workload
- Causes serious or fatal injuries among the passengers
  - Lufthansa Flight 2904 – September 14, 1993

Level C - Major

- Failure is significant, but has a lesser impact than a Hazardous failure
  - Nobody gets killed
  - Cabin fire monitoring system releases fire-suppression gases into the cabin when there was no fire to suppress

Level D - Minor

- Failure is noticeable, but has a lesser impact than a Major failure
  - Database of navigation aids becomes unavailable causing a change in route
  - Smoke is seen to rise from the in-flight entertainment console

Level E – No Effect

- Failure has no impact on safety, aircraft operation, or crew workload
  - Emergency Transponder Beacon fails

OOTiA

- Concern was expressed that 178 series did not address new paradigms
- *Handbook for Object-Oriented Technology in Aviation*
  - Best practices guide
- Work began in 2000
- Representatives from NASA, BF Goodrich, Boeing and others
- Discontinued in 2005

Problems with DO-178B/ED-12B

- Configuration Control too high for tools
- Common mode errors not really addressed
- Not enough goal oriented
  - Forces the applicant to address the objectives directly - may not be applicable
  - Objectives in tables are not all objectives
    - Some are specific means of compliance (MC/DC) so no alternative means of compliance is feasible

SC-205/WG-71

- In 2004, FAA and EASA both wanted a revision to DO-178B/ED-12B
  - Legacy from the clarification group
  - Lessons learned from DO-178B/ED-12B
  - Newly available techniques
  - Not enough goal oriented
  - COTS issues not addressed
- SC-205/WG-71 formed in early 2005
  - First plenary session – London, 2005

Supplements for optional methods

- DO-178C/ED-12C designed to be extended through the addition of supplements
  - Tools Qualification (SG3)
  - Model-Based Development (SG4)
  - Object-oriented and Related Technologies (SG5)
  - Formal Methods (SG6)
- Possibility of future supplements
  - SC-216/WG-72 working on airborne security aspects of airworthiness

SG5 – OO & Related Technologies

- Initially expected to massage, correct and amplify on OOTiA
- This brought about substantial discord
- SG5 became the problem child after two years of little progress
  - Vienna plenary – new people and a new direction
  - Reorganization of what it means to be a supplement
  - Abandonment of OOTiA

Supplement Purpose

- Provide guidance for the production of software using OO and related technologies
- Provide a common framework for evaluation of OO&RT developed software for airworthiness in civil airspace
- Provide guidance for the evidence that compliance has been achieved
- Provide one alternative to the procedural-based orientation of DO-178B / ED-12B

Address Coding Issues

- Parametric polymorphism
- Exception handling
- Code re-use
- Dead and deactivated code
- Component-based development
- Automated resource management
- Virtualization
- Closures

Impact on Process

- Allow more modern software practices to be utilized
  - Move away from heavily process-oriented methods
  - Towards that end, just mentioning something gives the applicant and certification authority common grounds for discussion

Impact on Testing

- Decrease the testing burden
  - Allow practitioners to plan testing in a hierarchical manner which provides a mechanism for re-use of testing results
  - Where LSP really comes into play

Impact on Re-use

- More easily permit the use of pre-built, reusable software
  - DO-178B / ED-12B focused on having everything custom built
  - Provide a path for the adoption of component libraries

Dead and Deactivated Code

- Not really OO, but an example of “related technology”
- Impact on software due to issues with reuse of components
  - Stack with push, pop, peek
  - The peek() method never gets invoked
  - DO-178B/ED-12B – that's dead code

Type Theory

- Once OOTiA was left behind, something was needed as an underlying organizational theme
  - Three compiler guys with formal language theory backgrounds
- Type theory adopted as a unifying means of description
  - Change terminology to match the theory
  - Generics and templates became *parametric polymorphism*

Type Theory Applied

- Consideration of the concept of *type* as the set of legal values a program may assign a particular typed object
- Retention of type consistency became the foundation for applying substitution rules. This impacts:
  - Traceability
  - Code coverage
  - Test result reuse

LSP

- The Liskov Substitution Principle

Let *q(x)* be a property provable about objects *x* of type *T.*

Then *q(y)* should be true for objects *y* of type *S* where *S* is a subtype of *T*.

  - Barbara Liskov – 1987

- LSP became the basis for all our class hierarchy arguments

LSP Counter Example

- A base class SpeedController is created for which subclasses are intended to be implemented for different hardware implementations
- An adjustSpeed(int delta) method is part of the class declaration
- Speed is considered to be the magnitude of the velocity vector, therefore never negative
- Post condition: after adjustSpeed() is invoked with a positive value, the speed cannot be zero

Potential problems

- There is a method timeToGo(int dist) which returns time to go a given distance
- This is computed by dist / speed
- As long is speed is non-zero, this is fine

LSP violation

- Subclass AutoSpeedController
  - Introduces method setSpeed(int speed) which takes a desired speed value and manages the speed itself
  - The adjustSpeed() method is meaningless and therefore stubbed out
- Speed is zero
  - Invoking code uses adjustSpeed() instead of setSpeed()
  - Exception thrown: division by zero

Class Hierarchies

- Directed acyclic graph of subclass to superclass relationships
- Can be a powerful tool in managing complexity
  - Can reduce verification activities
  - Improve understanding, maintainability and re-use
- Generally implemented using language supported features
  - Inheritance, overloading, run-time polymorphism

Hierarchy of Polymorphism

- Universal Polymorphism
  - Parametric
  - Inclusion
- Ad-Hoc Polymorphism
  - Overloading
  - Coercion

Virtualization

- Virtual machines are identified as a potential execution environment
  - The term ‘execution environment’ replaces ‘target computer’ in the core document
- This requires a more precise notion of object code vs. data
  - Java byte code is object code, not data being interpreted by the VM
  - Similar situation with XML being interpreted by an XML parser

Garbage Collection

- Garbage collection is still somewhat controversial in safety-critical software field
  - Needed for typical object-oriented programming practice
- Several real-time collection strategies discussed
  - Time, slack, work
  - Requires available heap space monitoring with degraded mode notification when threshold is reached

Real-Time Garbage Collection

- Other granularities of GC allowed, beyond collection at the individual object level – for example:
  - Scoped memory
  - Immutable memory
- Garbage collection often dismissed as being “too complicated”
  - Degree of complication is 8.5 whereas we can allow no more than 6.6

Memory Management Techniques

- Long been a sore point
  - malloc() used but not free()
- Real-time garbage collectors for Java today
  - Time-based Metronome used by IBM
  - Henrikkson work-based GC used by Sun
  - Siebert work-based GC used by aicas
  - Nilsen concurrent mark/sweep used by Aonix
  - Great research topic

Language Independence

- Conscious decision made to retain programming language independence
  - But some biases crept in
- Focus was on OO languages of the present
  - Ada, Java, C++
- Future issues explicitly called out
  - Closures
- We think we allowed for functional languages
  - We'll find out in 10 years

OO&RT Supplement Acceptance

- Voted on and passed at the Paris plenary, October 29, 2009
  - FAQs and Glossary remained to be completed
- FAQs voted on and passed at the Marseille plenary, June 25, 2010
  - Bulk of the document turned into a discussion paper between plenary sessions
- Problems still remain as of November, 2010

DO-178C/ED-12C Completion

- Long Beach plenary (November 8 – 12, 2010) was supposed to be the wrapup
  - Still having problems with Model-based supplement
  - Revisionists attacking the OO&RT supplement
- Two additional plenaries planned
  - Stockholm – April 11 – 15, 2011 (canceled)
  - Washington, DC – September? 2011
- Rumblings about DO-178D

Acronyms and Abbreviations

- FAA – Federal Aviation Administration
- EASA – European Aviation Safety Agency
- RTCA – RTCA Incorporated
- EUROCAE – European Organization for Civil Aviation Equipment
- OOTiA – Object-Oriented Technology in Aviation
- TSO – Technical Standard Order
- SC-205 – Special Committee 205

WG-71 – Working Group 71

LSP – Liskov Substitution Principle

VM – Virtual Machine

XML – Extended Markup Language

GC – Garbage Collection

OO&RT – Object-oriented and Related Technologies

FAQ – Frequently Asked Question