Ada Conformity Assessments: A Model for Other Programming Languages?

Michael Tonndorf
IABG
Ada Conformity Assessment Laboratory
D-85521 Ottobrunn Germany
Tel: +49 89 6088 2477
Tonndorf@iabg.de

1 ABSTRACT

This paper presents the actual status of Ada Conformity Assessments after the transition of Ada Conformity Assessments from the Ada Joint Program Office to ISO. The principles of Ada Conformity Assessments according to the ISO/IEE Final Committee Draft 18009 are summarized and the commonalties and differences to the previous practices are discussed. In the main part of the work conformity assessments for Ada C, C++, and Java are compared. It is shown that the process as practiced with Ada is unique compared to other programming languages. This can be understood by looking at the special culture of the language Ada and its validation system. Both were sponsored by one single party (the US DoD) and not the IT industry. The paper concludes with an assessment and outlook on the future development on compiler conformity assessments in general.

SIGAda'99 10/99 Redondo Beach, CA, USA

1.1 Keywords

Ada Conformity Assessments, Ada Standardization, Programming Language Validations

2 INTRODUCTION

An important requirement that accompanied the introduction of the Ada language was to develop a testsuite simultaneously with the language definition for Ada implementations. Thus an efficient and practicable procedure, impartial of manufacturer influences, was to be introduced. This allowed to test an Ada processor could be reviewed for standard conformity. This procedure was named by the DoD environment connected with this *Ada Compiler Validation* although the term validation in quality management has quite a different meaning. We shall be looking at the development of terms later in greater detail. After the closing of the AJPO October 1998 the assessment process of Ada compilers had to be re-organized. This is realized by moving Ada validation, or as it is called now *conformity assessment* under the sole roof of ISO. This leads to the question how conformity assessment is practiced for other main programming languages in use, e.g. C, C++, and Java.

The achievements until the end of the sponsorship of the US Department of Defense were described in [5]. After the closing of the AJPO in October 1998 the assessment process of Ada compilers had to be re-organized. The claim, which is linked to the distinction 'certified conforming' can at this point not be finally assessed. Naturally there is no implication with the distinction *validated* or *certified conforming* that the implementation

tested is error-free or 100% standard conforming. Also conformity assessments may not be considered solely as quality-promoting measures for compilers. Seen as a tool for software development a compiler is firstly the pre-requisite for the development of an application with special reliability requirements.

3 DEVELOPMENT OF THE ADA CERTIFICATION SYSTEM

The history of validations or conformity assessments for Ada cannot be described at this point in detail. Nevertheless, the most important stages and the process after the closing of the Ada Joint Program Office are reflected here.

ISO with its headquarters in Geneva, Switzerland, is the world's top level organization for standardization. ISO's policy is put into effect by the national member bodies. As long as the national member bodies implement the standardization procedures, ISO itself then does not directly participate in the procedures. In the case of the standardization of Ada 83 the role of the US national ISO member was carried out by NIST (the National Institute for Standards and Technology). This comprised the (further) development of the ACVC testsuite and the setting up of infrastructure for the performance of the Ada Compiler Validation (the so-called Certification Body). However NIST transferred the right to implement the Ada standardization policy to DoD's Ada Joint Program Office for practical reasons. And as an Ada Validation Facility the NIST was always in the loop.

During the prospering period of Ada 83 from 1985 to 1993 the world of validation was well ordered:

- The *Ada Joint Program Office* as sponsor and political instance,
- The *Ada Validation Organization* (AVO) as technical coordinator of the validation process and the Ada Validation Facilities;
- Five *Ada Validation Facilities* (AVFs; USA: NIST, Wright Patterson Airforce Base; UK: NCC; France: AFNOR; Germany: IABG) as testing laboratories responsible for conducting validations,
- an *ACVC team* contracted for the maintenance of the ACVC testsuite.

However with the upcoming standard Ada 9X this world started to crumble. At first the number of annually conducted validations decreased significantly. The French AFNOR and the British NCC went out of business due to a lack of validation contracts. What then occurred in USA in connection with responsibility for validation may be described as curious. First AJPO announced it would withdraw completely from Ada standardization by September 1997. Thus NIST would again be responsible in accordance with the original distribution of responsibilities. In fall 1997 however NIST announced it was giving up all direct IT standardization activities in spring 1998. This brought back AJPO with respect to IDA's Ada Validation Organization to validation business until September 1998. So after the closing of the Wright Patterson Airforce Base AVF and NIST only EDS and Germany's IABG remained operational. IABG's experiences in the first decade of *Ada Validation* is described in more detail in [5].

In parallel the trend accelerated to withdraw the Ada mandate in general. This development must however be seen as part of the general movement of the State standardization organizations (regardless of whether military or civilian) of withdrawing from direct standardization of products. This resulted in April 1997 in the Ada mandate being terminated, i.e. the general obligation to use Ada for all defense projects. Even in the case that Ada was used, a validation certificate was no longer required. In the spring of 1998, it became obvious that the responsibility for Ada validation would again be vacant from the fall of 1998. In WG9, the ISO working group responsible for Ada, the idea was developed to build up a workable infrastructure for Ada validation under ISO.

A number of groups had already announced their interest, what was missing were the organizational basics. In all procedures, reference was always made to the decision-making chain Ada Validation Facility – Ada Validation Organization – Ada Joint Program Office. However, the last two do not exist from September 1998 on. So in the course of the year 1998 a decision between two alternatives had to be made:

- No technical and organizational umbrella organization for Ada at all. This would mean that the work of the validation facilities would not be coordinated. The certificates would no longer be comparable and each AVF would follow its own policy. In particular uniform maintenance of the testsuite would no longer be guaranteed. Further consequences would be no structured *Ada certification body*; no coordination of validation laboratories; no coordinated maintenance and development of the Ada testsuite, no mutual recognition of validation certificates,
- establishing of an authority, that ensures a uniform validation process, using the same testsuite with the same procedures and rules.

In fall 1998 the Ada community made a movement for this second alternative. It was again only not clear who

would provide the funding for this, for this task utilizes on average one person for more than half a year. After several weeks of to and fro between USA and Europe, finally a solution was proposed. The Ada Resource Association accepted applications for an institution named *Ada Conformity Assessment Authority (ACAA).* This authority is to be represented by an ACAA Technical Agent. The basic funding for this is provided – and this closes the loop – by the DISA (Defense Information Systems Agency). The authority of the US DoD for information systems. Thus DoD is involved again – or rather still – in the system for Ada conformity assessments. However, it has no co-determination rights.

4 ACTUAL SITUATION FOR ADA

An ISO standard exists for all major programming languages. As this is true for Ada as well it was obvious to integrate the new Ada certification system under ISO. With its Working Group 9 (WG9) ISO was already continuously involved in the Ada standardization. So the main goal under direct ISO participation was to develop a uniform terminology, compatible with the other ISO standardized programming languages, and to otherwise continue with the proven and accepted validation process as developed under the AJPO. Since spring 1999 draft procedures exist containing requirements on the "new" validation process and certification body: *ISO/IEC Final Committee Draft 18009 Information Technology – Programming Languages – Ada: Conformity Assessment of Language Processor* [3]. This document reflects the basic requirements in the following areas:

- Ada Conformity Assessment Testsuite,
- Ada Conformity Assessment Authority,
- Ada Conformity Assessment Laboratory,
- Operating Procedures for Ada Conformity Assessments.

The possibilities for influence of ARA as association of Ada manufacturers cannot however be compared with those which were once available to the former AJPO. What is missing here is quite simply the economic clout.

The starting point for the new orientation under ISO is [3]. Minimum requirements are formulated there into the following topics:

- Ada Conformity Assessment Testsuite
- Ada Conformity Assessment Authority
- Ada Conformity Assessment Laboratory
- Operating Procedures for Ada Conformity Assessments

This standard is rather a collection of requirements than a set of rules. The requirements are then converted in subordinate standards into rules and procedural instructions. This certification system has been in place in this form since October 1998. Randy Brukardt from R.R. Software was commissioned to be the technical agent for ACAA. In addition, an ACAA Advisory Board was set up to provide advice for ACAA. Currently there are 9 members from industry, users, universities and ACA test laboratories.

All languages for which an ISO working group exists are listed in Table 2 for arrangement into the standardization of the programming languages in general. These groups are structured into the Subcommittee (SC) 22 *Programming Languages and Their Environments* and *Software System Interfaces in ISO/IEC Joint Technical Committee (JTC).*

Moreover there are working groups which are no longer active but have not been deleted from the list of SC 22 working groups.

ISO-WG	Programming Language
WG3	APL
WG4	COBOL
WG5	FORTRAN
WG9	Ada
WG11	Binding Techniques
WG13	Modula-2
WG14	C
WG15	POSIX
WG16	ISLisp
WG17	Prolog
WG19	Formal Specification Languages
WG20	Internationalization
WG21	C++
SC22/ JSG	Java Study Group

Table 1

The Ada Letters from March 1999 contain two contributions that reflect the current changes from the point of view of the US Ada Conformity Assessment Laboratory and the Ada Conformity Assessment Authority (see [4] and [5]). Both papers also provide a

mapping of terms used under DoD to terms now defined in the ISO procedures. Therefore we do not list the table here again.

This paper doesn't go into further details of the actual Ada testsuite ACATS Vers. 2.2. The testsuite and all relevant information can be retrieved from `www.adaic.org`.

5 ADA AS A MODEL FOR COMFORMITY ASSESSMENTS OF PROGRAMMING LANGUAGES

Which characteristics do we expect from a „Model for Compiler Conformity Assessments"? For that purpose we start with some basics in software quality and build a framework of terms yielding the unified terminology of ISO. Interlaced with the general situation the realization in Ada is examined.

An obvious requirement in conjunction with an international standard is to have a single worldwide certification system, in this case a certification system for the programming language Ada.

Testing object is an identified language processor. Test objective is to verify that the testing object complies with ISO standard, here ISO/IEC 8652:1995 (Ada). Test conducting requires a testing method. The testing method for Ada is the Ada Compiler Validation Capability (ACVC) which was developed in parallel with the first standardization of Ada as ANSI/Mil-Std 1815A in 1983. The ACVC is the foundation of a testing procedure that comprises a defined set of test cases. These test cases can be partitioned into two groups: negative and positive test cases. A positive test case is a program sequence that complies with the Ada standard. The tested implementation has to *process* this sequence in accordance with the standard. A negative test case is the intended violation of the standard which has to be detected by the implementation. For efficiency reasons usually a test is built by a series of related test cases which have equal or similar test objectives. Thus a test is the smallest component of the test suite, identified by a name. Furthermore part of a testing procedure are rules guiding the evaluator of a test result by grading the test result as *conforming* or *not conforming*. Obviously it's inefficient for the tester to lookup the language reference manual every time in order to grade the test result. For negative test cases these rules are source code comments at appropriate places, positive test cases report their results usually automatically, following a self-checking mechanism. This method is described in more detail in [4].

The tight interpretation of the term test procedure – as a systematic way of conducting tests - leads to the broader understanding of a test procedure as a detailed set of rules. The procedures cover the whole process of a conformity assessment including the rules how to interpret and grade test rest results.

Although it seems obvious; it is not: for a programming language there shall be only one generally accepted testsuite. Only this condition guarantees that implementations can be compared. Notwithstanding the Plum Hall testsuite (see Chapter 6) praises itself to be winner of the one and only once held contest for C-testsuites. A testsuite itself is like any other large and complex piece of software subject to quality management and configuration management. The ISO procedures define requirements for a sensible use of the testsuite. For Ada the testsuite is controlled and maintained by a single instance (the Ada Conformity Assessment Authority).

Result of a conformity assessment is always a report listing a detailed log of all test results. If all test results for the implementation comply with the grading criteria then a *Mark of Conformity* can be issued by the testing laboratory.

Conducting conformity assessments requires a certain qualification. Therefore these shall only be executed by recognized testing laboratories. The meaning of *recognized* or *accredited* has to be discussed later. [3] requires that the testing laboratory works according to the principles proposed by ISO. This means it shall be embedded in a well defined organization and operate on the basis of an approved quality manual. Finally it is required that all testing laboratories recognize themselves equally in order to provide a uniform conformity assessment process.

In addition to the executing role of a testing laboratory a technical instance is needed that performs general tasks. The technical authority has to enforce the *language policy* by taking care of the issues

- quality management and configuration management of the testsuite,
- future maintenance of the testsuite,
- accompanying individual conformity assessments:
  - approve special test modifications,
  - disputes ruling,
  - quality control of the test reports.

For Ada there was always a discussion how *general* a certificate should be, which platforms are be covered by a specific certificate. In Ada a common understanding emerged, although the opinions never converged completely. A conformity assessment is always conducted on a specific base platform. Following up the vendor then has the choice to *extend* the status *certified*

conforming to related implementation classes which were obtained by

- maintaining an implementation within certain limits,
- rehosting an implementation.

The Operating Procedures for Ada Conformity Assessments [6] provide the principles for extending the certified conforming status. Finally this infrastructure does not operate for free. The costs cannot entirely be covered by the fees for a compiler conformity assessment. An important role for conformity assessments is a sponsor representing the interests of the users. They expect to find a working certification system which is kept alive by the sponsor. However the sponsor must not use his influence to control the policy of the certification system. Since 1998 the Ada Resource Association (ARA) is sponsor of the Ada certification system.

In summary the features for a model we postulate are presented below together with the actual realization in the new Ada certification system as required by [3]:

1. worldwide unique authority controlling the evolution of the language standard (here: ISO WG9 Ada),
2. sponsor and authority for public relations (here: ARA and Ada Information Clearinghouse),
3. (technical) Conformity Assessment Authority,
4. operational Conformity Assessment Laboratories,
5. testsuite including user- and version-documentation,
6. Operating Procedures for Ada Conformity Assessments.

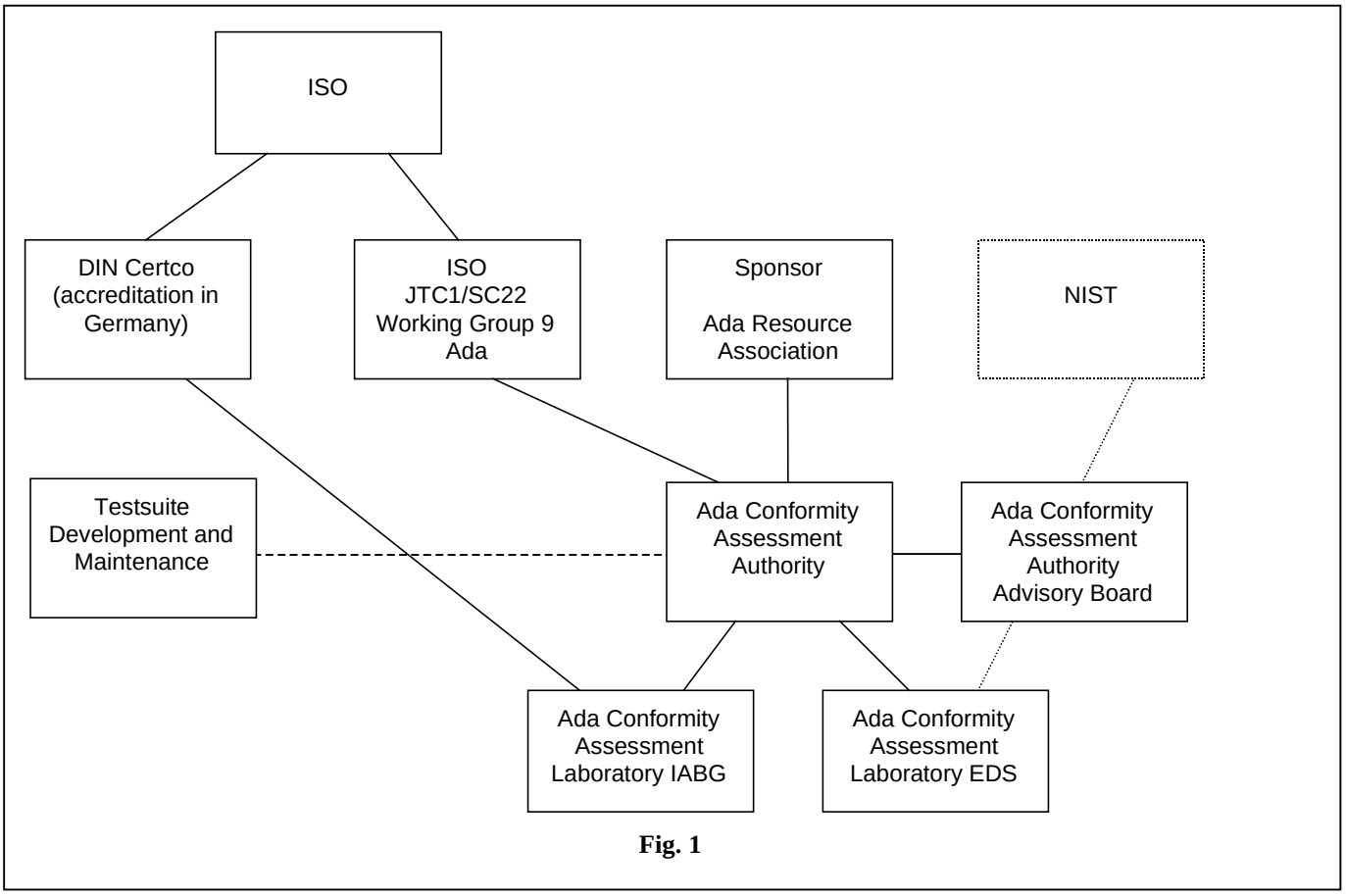

Fig. 1

Fig. 1 shows the parties involved in Ada conformity assessments and their dependencies.

6 THE SITUATION WITH OTHER PROGRAMMING LANGUAGES

A direct comparison of Ada's certification system with that one of other programming languages is difficult. This is due to the unique situation of Ada. Ada is the only language for which a certification system was built up by the sponsor and customer of that language and will

continue to be centrallistically organized in this way by ISO. Therefore Ada is the only language for which information about certified implementations is freely available. To this extent, the question posed in the title of this presentation can be answered in advance with 'yes'. However, there are restrictions concerning user-friendliness for Ada. The remainder of this paper is intended to support this premise by means of characteristic examples.

As mentioned already the NIST discontinued IT standardization, especially standardization and validation of programming languages. There are many implications of this decision: first of all there is no party entitled to accredit conformity assessment laboratories. This means that any organization or institution can declare itself as an accreditation facility and thus accredit new testing laboratories. Moreover nothing prevents an organization form declaring itself as a testing facility. The only controlling mechanism for that is the market.

Furthermore the information pool role is changing. NIST was the editor of a „validated products list" which was also available in printed form until 1995. Basis of these products are US „Federal Processing Information Standards (FIPS). After all Ada is also a FIPS Standard (FIPS Pub 119-1). From 1997 on however there are no new entries to this list. This means that the length of the list is continuously approaching zero, which is expected for the end of 1999. It is obvious that this list does not reflect the actual status of validated implementations by any means.

For a comparison of conformity assessment candidates for Ada living languages should be selected for which conformity assessments are actually being conducted. This statement is actually true for C, C++, and Java. In addition conformity assessment for the Draft Standard Embedded C++ and for the C++ standard library are emerging. On the market of compiler conformity assessments two main players can be identified: Perennial and Plum Hall. They gathered their experience with conformity assessments during the involvement of their staff in the standardization process of C, C++, and Java.

Information about the testsuites was retrieved from the internet. Below two tables with some characteristic figures are given. The metrics of the test suites (size of code, numbers of test cases etc.) are not given in a consistent manner so that a direct comparison is impossible.

The development of the testsuites for C and C++ at Perennial reflects also the development of the languages. ACVS was contracted by the US government after the standardization of C early 1990. The suite was developed further as commercial product CVSA which contains ACVC as a subset. CVSA eventually was extended from 1992 on to result in C++VS. The suites have an open architecture allowing a customer to add individual tests to the suite. All testsuites are structured according to the resp. language reference manual.

The test suites of Perennial and Plum Hall are constructed according to the same principles. Contrary to Ada however they are delivered together with script programs that allow immediate processing on standard operating systems (Unix, WinNT). Thus the user has a tool at hand allowing him to process arbitrary parts of the testsuite and to display the evaluated results in a human readable form. The scripts also support regression testing. In these aspects the commercial testsuites must be clearly attested increased user friendliness. However this also has a price, with license fees in the range of man-months. The philosophy of testing is the same for every language: using positive and negative test cases grouped together in identified test programs. It is remarkable however that Ada puts increased emphasis on negative test cases (the "B-tests" for those who are familiar with).The goal of using the testsuites is again different for Ada and C/C++/Java. For Ada the ultimate goal is to go through a formal testing procedure and obtain a certificate issued by a neutral third party testing organization. For the other languages the testsuite is primarily the basis for compiler vendors' self tests. Issuing a certificate is an exception, *branding* is offered on demand. This means that compiler validation or conformity assessment is subject to free enterprise. Except for Ada there is no direct comparability of the tested implementations:

- competing testsuites,
- no single list of tested implementations,
- no expiration of validated status,
- conformity assessment by independent testing laboratories not as the regular case.

Along the lines with the new development for Ada there is no national accreditation for any testing laboratory for C, C++, or Java. Moreover after NIST's move out of standardization there will be no national accreditation for IT testing laboratories in the US anymore. The same statement holds true for Europe except IABG's valid DIN accreditation, which was actually renewed for Ada 95 in February 1999 by Certco, DIN's subsidiary for national accreditation.

Eventually there is no link between the C/C++/Java certification systems and ISO. As a consequence there can be no recognition of any conformity assessment result by an impartial standardization organization.

It should be noted that under ISO the term *language processor* is preferred over the term *compiler*, as there are language processors which are different from a compiler.

6.1 A Brief Sketch of the Family of Plum Hall Testsuites

In the following the family of testsuites from Plum Hall shall be presented as an example for non Ada validations. Presenting Plum Hall in more detail does not express any preference or recommendation for or against any vendor of validation suites. However a detailed overview over all testsuites on the market is not the goal of this paper. The reader is invited to search the Web Sites of Plum Hall and Perennial for their latest products and announcements.

Tracking the development from C over C++ to Java is a good way to demonstrate the evolution of the testsuites in parallel. For CVS, the Plum Hall validation suite for C, the simple objectives

- checking conformance with the ANSI/ISO C Standard (the *Conform* part of the testsuite),
- compiler testing and bug finding (the *Testing* part of the testsuite),

are identified. Note that the first enumeration is an extension to the approach taken with Ada. This goal implies that exhaustive error detection is performed, e.g. a permutation of operands in expressions. Specifically within the test procedures new tests are produced dynamically, which means that a number of tests only exist in a volatile manner. This kind of test processing is not practiced for Ada. With Ada validation the objective of the ACVC was a coverage of all features of the language as defined in the Ada Reference Manual. A *repeated* testing of features by generating test code is not part of the Ada testing philosophy. At the time the ACVC was designed no one simply was willing to spend additional effort to provide such coverage testing.

The following information was compiled using actual Web pages from Plum Hall. More material on their validation testing offerings can be found there.

6.1.1 CVS Testing

The components of CVS are

- EXIN, the EXecutive INterpreter is a script language processor. When it is built and passes its own test set, the script processing is used as a basic tool in subsequent sections of the Suite.
- COVER, this section uses EXIN scripts to generate self-checking C programs that test coverage of all permutations of operators and data types. The standard scripts in the COVER section will generate almost 200 megabytes of test files.
- NCOVER, A variation of the COVER scripts. These scripts produce the same set of C language tests, but they are more compact (and less readable). This is intended for systems where the entire set of tests is generated and kept on disk (approximately 35 MB).
- LIMITS, more EXIN scripts that are used to determine the size of certain compile time limits (e.g. significant length of identifiers or how deeply include files may be nested).
- EGEN, the Expression GENerator is a test program, written in C, which generates self-checking expressions of arbitrary complexity. It is the tool used by the STRESS section.
- EGEN64, as above but with support for 64 bit integer types.
- STRESS, since it is impossible to test all possible legal expressions, a sampling approach is taken. Under the control of EXIN scripts, EGEN is used to generate complex self-checking expressions. These can be completely random, under the control of a basis expression template, or driven from an EGEN script file. (continuous runs in the background to check for compiler errors)

The test harness of CVS allows to feed in additional user-supplied tests. This is feasible for Ada ACATS testing too, but is not along the lines of Ada validation policy. In summary conformance testing is only one part of CVS validation testing.

6.1.2 CVS COVER

This part of the testsuite is used to test conformance of compilers against the ISO/ANSI C language standard. The entire suite consists of the above, plus the following which enables quality evaluation of compilers.

Test Class	Number of Tests	Target of Tests
Lang	1223	language syntax and semantics
Prec	2001	precedence of operators
Lib	8781	standard library
Exprtest	12471	expression code generation
Errtests	431	error handling
Capacity	1	translation-environment limits
Environ	20	Section 2 of the Standard
Interp91	28	C standard defect report tests

Table 2

The conformance validation section of the CVS suite COVER currently consists of the following tests listed in table 2.

6.1.3 Suite++®

Suite++ is Plum Hall's testsuite for C++. In Suite++ a *test fact* is defined as one sentence from the C++ standard, or one alternative rule of syntax, which embodies a change from C. Those sentences that imply both positive tests and negative tests are counted as two cases. By this criterion, Plum Hall estimate the total test facts in the (1995) C++ draft as 1800 positive test facts and 1192 negative test facts, for a total exceeding 2992 test facts. (These totals cover the language definition of C++, and do not include any cases for the library.) Each positive fact may embody several executable tests in its source code. There are over 4700 executable tests in **Suite++**. It also includes tests for the more difficult features like templates and exceptions.

Each test fact, as defined above, is individually designed C++ executable code. Plum Hall claim to provide a level of coverage comparable to that CVS. Each test fact is individually written: sometimes the simplest possible example and sometimes more complicated. The examples are all meant to be intuitively graspable pieces of code; enormous constructs whose correctness is beyond human parsing are avoided. Attempting to cover all possible interactions of features in every test fact would be infeasible, and each extra case would contribute little marginal utility.

For Suite++ the issue of interaction of features is identified as a problem but cannot be solved entirely, which holds true for Ada as well. The approach taken is to provide examples of all important interactions somewhere, in one or more facts.

Suite++ consists of three sections:

- Conform: positive tests coverage for C++ standard,
- Conform/Negtests: each test contains an erroneous construct that should lead to an error message from the compiler,
- Conform/Ctests: the sections testing the C-like part of C++ (CVS must be accessible for that).

The Delivery Harness of Suite++ has among others the following properties:

- Multiple Trees: support of multiple installations and products,
- Distinction between host and target: support of multiple platforms,
- Uniform batch-script conventions: these integrate the use of MAKE and the SNAP tool.

6.1.4 LibSuite++™

LibSuite++, the Plum Hall Validation Suite for the C++ Library, is a set of C++ programs for testing and evaluating a C++ library implementation. LibSuite++ covers Chapters 17 through 27 and Normative Annex D of ISO/IEC 14882:1998 (the ANSI/ISO Standard for C++) which was published by ISO and ANSI in September 1998. Each test in the suite is derived from a specific statement in the Standard.

The subdirectories of the CONFORM section provide 80 C++ programs, each covering part of a clause in the Library section of the Working Paper.

If a particular C++ compiler cannot compile a special test case, that case can be disabled using a flag. In addition DISALLOW flags can be defined to globally disable certain language features that a compiler may not be able to handle. These mechanisms were introduced to cope with the still dynamic development in the interpretation of the standard and certain target or runtime dependencies.

6.1.5 Plum Hall JVS™

Java is the youngest language among those presented here. The JVS Plum Hall validation suite for Java is still under development

- **JVS-Grinder**

  A selection of self-checking Java programs that test permutations of operators, primitive and reference data types.

- **Expresso Tests**

  A selection of machine generated Java programs of arbitrary complexity which are designed to test the expression processing capability of a Java compiler. Expressions are generated using each of the applicable operators, compiled and executed, and calculated results are compared against similarly derived results using the smallest component subexpressions.

- **Language Validation Tests**

  JVS contains a large number of hand written language conformance tests. To date the suite has tests for chapters 4, 5, 8, 9, 10, 20, 21 of *The Java Language Specification* (JLS) and the majority of the available Core library 1.1 specifications. Statements in the specifications have been translated into two categories of tests:
  - *positive tests* for valid assertions of statements, which should compile and run successfully, and

- *negative tests* for generation of diagnostics for invalid conditions, checking the compiler's capability to detect bad code.

➢ **Class Library Validation Tests**

The JVS currently has over 2,000 test programs for the original core library as described by the JLS, with the following packages upgraded to Java 1.1 level

- java.lang
- java.lang.reflect
- java.util

The java.io package currently covers the methods described in JLS. Both this and java.util .zip are work in progress.

6.2 Validation of Embedded C++

For sake of completeness it must be stated that Perennial offer testsuites for C/C++/Java as well. Whereas they do not seem to support explicit testing of the C++ library they have an Embedded C++ Validation Suite named EC++VS since August 1998, supporting tests according to the Embedded C++ Technical Committee Draft from August 1997. There are currently 22_000 test cases.

6.3 Overview of the Testssuites for C, C++, and Java

A summary of the testsuites from Plum Hall and Perennial is given in Table 3 and Table 4. The numbers given in the tables should not be taken as any quality metric of the testsuites. Up to date information can be retrieved from the Internet pages `www.peren.com` and `www.plumhall.com`.

Plum Hall		
Language / Library	**Testsuite**	**Metric**
C	CVS 9.00 1998	24_956 test cases, 56_000
C++	Suite++ Vers. XVS5	>4_700 executable tests, 1_800 positive test facts, 1_192 negative test facts
C++ standard library	LibSuite++	2_000 test cases
Java	JVS	6_200 test cases, 2.4 Mega Lines of Java source Code, 800_000 executable items

Table 3

Perennial			
Language	**Testsuite**	**Version / Release-Date**	**Number of test cases**
C	ACVS	Vers. 4.5 Jan. 98	8_000
C	CVSA	Vers. 6.7 Oct. 98	43_000
C++	C++VS	Vers. 5.1 Feb. 99	72_000
Embedded C++	EC++VS	Vers. 1.0 Aug. 98	22_000
Java	JETS	Vers. 1.1 Oct. 98	18_000

Table 4

7 SUMMARY AND ASSESSMENT

It has been shown that conformity assessments for Ada play a special role. A considerable investment by the DoD led to a single testsuite and to a working certification infrastructure. The achievements of the certification system could be preserved after DoD's withdrawal from validation. The testsuite covers the whole language and is freely available, however the development of the ACATS cannot be regarded as completed. The approach of exhaustive testing by generating new tests out of the testsuite is not pursued. Information about test implementations can be always be retrieved from a central internet site (www.adaic.org). For C, C++, and Java there are competing testsuites available as commercial products. The purpose of these testsuites is primarily self-testing for the compiler developers and vendors. Neither conformity assessments of independent testing laboratories nor certificates as evidence for successful testing are the regular case. In contrary to Ada these testsuites contain elements of performance testing of a language implementation. However this approach seems not to be neither systematic nor complete. For Ada the understanding always was to strictly separate conformance testing from performance testing.

Table 5 summarizes the main differences between the conformity assessments of Ada and C/C++/Java.

Assuming a realistic viewpoint the importance of compiler conformity assessments shows a downward tendency. Vendors use testsuites as part of their internal quality management cycles. A certificate demonstrating a successful conformity assessment does not really increase the market chances for a compiler. Seen positively compiler technology is becoming more and more mature while still being away from a satisfactory maturity level. This reduces the demand for an expensive third-party-testing procedure. The concentration process on a handful of platforms also contributes to a

Criteria	Ada	C, C++, Java (Plum Hall, Perennial)
Objective of Testing	➢ Demonstrate conformance with the ISO language standard and implicitly detect compiler errors ➢ No performance testing	➢ Demonstrate conformance with the ISO language standard and explicitly detect compiler errors ➢ Elements of performance testing
Certificates	Part of the process	Branding on demand
Third-party-testing	Regular	Exceptional (e.g.. "Perennial Conformance Test Center")
Independent testing laboratories	Yes	Identical with testsuite vendors
Accreditation	Germany: yes (IABG with DIN CERTCO), USA: no (EDS)	No national accreditation at present
List of tested implementations	Yes, available on the internet	No, competing testsuites
Testsuites price	Free	High: testsuites as trademarks
Testsuite maintenance	Independent of the customer contract	Maintenance as part of the customer contract (6/12 months)
Testsuite comfort	Only tests themselves are under configuration management	Testsuite includes execution/evaluation scripts
Regression Testing	Not directly supported	Supported by the scripts
Technical authority as a separate organization	Yes	No

Table 5

consolidation in compiler development.

In summary the thesis of this paper can be answered by ***yes*** *with restrictions*. With *Ada validation* a model was created which is leading with regard to objectivity, impartiality, and completeness. The development of the last decade has shown that this is not a general need of the compiler business outside of Ada. In this domain primarily a comfortable tool for compiler self-tests is sought. However as Ada is the first choice in the growing safety critical domain Ada conformity assessment will not loose its importance, just the opposite holds true. Moreover the use of certified tools is only one requirement within the software certification cycle for safety critical software. And the need for the propagation of vendor independent standards in the software industry is more urgent than ever.

8 REFERENCES

[1] Ada Validation := Ada Conformity Assessment. Phil Brashear, EDS Conformance Testing Center. Ada Letters Volume XIX Number 1, ACM SIGAda, March 1999.

[2] Ada 95 Conformity Assessment – Only the Names Have Changed. Randall Brukardt, Steven Deller, Joyce L. Tokar. Ada Letters Volume XIX Number 1, ACM SIGAda, March 1999.

[3] ISO/IEC Final Committee Draft 18009. Information Technology – Programming Languages – Ada: Conformity Assessment of a Language Processor, 1999.

[4] An Efficient Compiler Validation Method for Ada 9X. Michael Tonndorf, Ada-Europe '93 Conference Proceedings, Lecture Notes in Computer Science 688, Springer Berlin, Heidelberg, New York.

[5] Ten Years of Tool Based Ada Compiler Validations. An Experience Report. Michael Tonndorf, Ada-Europe '98 International Conference on Reliable Software Technologies. Conference Proceedings, Lecture Notes in Computer Science 1411, Springer Berlin, Heidelberg, New York.

[6] Operating Procedures for Ada Conformity Assessments, Version 2.0. Ada Resource Association, Fairfax VA, USA.