

SAFETY ENGINEERING GROUP

ASSC C++ Strategy Paper

B Bateman

ERA Report 2005-0293
ERA Project 7D01348
Final Report

Client: MOD (PE)

Client Reference: FBG/01189

ERA Report Checked by:



C Lee
Senior Consultant Engineer
Safety Engineering

Approved by:



K Moore
Head of Safety Engineering

July 05

Ref. 7D01348/4005

© Copyright ERA Technology Limited 2005
All Rights Reserved

No part of this document may be copied or otherwise reproduced without the prior written permission of ERA Technology Limited. If received electronically, recipient is permitted to make such copies as are necessary to: view the document on a computer system; comply with a reasonable corporate computer data protection and back-up policy and produce one paper copy for personal use.

DISTRIBUTION

M. Suitters

Future Business Group (DPA)

Summary

The use of C++ for safety related and safety critical defence applications is becoming more prevalent. Certification of these applications is being carried out on a case-by-case basis. In order to reduce costs the MoD need to define a universal process for practitioners and assessors of these systems. It is therefore considered necessary for a guidance document to be produced for the practitioners of C++ for safety related and safety critical defence applications.

The Avionics Systems Standardization Committee (ASSC) seeks to support the MoD in this requirement and this strategy paper is in response to this aim. The report summarises recent ASSC activities and provides the following recommendations for future support:

1. Provide a forum for MoD and industry to provide input into guidance document requirements, scope and development via a series of workshops. This will provide the benefit of maximizing buy-in to developed guidance.
2. Interface with MISRA to determine scope of their activities and timescales in relation to C++ guidance material development. If appropriate, set up an arrangement for working with MISRA and determine a strategy for addressing the gaps between MISRA's agenda and the MoD's requirements. This will maximize cross-industry participation and hence lower costs in guidance material development.
3. Monitor and report on progress of guidance material development by keeping this strategy paper up to date and holding regular meetings with MoD stakeholders.
4. Liaise with Sub Committee-22 to determine progress and outputs of their guidance material, and if appropriate, set up a working relationship to enable cross-flow of ideas. This may assist with endorsement of UK developed guidance material.
5. Monitor the progress of the upgrade of RTCA DO-178B as regards to the inclusion of Object Oriented Technology (OOT). Determine if there is scope for providing UK input to the development of this update, and if so liaise with SC-205/WG-71 to achieve this aim.
6. Disseminate information to MoD Integrated Project Teams (IPT) and industry via workshops and a software web portal. This will ensure usability of developed guidance material, and provide a basis for contribution to its development.
7. Provide training in the use of guidance material, once again ensuring usability.
8. Hold discussions with key MoD stakeholders as to whether guidance should be developed into a standard, and development of a route to endorsement.

Contents

	Page No.
1 Introduction	6
1.1 Background	6
1.2 Report Structure	6
2 Background – Documents and Working Groups	7
3 MoD and ASSC Progress: C++	8
3.1 DARP ASSC C++ Workshop	8
3.2 C++ Robust Language Research Proposal	9
3.3 ASSC C++ Strategy Meeting	9
3.4 Meeting with MISRA	10
4 Conclusions	11
5 Recommendations and Role of the ASSC	11
5.1 Role of the ASSC	12
6 References	13
APPENDIX A – Object Oriented Technology	16
APPENDIX B – Document Review	18
APPENDIX C – Working Groups	22

Abbreviations List

ASSC	Avionics Systems Standardization Committee
CAA	Civil Aviation Authority
CAST	Certification Authorities Software Team
DARP	Defence and Aerospace Research Partnership
DSTL	Defence Science and Technology Laboratory
EUROCAE	European Organisation for Civil Aviation Equipment
FAA	Federal Aviation Authority
HIJA	High Integrity Java
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Standards Organisation
JSR	Java Specification Request
JTCI	Joint Technical Committee – Information Technology
MIRA	Motor Industry Research Association
MISRA	Motor Industry Software Reliability Association
MoD	Ministry of Defence
NIST	National Institute for Standards and Technology
OO	Object Oriented
OOT	Object Oriented Technology
OOTiA	Object Oriented Technology in Aviation
RI	Reference Implementation
RTCA	Radio Technical Commission for Aeronautics
SBAC	Society of British Aircraft Companies
SC	Sub Committee
TCK	Technology Compatible Kit
UML	Unified Modelling Language

1 Introduction

1.1 Background

The use of Object Oriented (OO) programming languages has revolutionized software engineering, from design processes used, to coding itself. As these languages become more popular (programmer training, tool support etc.) there has been commercial pressure for their adoption in defence applications. For safety related and safety critical systems, choice of programming language cannot be driven by economics alone. ‘Safe’ subsets of languages have been developed to ensure predictability of running software applications and guard against programming errors.

The MISRA C Guidelines is one such example (Ref 1). Developed for the automotive industry, it provides rules for restricting the use of the C programming language to reduce the risk that errors could be introduced by a programmer due to misunderstandings, ambiguities and insecurities associated with software development in C. The guidelines are now supported by tools that can detect most of the rule violations in a C program. With the use of the guidelines supplemented by review and tool support, the use of C is considered to be acceptable for low to medium integrity applications such as SIL 1 or even SIL 2.

There is at present no universally accepted safe set of an OO programming language. There is a selection of research and papers analyzing and identifying vulnerabilities and proposing safe subsets of OO programming languages such as C++ and Java. The use of C++ is being used more frequently in safety related defence applications. Certification of these applications is being carried out on a case-by-case basis. Replication of effort for these projects is an expensive waste of MoD resources.

The role of the ASSC (Avionics Systems Standardization Committee) is to provide an MoD and industry forum to implement appropriate avionics systems standardization strategies. In this role, the ASSC has been supporting the MoD in realizing a strategy for efficiently tackling the certification issues related to the use of C++ in safety related and safety critical applications.

This report provides a summary of the work that the ASSC has been carrying out to achieve these aims, and concludes with a number of recommendations that are considered necessary for achieving uniformity in approach.

1.2 Report Structure

The focus of this report is on OO programming languages not traditionally used for high integrity systems¹, specifically C++. To provide a background for the current situation, Section 2 provides a

¹ Although not included in this report, the next 10-year update to Ada95 (Ravenscar Ada) will provide some OO functionality.

summary of documents and working groups relating to the use of Object Oriented Technology (OOT) and programming languages for safety critical applications. Current MoD activities are presented in Section 3, which includes the outcome of the recent ASSC strategy meeting. Conclusions and recommendations are provided in Sections 4 and 5. As OO programming is a part of OOT, a definition of OOT is presented in Appendix A, in order to provide the reader with background information on the defining principles.

2 Background – Documents and Working Groups

Since the mid 1990s several reports have been produced that examine the issues relating to the use of C++ for safety critical applications. These documents provide an explanation of the issues relating to the use of C++ for safety critical systems, many defining subsets of the language, and are listed as follows (for further information, see Appendix B):

- Binkley, *C++ In Safety Critical Systems* (1996, Ref 2)
- US Nuclear Regulation Commission, *C and C++ Guidelines: Review Guidelines on Software Languages for use in Nuclear Power Plant Safety Systems (NUREG/CR-6463)* (1996, Ref 3)
- Exida, *C/C++ Coding Standard Recommendations for IEC 61508* (2002, Ref. 4)
- FAA – Certification Authorities Software Team (CAST), *Use of C++ Programming Language* (2002, Ref 5)
- QinetiQ, *An Investigation of the Unpredictable Features of the C++ Language* (2004, Ref 6)
- Programming Research Group, *High Integrity C++ Coding Standard Manual* (2004, Ref 7)
- Derek Reinhardt, *Use of C++ Programming Language in Safety Critical Systems* (2004, Ref. 8)

There are five relevant working groups looking at the effects of OO methodologies including programming languages on safety critical system development and certification. None of these are looking at C++ specifically (for further details see Appendix C):

- SC-22
Sub Committee 22 from the ISO/IEC JTC1 (Joint Technical Committee – Information Technology) has recently released a proposal for the preparation of generic guidance for a large number of programming languages, entitled “Guidance for Avoiding Vulnerabilities in Programming Languages”, that will deal with safety and security issues.

- SC-205/WG-71
Working groups from RTCA (SC205 Committee – Software Considerations) and EUROCAE (WG71 – Aeronautical Software) have joined together to produce the new version of RTCA *DO 178B: Software Considerations in Airborne Systems and Equipment Certification* (Ref 9). One of the issues being addressed is OO techniques.
- FAA CAST
CAST is providing guidance for the use of OOT in avionics systems, and expects this guidance to contribute to DO 178C.
- MISRA C
This automotive working group produced the MISRA C Guidelines and members are interested in producing a similar guideline for C++.
- Real-Time for Java Expert Group and the J-Consortium
Both working groups are involved with the development of a subset of Java for real-time high integrity applications.

3 MoD and ASSC Progress: C++

There are three MoD activities of note relating to the use of C++ for safety critical systems. These are the recent Defence and Aerospace Research Partnership (DARP) conference at which the use of C++ for safety critical systems was discussed, a research proposal submitted to the MoD by QinetiQ, and the recent ASSC C++ strategy meeting.

3.1 DARP ASSC C++ Workshop

The DARP conference took place on 26th April 2005. There were two conclusions from the ASSC held C++ workshop (Ref. 10):

1. General support for a 'MISRA-like' approach to the development of a C++ coding standard for critical systems (that is a combination of a language subset and verification obligations). Key factors were identified:
 - Development should be industry led;
 - Clear buy-in/patronage is needed from the customers/certifiers (MoD/CAA/Boscombe Down);
 - The solution must be commercially viable;

- Tool support is highly desirable (though it was not clear if tool specific language extensions, like SPARK annotations, were desirable);
 - The approach should be evolutionary, building upon existing guidance;
 - Guidance should be objective-based rather than prescriptive.
2. Guidance in Def Stan 00-55 as to the suitability of programming languages should be updated to be less prescriptive and more objective focussed.

Additionally, in a keynote presentation in the main DARP conference, Joe Jarzombek (Director for Software Assurance, National Cyber Security Division of the US Department of Homeland Security) announced that a US national company had spent \$2M on the development of a high-integrity subset of C++. This demonstrates the level of effort that commercial organisations are putting into the use of C++.

3.2 C++ Robust Language Research Proposal

As a follow on from research described in Section 2, QinetiQ have submitted proposals to the MoD. There are two aims of the proposed research:

1. To define high integrity programming best practices and the types of hazards that should be mitigated;
2. To define semantics of subsets of programming languages (C and C++).

3.3 ASSC C++ Strategy Meeting

The ASSC C++ Strategy Meeting was held at Abbey Wood on 25th May 2005. Attendees were Flt Lieutenant Nick Higham, and Duncan Williams from the MoD, Clive Pygott from QinetiQ, Paul Casely from DSTL, and Kevin Moore and Beth Bateman from the ASSC.

The purpose of the meeting was to determine a strategy for dealing with the rise in the use of C++ for safety related defence applications. It was decided that due to the proliferation of effort on a case-by-case basis for projects using C++, a coding guidance document was needed for practitioners of safety critical systems that could be used across systems and platforms, and even across industries.

It was agreed that generic guidance would provide an agreed starting point from which to drill down into language specific guidance for the various languages under review (specifically C++). It was agreed that the format this could take could be a set of generic issues that are then expanded for specific languages.

Three different courses of action were identified and are presented in order of desirability:

1. Input into the IEEE Sub Committee-22 work. It was noted that there might be a large overlap between MoD requirements and the work that SC-22 is proposing to undertake. If there were an opportunity to provide an input into this work, this may reduce duplication of effort. Additionally, SC-22's IEEE support could provide a route to endorsement. It was noted that timescales could be very lengthy if this course of action is taken.
2. Work with Motor Industry Software Reliability Association (MISRA) to produce guidance material. MISRA is considering producing C++ guidelines. There is the potential to work with MISRA to produce the guidance material. Obviously there will be differences in agenda such as levels of criticality that are being considered, and regulatory and certification requirements.
3. Provide a defence and aerospace only guidance document. If the previous options are not applicable, guidance that meets the MoD's requirements should be developed independently.

Central to these three options is the use of QinetiQ planned research as a basis (see Section 3.2).

A rough route map was discussed and is presented in Figure 1 showing approximate timescales discussed for the first option.

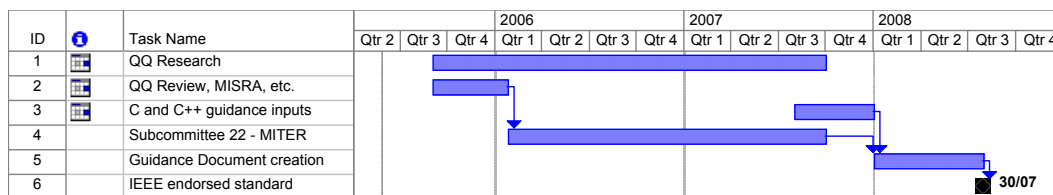


Figure 1 - Route Map

3.4 Meeting with MISRA

A meeting was organized with key MISRA players on 14th July at the Motor Industry Research Association (MIRA) Headquarters. MISRA attendees were: David Ward, MIRA, Chris Tapp, KeyLevel Consultants, Dr Michael Hennel, LDRA, and Steve Montgomery, Ricardo, and from the ASSC, Kevin Moore, Clive Pygott, and Beth Bateman. The meeting agenda was to determine the timescales of MISRA's proposed C++ guidance material and determine the potential for cross-sector co-operation.

The motor industry is starting to use C++ for safety critical applications and there is much need for a guidance document similar to MISRA C to provide support to practitioners and assessors. MISRA C++ will commence in response to this need and will cover all integrity levels. MISRA is aiming to begin work on a C++ guidance document in September this year, and anticipates that material will take a couple of years to develop. As MISRA C is successfully used across industries it is envisaged that MISRA C++ will have the same appeal and should be developed accordingly with cross-industry

and sector involvement. The work that has already been carried out in the defence sector (QinetiQ and Derek Reinhardt, See Appendix B) was considered an ideal basis for rules, and the MISRA attendees were keen to get defence input.

4 Conclusions

C++ is being used in safety related defence applications, and certification is being tackled on a case-by-case basis. This is leading to duplication of IPT effort and ultimately extra cost to the MoD.

There is a body of research presenting the problematic aspects of OO programming languages, such as C++ and Java, for safety critical applications. Additionally, 'safe' sub-sets of these languages have been developed, although these have not as yet been universally adopted.

To reduce duplication of effort and cost to MoD and industry, there is a need for a unified process of certification based on the development of guidance material. The ASSC is supporting the MoD to realize this requirement. Recommendations have been made and are summarized in the next section.

5 Recommendations and Role of the ASSC

The following 11 recommendations are made, the ultimate benefit of which is to reduce cost to the MoD:

1. Develop a guidance document

A general 'MISRA-like' coding guidance document should be developed for C++, which is objective rather than prescriptive based. Consistency via the adoption of a single sub-set and semantics for C++ will guard against duplication of IPT and industry effort at MoD cost.

2. Gain stakeholder involvement

It is vital that MoD and industry are involved at all stages in the development of the guidance, in order to ensure that it meets both their needs, is supported, and ultimately used.

3. Maximize usability of guidance

Guidance should be specific as well as generic, to ensure maximum usability. It is envisaged that generic cross-language guidance be provided, but it is essential that language-specific issues, rules and recommendations support this.

4. Build on existing work

A lot of work has already been carried out (See Section 2). The development of guidance material should build upon this foundation.

5. Enhance understanding of issues

Guidance material should provide rules that are traceable to reasons. It is envisaged that this will enhance practitioners understanding of the guidance material and thus encourage use.

6. Seek buy-in and endorsement

Buy-in to the guidance document is essential if the objective of the exercise is to be achieved. The route to endorsement needs to be considered and a plan put into place for achieving this.

7. Decide whether to produce a guidance document or a standard

A decision needs to be made as to whether the guidance document is developed into a standard.

8. Determine scope of guidance

The DARP workshop and C++ strategy meeting concluded that the C++ programming language be the focus of guidance material. However it needs to be decided whether guidance should cover other OO languages, for example Java is a popular language of which subsets are being developed for high integrity applications. Additionally, the boundary of the guidance needs to be set. For example, it needs to be decided whether or not tool support should be included.

9. Determine support from MISRA

The adoption of the MISRA C guidelines has been very successful, and this success is not limited to the automotive industry. MISRA is shortly due to begin creating similar guidance for C++. It is envisaged that work carried out for MoD's objectives could achieve valuable input from MISRA's activities. There may be gaps between the guidance that MISRA produce and MoD requirements (e.g. certification requirements) and these gaps would need to be filled. It is therefore recommended that liaison be made with MISRA.

10. Achieve synchronization with international activities

Synchronization with programs of work being carried out internationally may be beneficial to avoiding effort duplication. SC-22 should be approached to determine if cross-flow of ideas and developments is possible. Additionally, as DO-178B is being revised, there is an opportunity to coordinate a UK position as regards to C++ involving the MoD, CAA and industry.

11. Support adoption of guidance

Support will be needed for IPTs and industry to adopt the guideline efficiently into work practices.

5.1 Role of the ASSC

The role of the ASSC is to provide a MoD and industry forum to implement appropriate avionics systems standardization strategy. With this in mind, there are 8 tasks that the ASSC could undertake to assist with the implementation of the recommendations made in this strategy paper (See Table 1).

Task	Recommendations addressed
A. Provide a forum for MoD and industry to provide input into guidance document requirements, scope and development via a series of workshops.	2, 3, 4, 5, 6, 7, 8
B. Interface with MISRA to determine scope of their activities and timescales. If appropriate, set up an arrangement for working with MISRA and determine a strategy for addressing the gaps between MISRA's agenda and the MoD's requirements.	9
C. Monitor and report on progress of guidance material development by keeping this strategy paper up to date, and holding regular meetings with MoD stakeholders.	1
D. Liaise with SC-22 to determine progress, outputs, and if appropriate, set up a working relationship to enable cross-flow of ideas.	10
E. Monitor the progress of the upgrade of RTCA DO-178B as regards to the inclusion of OOT. Determine if there is scope for providing a UK input into the development of this update, and if so liaise with SC-205/WG-71 to achieve this aim.	10
F. Disseminate information to MoD IPTs and industry via workshops and a software web portal.	3, 11
G. Provide training in the use of guidance material.	11
H. Hold discussions with key MoD stakeholders as to whether guidance should be developed into a standard, and development of a route to endorsement.	6, 7

Table 1 - ASSC Tasks

6 References

1. MISRA-C: 2004
Guidelines for use of the C language in critical systems
2. Binkley, David, National Institute of Standards and Technology
"C++ In Safety Critical Systems"
February 29, 1996
3. US Nuclear Regulation Commission
C and C++ Guidelines: Review Guidelines on Software Languages for use in Nuclear Power

- Plan Safety Systems
NUREG/CR-6463, 1996
4. Exida
C/C++ Coding Standard Recommendations for IEC 61508
Version V1, Revision R1.0, July 28, 2002
 5. Certification Authorities Software Team (CAST) Position Paper, CAST- 8
Use of the C++ Programming Language
January, 2002
 6. QinetiQ
An Investigation of the Unpredictable Features of the C++ Language
May, 2004
 7. The Programming Research Group
High Integrity C++ Coding Standard Manual
Version 2.2, May, 2004
 8. Flight Lieutenant Derek Reinhardt (Australian RAF)
Use of C++ Programming Language in Safety Critical Systems
MSc Thesis conducted with York University, September 2004
 9. RTCA DO-178B/EUROCAE ED 12B
Software Considerations in Airbourne Systems and Equipment Certification
December, 1992
 10. ASSC DARP C++ Workshop Minutes
7D01348/4006
April, 2005
 11. FAA Handbook for Object Oriented Technology in Aviation (OOTiA)
October 26, 2004
 12. FAA Aircraft Certification Service. *Conducting Software Reviews Prior to Certification*. Job Aid, June 1998
 13. Object-Oriented Technology (OOT) in Civil Aviation Projects: Certification Concerns
Certification Authorities Software Team (CAST) Position Paper, CAST-4
January, 2000
 14. J. Kwon, A. Wellings and S. King
Assessment of the Java Programming Language for Use in High Integrity Systems
Department of Computer Science, University of York, 2002
 15. NIST Special Publication 500-243
Requirements for Real-Time Extensions for the Java Platform, Report from the Requirements Group for Real-time Extensions for the Java Platform
September, 1999

This page is intentionally left blank

APPENDIX A – Object Oriented Technology

OO Technology (OOT) is a software engineering technique that relies on the use of objects, each having its own characteristics and functions, to break a problem down into facets. The FAA's Handbook for Object-Oriented Technology in Aviation (OOTiA) gives the following description of an object:

“An object can be compared to a “black box” at the software level – it sends and receives messages. The object contains both code (methods) and data (structures). The user does not need to have insight into the internal details of the object in order to use the object, hence the comparison to a black box. An object can model real world entities, such as a sensor or hardware controller, as separate software components with defined behaviours.”(Ref. 11, p. 1-4).

OOT methodology is divided between four activities: analysis, design, programming and verification. Figure 2 shows the dependencies between these four methodologies. OO analysis is the “process of defining all classes [superset of specific object types] that are relevant to solve the problem and the relationships and behaviour associated with them” (Ref. 11, p. 1-6). OO design is an extension of OO analysis, in which the relationships and interactions between the classes and objects are defined. OO programming takes the OO design and encodes it in an OO programming language, and finally, OO verification tests the correctness of the OO analysis, design and code.

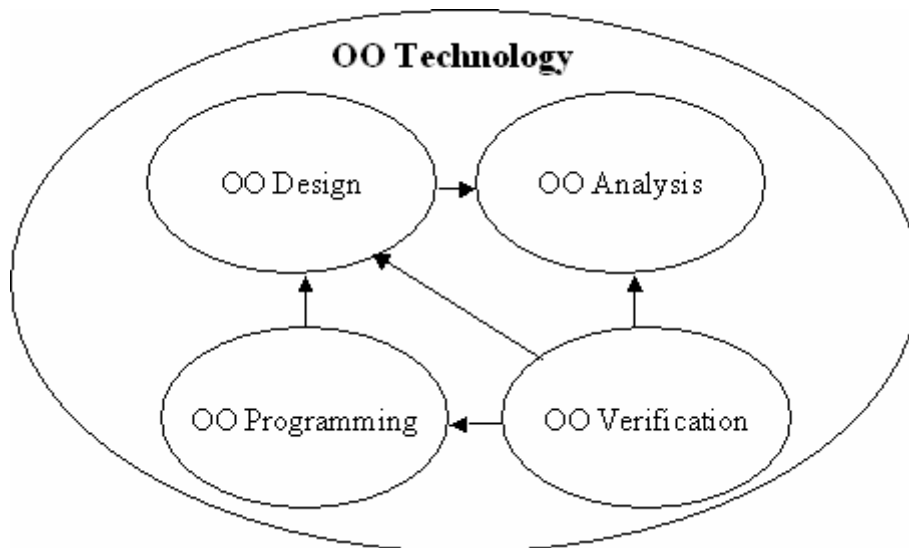


Figure 2: The Relationships between OO Methodologies

The OOTAi describes seven principles of OOT:

1. Abstraction - the essential attributes and methods of an object are defined and the details are hidden;

2. Encapsulation - an object 'owns' the attributes and methods ascribed to it and only those elements that are necessary for the use or manipulation of that object are available, the rest are hidden;
3. Modularity - a problem is divided into modules such as classes, to minimize dependencies;
4. Hierarchy - a relationship between classes is defined so that a higher order class's attributes and methods can be passed down and used by subordinate classes;
5. Typing - a variable is assigned to be an object of a specific class;
6. Concurrency – the ability of several elements of a program to run simultaneously;
7. Persistence - an object can persist in time whilst its use is necessary.

Additionally, an important principle of OOT to safety critical systems is the property of re-use. As an OO program/design is defined using modules, it is possible to re-use these for different applications.

Some of the principles of OOT are problematic when considering the use of OO programming languages for safety critical applications. For example, lack of predictability of OO program execution can be caused by the flexibility of the program due to polymorphism. Polymorphism is enabled due to the combination of typing and inheritance. Polymorphism is when an object of a higher class is assigned as an object of a lower class at run-time, so that it can implement different/specific functionality. Polymorphism manifests itself in the following two capabilities, both of which reduce the predictability of the run-time program:

- Dynamic binding – method execution bound to an object at run-time;
- Dynamic typing - assignment of an object to a different class at run-time.

APPENDIX B – Document Review

The tables in this section provide a summary of available documentation relating to developments in the use of OOT in general and OO programming languages, for safety critical applications. Documents are divided between those regarding general OOT and those related to specific OO programming languages such as C++ and Java. Documents are listed from most recent.

Documents – OOT

Organisation	Document Title	Year
FAA	Handbook for Object Oriented Technology in Aviation (OOTiA) (Ref 11)	2004
<p>This handbook is divided into 4 volumes: Overview, Considerations and Issues, Best Practices, and Certification Practices. The handbook states that it “does not constitute FAA policy or guidance” and that it should not be used as a stand-alone product but instead should be considered within project context.</p> <p>Volume 1 highlights the current issue for certification of OOT in airborne systems: “no universal guidelines exist for using OOT in safety-critical systems. Certification authorities have been using issue papers on a project-by-project basis to address OOT concerns. These project-specific issue papers document high-level safety issues and concerns with OOT but do not suggest or provide detailed issues or acceptable solutions.” (p. 1-1). The aim of the manual is to extend these issue papers by providing guidance for compliance with DO 178B.</p> <p>Questionnaires for assessors and designees are included in Volume 4: Certification Practices. These questions, which are to be considered in conjunction with those in the FAA Job Aid (Ref 12), are cross-referenced to DO 178B requirements.</p>		

Organisation	Document Title	Year
FAA – Certification Authorities Software Team (CAST)	Object-Oriented Technology (OOT) in Civil Aviation Projects: Certification Concerns (Ref 13)	2000
<p>Concerns are noted for the use of OOT in airborne software and the requirements of DO 178B. These include certification buy-in, traceability issues, verification of dynamic memory allocation, structural coverage and the effects of inheritance and polymorphism on this, dead code, test coverage and testability, coupling forced by inheritance, ambiguity due to inheritance, polymorphism and operator overloading, coding features such as dynamic binding and exception handling, and finally library dependence in which libraries may not have been developed for safety critical applications.</p> <p>Two issues with using C++ for safety critical systems are noted: dynamic memory management functions and memory leakage. Binkley’s paper is referenced: “Adherence to these guidelines can lead to safer and more maintainable C++ programs” (p. 5).</p>		

Documents – OO Programming Languages

Organisation	Document Title	Year
Australian RAF – Flight Lieutenant Derek Reinhardt, MSc thesis conducted with the University of York	Use of C++ Programming Language in Safety Critical Systems (Ref. 8)	2004
<p>This MSc thesis provides a discussion of the issues as regards to the use of C++ for safety critical systems. Building on previous work by the Programming Research Group (Ref. 7), a subset of the language is defined, a means of justifying the use of a programming language for a safety critical system is presented, and an analysis of tool support for C++ with an emphasis on verification in terms of safety.</p>		

Organisation	Document Title	Year
The Programming Research Group	High Integrity C++ Coding Standard Manual (Ref 7)	2004
<p>Guidance manual for developing high quality C++ code. Guiding principles are maintenance, portability, readability and safety. A subset of the programming language is proposed via a set of rules in order to “minimize problems created either by compiler diversity, different programming styles, or dangerous/confusing aspects of the language” (p. 3).</p>		

Organisation	Document Title	Year
QinetiQ	An Investigation of the Unpredictable Features of the C++ Language (Ref 6)	2004
<p>Issues regarding the use of C++ for safety related and safety critical applications are discussed and a set of rules for restricting the use of the language for these applications is presented. One of the conclusions of the report is that the dynamic predictability of the language is still not fully understood. It is noted that as C++ inherited a lot of its predictability issues from C, it is important that these aspects are addressed. The MISRA C guidelines define these issues. It is recommended that: “a public domain safe/analyzable subset of C++ should be developed to (traceably) address the identified issues. This again should aim to achieve wide consensus. (p. 11)”</p>		

Organisation	Document Title	Year
University of York	Assessment of the Java Language for Use in High Integrity Systems (Ref 14)	2003
<p>Discussion of unpredictable performance issues (such as dynamic class loading) and security issues. The report summarises an assessment of the language against the requirements of a programming language for safety related applications. Subsets of the language are also discussed.</p>		

Organisation	Document Title	Year
FAA – Certification Authorities Software Team (CAST)	Use of C++ Programming Language (Ref 5)	2002
<p>This paper describes features of C++ that can lead to unknown configurations of object code and unverified code. These features are divided between compile time (covering dead code, encapsulation, inheritance, and overloading) and runtime (covering dynamic binding and polymorphism).</p>		

Organisation	Document Title	Year
Exida (US company providing safety services across process and automotive industries)	C/C++ Coding Standard Recommendations for IEC 61508 (Ref 4)	2002
<p>This report proposes a methodology for defining a C/C++ coding standard compliant to IEC 61508 requirements. Two of the five listed IEC 61508 requirements for programming language selection are problematic for C++. These requirements are, that the language shall “be completely and unambiguously defined or restricted to unambiguously defined features” and “facilitate the detection of programming mistakes”. Secondly, justification should be supplied as to the fitness of the language, which also addresses shortcomings. Four behaviour issues for C and C++ are identified which require addressing. These are:</p> <ol style="list-style-type: none"> 1. unspecified behaviour (due to compiler freedom), 2. undefined behaviour (programmer can violate strict use of language without error generation), 3. implementation-defined behaviour (code may be non-portable between compilers), 4. locale-specific behaviour (operation that may vary with international requirements). <p>A set of principles is defined for mitigation against C and C++ weaknesses. Additionally, a set of rules is defined based on these principles that are cross referenced to SILs 2 and 3.</p>		

Organisation	Document Title	Year
Binkley (Department of Computer Science, Layola College in Maryland)	C++ In Safety Critical Systems (Ref 2)	1996
<p>This report considers C++ language features from the perspective of guidelines for use in safety-critical systems. The author states that: “adhering to these guidelines can lead to safer, more maintainable, C++ programs”.</p>		

Organisation	Document Title	Year
US Nuclear Regulation Commission	C and C++ Guidelines: Review Guidelines on Software Languages for use in Nuclear Power Plant Safety Systems (NUREG/CR-6463) (Ref 3)	1996
Report is unavailable. Electronically available from NUREG by application.		

APPENDIX C – Working Groups

Working groups cover the use of OOT for aeronautical systems in SC-205/WG-71, and FAA CAST, the automotive industry, the MISRA C working group, to working groups for Java. A brief description of each is provided in the following subsections. These have been divided between working groups studying OOT in general and those studying programming languages.

Working groups - OOT

Two working groups are of note. Both are looking at OOT for software in aviation and are the SC-205/WG-71, a combination of working groups from EUROCAE and RTCA, and the FAA CAST working group.

SC-205/WG-71

This working group is a combination of working groups:

- EUROCAE WG71 – Aeronautical Software
- RTCA SC205 Committee – Software Considerations

The purpose of this subgroup is to produce the new version of: RTCA DO 178B/ED-12B: Software Considerations in Airborne Systems and Equipment Certification (Ref 9). One reason for the upgrade of the standard is the changing nature of technology, and one of these issues is that OO techniques are becoming more commonly used.

The group met in January and April this year. There is another meeting scheduled at the EUROCONTROL headquarters in Brussels, 24 – 28th October 2005, which will be attended by Avionics Authorities, Airframers, and Avionics Software Developers.

There are two co-chairmen of the group:

- Gerard Ladiar (Airbus)
- Jim Krodel (Pratt & Whitney)

The joint group is attended by FAA CAST representatives, as well as all of the major manufacturers.

There are various subgroups within SC-205/WG-71. Of most relevance to this report is subgroup SG5, which is the OOT subgroup. A presentation at a recent SG5 working group meeting posed the following questions:

- What is the relationship of the documents (DO-178B, DO-248 C, Supplements, OOTiA Handbook)?
- What is the status of the OOTiA Handbook once a supplement is produced?

- How will supplements be called out by the Certification Authorities?

There is an interim meeting of the OOT subgroup in July in Hamburg.

FAA - CAST

In their 5-year plan (2003 – 2008), CAST (Certification Authorities Software Team) has acknowledged the need to address OOT in regards to safety and certification. One of these deliverables: the OOTiA is already available on their website (See Section 0). In October 2005, it is planned that a prototype training course (interactive video teletraining) on OOT basics will be made available. Finally, it is aimed that either an OOT Advisory Circular is developed, or guidance regarding OOT is implemented in DO-178C .

Contacts for OOT at FAA (CAST) are Barbara Lingberg (Lead), and John Lewis.

OOTiA workshop committee members (as of 2003) were (additional to those above):

- Kelly Hayhurst, NASA
- Brenda Ocker, FAA
- Gary Daugherty, Collins
- John Chilenski, Boeing
- Salah Obeid, EmbeddedPlus
- Jeff Knickerbocker, Honeywell
- Tom Rhoads, Goodrich
- Dennis Wallace, FAA
- Will Stuck, FAA
- Chuck Chilgore, FAA

Working groups – programming languages

Four language specific working groups are of note, two for Java, one for C, and one for programming languages in general.

SC-22

Subcommittee 22 from the ISO/IEC JTC1 (Joint Technical Committee – Information Technology) are involved in developing standards for programming languages and Operating System environments. They have working groups for different programming languages such as C++, Ada, Fortran, COBOL, C, and Prolog, as well as working groups for cross-language issues such as binding techniques, and internationalization. A SC-22 proposal has been recently released to prepare comparative guidance for a large number of programming languages, entitled “Guidance for Avoiding Vulnerabilities in Programming Languages”. The aim of the work will be to provide guidance that covers safety and security issues. Vulnerabilities of programming languages have been defined as covering 3 issues:

- Implementation dependencies, where the execution environment of a language effects its behaviour;
- Common-mode failures that occur across languages;
- Weaknesses in language constructs.

MISRA C working group

The MISRA C working group has developed a safe subset of the C programming language for safety critical automotive applications. Although C is not an OO programming language, the MISRA C working group has been included in this report, as members are interested in becoming involved in the development of a safe subset for C++.

Members of the MISRA C working group are:

- Gavin McCall, Visteon UK (Chairman of the Working Group)
- Andrew Burnard, Land-Rover
- Paul Burden, Programming Research
- Liz Whiting, QinetiQ Malvern Technology Centre
- Chris Tapp, Keylevel Consultants
- Mike Hennell, LDRA
- Chris Hills, iSystem Ltd
- Steve Montgomery, Ricardo Tarragon

Java working groups

A safety issue regarding the use of Java is its ability to support real-time applications. A lot of work is currently taking place to enable this facility.

Two working groups were set up in response to the National Institute for Standards and Technology (NIST) publication: “Requirements for Real-Time Extensions for the Java Platform” (Ref 15):

1. Real-Time for Java Expert Group:

Sun Microsystems and IBM formed the Real-Time for Java Expert Group to work on the first Java Specification Request (JSR-1). When IBM retreated from its involvement with real-time Java, TimeSys took over the role of specification lead.

2. J-Consortium:

The J-Consortium hosts the Real-Time Java Working Group and publishes the “Real-Time Core Extensions Specification”. Kelvin Nilsen, Chief Technology Officer at Aonix is

Technical Chair of the J-Consortium, as well as the editor of the “Real-Time Core Extensions Specification”.

In order to reduce confusion caused by the existence of two real-time specifications for Java, The Open Group began focusing on a combined standard for safety-critical Java. A Java Specification Request (JSR) was submitted for a safety critical Java specification and draft specification has been developed and work is under way to implement the Reference Implementation (RI) and technology Compatibility Kit (TCK).

The RI was kick-started by funding provided by NASA for a project lead by Kestrel Technology. The company was given one year for their first milestone to deliver at least a partial implementation of a safety-critical Java reference implementation.

Additionally, Aonix is delivering a commercial product based on the developing safety-critical Java specification (JRTK). JRTK is a real-time mission-critical subset of the Real-Time Specification for Java (RTSJ) and includes the following characteristics:

1. No garbage collection used on objects in the real-time heap.
2. A standard subset of Java libraries is restricted with each library's time and memory resources clearly defined.
3. Partitioning separates soft real-time components from hard real-time components to ensure hard real-time schedules as well as program reliability and robustness.

In parallel, the High Integrity Java (HIJA) project has started work on defining and implementing a new High-Integrity Java for future networked real-time embedded systems. The HIJA project is jointly funded by the European Community and participating companies. Running through 2006, the project is expected to contribute to ongoing standards development activities related to safety-critical Java, mission-critical Java, and ARINC-653 Part 2 Apex Extensions.

Project partners in the HIJA project are:

- The Open Group (Leader)
- aicas
- Aonix
- Bellstream
- Fiat Research Centre
- FZI
- Thales-Avionics
- Telecom Italia

- Trialog
- Universitat Karlsruhe
- Universidad Politecnica de Madrid (DIT-UPM)
- University of York