Chapter 3
Ada 95 Adoption: Prospects, Issues, and Actions

Ada 95 brings many opportunities to projects, including the ability to better integrate Ada into multi-language efforts; newer, more effective real-time primitives; and support for object-oriented programming to name just a few. However, adoption of any new technology raises issues which must be considered. Many of these issues are common to any programming language, while some are unique to Ada 95.

This chapter divides these issues into seven major categories, describes them, and offers techniques to resolve those issues. Each subsection in this chapter presents an issues- and answers-based forum. The seven categories are:

- Project Planning
- Tools and Environment
- Upward Compatibility
- Technology Transfer
- Software Development Methodologies
- COTS, Legacy Software, and Multi-Language Development
- The Adoption Process and Personnel

The prospects, issues, and actions related to software acquisition and contracting are found in Chapter 5.

This chapter addresses the issues for software acquisition organizations and software development groups (e.g., CDAs and SDCs) in an integrated fashion. Since this chapter does not focus on the life-cycle, the differences between the issues for contracted development and in-house development are minimal and, where present, will be addressed on a case-by-case basis.

In referring to acquisition organizations, the chapter uses the term *PM Office* to refer to the Government and the term *contractor* to refer to organizations hired to perform the work. In referring to development organizations, the chapter uses the term *developer* to refer to both the management and technical teams. When concepts apply equally to both groups, the term *contractor/developer* is used.

Since each technology prospects brings its own issues, PEOs and PMs should continue to use standard tested risk management techniques* (both risk assessment and risk mitigation) to gain an understanding of the major risks that may be associated with the adoption of a new technology such as Ada 95. *See Figure 3 for definitions.*

Figure 3: Risk Management Technique Definitions

***General Risk Analysis* Techniques**

- *Risk Identification* — Produce lists of project-specific risk items. The techniques used include: checklists, decision-driver analysis, assumption analysis, and decomposition.
- *Risk Analysis* — Assess the loss probability and loss magnitude for each risk item. The techniques used include: performance models, cost models, network analysis, statistical decision analysis, and quality-factor analysis.
- *Risk Prioritization* — Produce a ranked ordering of the risk items. The techniques used include: risk exposure, risk-reduction leverage analysis, and group consensus or Delphi.

***General Risk Mitigation* Techniques**

- *Risk Management Planning* — Create plans to address each of the identified risks. The techniques used to address risk include: information buying, risk avoidance, and risk transfer and/or risk reduction.
- *Risk Resolution* — Eliminate or resolve each risk item. Techniques to eliminate or resolve risk include: prototyping, benchmarking, simulation and data gathering, use of consultants or experienced personnel, and spiral or incremental development.
- *Risk Monitoring* — Track the adoption effort's progress toward risk resolution. Take the necessary actions to continue to make progress on the adoption and to continue to reduce risk. A top 10 risk item list is a useful technique for risk monitoring. Such a list is typically updated weekly and consists of input from both the management team and the contractor/developer.

This chapter provides PEOs and PMs with a concise set of prospects, issues, and actions that were developed using the general techniques discussed in Figure 3 adapted to Ada 95 adoption issues.

Each section in this chapter begins with a table. This table summarizes each section, including the opportunities Ada 95 brings, the major issues, and the corresponding actions needed to resolve those issues. There is *not* a one-to-one mapping of the table to the subsections that follow — although all of the information in the table is discussed in detail later.

* Basic information on Risk Analysis and Mitigation is taken from the classic article by Barry Boehm, "Software Risk Management: Principles and Practices," in *IEEE Software*, January 1991. Further details can be found in the original article.

Project Planning Impacts of the Transition to Ada 95

Some of the quantitative information in this section comes from the User/ Implementor Team Reports (see Appendix A), which published the early results of groups using the preliminary language definition and compilers. Other information in this section is derived from projections based on Ada 83 adoption and usage, as well as experiences of those adopting other languages and technologies.

Project Planning Prospects, Issues, and Actions	
This subsection considers common concerns PEOs and PMs have regarding their ability to plan project costs and schedule. These must be addressed by the PEO and PM to develop an accurate plan devoid of overruns.	
Prospects	• Ada 95 provides projects with the opportunity to positively affect their schedule and budget • Ada 95 provides projects with the opportunity to produce more within the same schedule and budget • Ada 95 provides these benefits as an *incremental* upgrade from Ada 83, causing minimal disruption to existing Ada projects
Issues	• How does one determine the cost and schedule impact of Ada 95 versus using Ada 83? • How does one determine the cost and schedule impact of Ada 95 versus using another language? • What will be the impact of Ada 95 on overall project quality? • What were the experiences of the early users of Ada 95 compilers? • Who has already begun using Ada 95? • Where are the experiences of the early adopters recorded?
Actions	• Use pilot project efforts to help calibrate cost estimation models, such as COCOMO, REVIC, SLIM and SoftCost-Ada, for a project's special needs • Minimize the number of simultaneous new technologies adopted, so that the number of variables is limited • Use the transition to Ada 83 as an approximate estimation for the transition to Ada 95 in cost and schedule planning tools • Make use of Ada 95's ability to reuse code written in Ada and other languages to increase quality • Contact early adopters to obtain first-hand experience reports

Assessment

Issue: How Does One Determine the Cost and Schedule Impact of Ada 95 Versus Using Ada 83?

The transition to Ada 95 will most likely have the same impact on project cost and schedule as the adoption of other new technologies. The changes are based on the specific conditions involved with the new technology insertion — in other words, you calibrate the model for your situation. Commonly used cost models such as Ada-COCOMO, REVIC, SLIM and SoftCost-Ada help managers estimate the impact of the adoption of new technologies. This impact is commonly represented by changing "coefficients" and "power" figures in the cost equations.

The learning curve associated with the adoption of any new technology will have an impact on project planning. In the transition to Ada 95, the learning curve is dominated by new concepts such as object-oriented programming, rather than by Ada 95 language constructs. Managers can shorten this learning curve and lessen the impact on the project budget and schedule by implementing an incremental transition to Ada 95, in which a few new Ada 95 technologies are introduced at first and others are adopted later.

An incremental transition will likely involve even fewer iterations than the number required for the initial transition to Ada 83, since Ada 95 adds only a few new large concepts and features. The downside: few benefits may be realized at first, at least until most of the Ada 95 features are in use.

Early data on Ada 95 from the reports of the User/Implementor teams tell us:

- Compilation time was faster than a corresponding Ada 83 program — this may accelerate schedules, but it is not yet possible to say by how much; and
- Code that has been re-engineered to use some new Ada 95 features (e.g., protected types) ran faster and was smaller than corresponding Ada 83 code — this may allow some projects to achieve their requirements with less risk.

PEOs and PMs should remember, however, that these were early results with partial compilers. It remains to be seen whether complete implementation of the standard language works better, the same, or not as well in these and other areas.

Issue: How Does One Determine the Cost and Schedule Impact of Ada 95 Versus Using Another Language?

The transition to Ada 95 from other languages will most likely resemble the transition from those languages to Ada 83. Therefore, PEOs and PMs should make use of Ada-based cost estimation tools such as Ada-COCOMO, REVIC, SLIM and SoftCost-Ada. In calibrating those models, the coefficients that control the insertion of new technology and the familiarity with Ada technology should be set to help project the transition from the group's previous language to Ada 95.

It should be expected that there will be a slightly higher learning curve when transitioning from the old language to Ada 95 than experienced in a transition to Ada 83, since Ada 95 contains additional technology. However, this effect can be mitigated by adopting an incremental transition strategy as advised in the *Ada 95 Transition Planning Guide*.

If a project adopts Ada 95 to exploit new technologies such as object-oriented programming, then learning to use new technologies such as object-oriented programming is a major item contributing to cost and schedule impacts. These underlying technology shifts are present whenever one changes programming languages, however. *If the incremental transition strategies mentioned above are followed, the impact of adopting Ada 95 should be predictable using historical data accumulated for groups adopting Ada 83 or other programming languages.*

Issue: What Will Be the Impact of Ada 95 on Overall Project Quality?

Ada 95 offers PMs an opportunity to increase the quality of their systems through increased reuse of Ada and COTS software. PMs have the ability to mitigate any risk of decreased quality caused Ada 95 inexperience. The results of the use of Ada 95 on project quality will directly depend on how successfully the developers exploit the new Ada 95 features. Several Ada 95 features may in fact lead to a general increase in quality:

- Ada 95's ability to make use of bindings to existing software packages in other languages,
- Enhanced program architecture support,
- Increased support for object-oriented programming, and
- New, simpler, real-time primitives.

These features will increase the amount of reusable components and COTS software that can be used on Ada 95 projects. These components will be used on more projects and exercised more than project-unique components. As a result, more errors will be found and removed from common software than are typically found in most project-unique software. Therefore, one of the chief ben-

efits of Ada 95's increased support for reuse and COTS becomes the associated increase in quality on each project. The new, simpler real-time features (e.g., protected types) will help developers produce software with fewer problems such as deadlocks and race conditions.

Building upon Early Experience with Ada 95

Issue: What Were the Experiences of the Early Users of Ada 95 Compilers?

Early users of Ada 95 features have had favorable experiences. As might be expected, those who have been waiting for this new version of the language are happy to use the features that they have requested. Preliminary users, using Ada 95 compilers under development, have reported the following: [All comments are based on anecdotes; no formal experiments have yet reported data.]

- Ada 95 software is easier to write and to modify.
- Ada 95 compilers [under development] are as fast as, or faster than, their existing Ada 83 counterparts.
- Some Ada 95 features (such as protected types) result in a noticeable increase of the application's executable speed.
- The execution speed and size costs of some features (such as the OOP features) are well within expectations.

Early users have been positive. Their single biggest complaint to date has been that not all of the features were (yet) implemented by every tool.

For more information, refer to the section called Early Results of Using Ada 95.

Issue: Who Has Already Begun Using Ada 95?

Since the inception of the language revision, groups have been providing feedback to the Language Revision Team and the rest of the Ada community in order to strengthen the emerging technology and knowledge base of the using community. These groups fall into several categories:

- *Advocates* — Information and support are available from literally thousands of experts and organizations through the Internet's World Wide Web. These resources provide ready, desktop access to consultation, vendors, reusable software, news, publications, on-line training, and community forums. Virtual links bring all this information together into one logical source — your desktop. Two very good entry points to this information are: the Association for Computing Machinery Special Interest Group on Ada [ACM SIGAda, http://www.acm.org/sig.ada/] and the Ada Information Clearing House [http://sw-eng.falls-church.va.us/AdaIC/]. See your network administrator for access to the Internet.
- *Early Adopters on Production Projects* — Projects have already begun using Ada 95. Some large projects expect to continue development for several more years; others are only a few months in duration. The most well known of these is the Army's Patriot Fire Control system, which is being rewritten in Ada 95. Three Technology Transition Partners are working with the AJPO on Ada 95 adoption projects: the Army's CASS project, a part of GCCS; the JAST Joint Avionics Support project for the Navy, Air Force, and Marines; and the Airfields Project, a part of DISA's GCCS effort.
- *Industrial and Government Pilot Projects* — Companies and parts of the Government have already begun pilot projects to help their organizations understand the issues associated with the adoption of Ada 95. NASA's Flight Data System Division is working on a pilot project to develop Ada 95 guidelines for avionics flight software and benchmark programs. The

United Kingdom's Ministry of Defence has commissioned a pilot study to re-implement Ada 83 software in Ada 95.

- *Trainers and Consultants* — For several years now, trainers and consultants have been creating courses in Ada 95, including the development of sample software using Ada 95. Some of these examples have served as the basis for testing the early partial Ada 95 compilers. These courses have been presented to many audiences. During the course presentation, instructors relay their early experiences in learning to use Ada 95.

- *Universities* — The AJPO, ARPA, and the Ada 95 Project Office have awarded grants to universities to facilitate the creation of Ada 95 courses and course material. Professors, graduate students and undergraduate students have, therefore, become some of the early users of Ada 95. Their course material includes examples of Ada 95 programs. Additionally, they have converted some university written software to Ada 95.

- *Language Evaluators* — The Ada 95 Program Office contracted with several teams to prototype compilers and to use those compilers — these were the User/Implementor teams. Three teams tested different aspects of the language in differing domains: real-time embedded systems, command and control systems, and information systems. Each team evaluated the impact of the language on existing software, on their compilers, and on the performance of the software. Many of their suggestions were incorporated into later language revisions.

Issue: Where Are the Experiences of the Early Adopters Recorded?

Several sources of information on early Ada 95 use exist, although the information is mostly anecdotal in nature; no large quantitative studies or experiments have been conducted. The User/Implementor Teams' experiences have been recorded in formal reports. Many papers on early Ada 95 experiences have been published in conference proceedings, including Tri-Ada 93, 94, and 95; the 1993, 1994, and 1995 Washington Ada Symposium; and ANCOAT 1994 and 1995. Also, articles on early Ada 95 use have appeared in periodicals such as Ada Letters. *For more information on obtaining these articles, contact the Ada Information Clearinghouse (AdaIC).*

TOOLS AND ENVIRONMENT

Early in the Ada 95 adoption effort, support tools will be one of the most crucial issures. Members of the Ada community, including the tool vendors, have learned hard lessons from their experiences with early Ada 83 compilers and support tools. In response, many vendors changed their approach to Ada 95 from their approach to Ada 83. This section describes the major issues associated with compilers and other tools and provides PEOs and PMs with techniques to both evaluate tools.

This section covers three major topics on Ada 95 tools:

- Ada 95 compiler maturity,
- Ada 95 compiler validation, and
- Other Ada 95 tools.

Tools and Environment Prospects, Issues, and Actions	
This subsection looks at the prospects and issues Ada 95 support tools bring to a project including their availability, maturity, cost and performance. PEOs and PMs must analyze these issues and take steps to mitigate them.	
Prospects	• Early use of Ada 95 tools can be used by pilot efforts to help evaluate Ada 95 and its impact • Ada 95 language features may make integration of tool environments and application programming interfaces easier • CASE tool vendors can exploit new Ada 95 features in their code generators
Issues	• What is the status of Ada 95 compilers? What is their projected schedule of availability? • How mature are the compilers for Ada 95? • How does one judge the stability and maturity of Ada 95 compilers? • What is the policy on Ada 95 compiler validation? • Under what circumstances can subsets and extensions of Ada 95 exist? • What is the status of other Ada 95 support tools? What is their projected schedule of availability? • How can the cost of tools for Ada 95 adoption be justified and financed by a project?
Actions	• Track and continue to follow the status and availability of compilers from *all* vendors • Track the DoD policy on the use of non-validated and validated compilers • Track the status of support tools for Ada 95 • Sponsor benchmarking activities to evaluate prospective compilers for use in your projected environments under lab conditions • Conduct pilot projects to evaluate the use of Ada 95 compilers under realistic circumstances for your project environments • Incrementally adopt Ada 95 and take advantage of low-cost tools and environments

Ada 95 Compiler Maturity

Issue: What Is the Status of Ada 95 Compilers? What Is Their Projected Schedule of Availability?

As of this writing [Fall 1995], partially complete Ada 95 compilers are already available. Several of them implement most of the Ada 95 language. They are quite usable in several ways:

- For use in pilot projects and early Ada 95 adoption efforts,
- As one part of an automated check for upward compatibility, and
- As training tools to help learn the language.

Many Ada compiler vendors have already released versions of their Ada 95 compilers. Many of these compilers process both Ada 83 and Ada 95 code and provide the user with a switch to choose the appropriate mode. A switch-selectable Ada 83/Ada 95 compiler will allow a project to use one tool to move back and forth from Ada 83 mode to Ada 95 mode until the project completely transitions to Ada 95.

The AdaIC tracks currently available compilers and vendors; to contact AdaIC, see Appendix A. PEOs and PMs should also directly contact vendors for up-to-date information. For more information on validation of Ada 95 compilers, see the issue later in this subsection.

By Tri-Ada 94 and the Ada 95 standardization, most Ada compiler vendors had announced their plans to produce compilers in 1995. Many had already begun to ship partial implementations of Ada 95. However, information on the status of products is subject to change. PEOs and PMs are encouraged to contact vendors directly for up-to-date information. Currently, validated compilers are

not available. Validated compilers are expected to appear in the last quarter of 1995.

For project use, compilers must be "production ready". PEOs and PMs should have their staff and contractors measure the compilers to verify their usability. The next subsection addresses this issue.

Issue: How Mature Are the Compilers for Ada 95?

Ada 95 vendors are building on 10 years of Ada 83 experience and compiler technology, not starting from scratch. Therefore, early Ada 95 compilers are likely to be significantly more mature and stable — in terms of compilation speed, compiler code quality and compiler reliability — than were early Ada 83 compilers. Just as Ada 95 is an incremental upgrade to the language, it is no surprise that Ada 95 compilers are upgrades to Ada 83 compilers. No major compiler vendors are known to be creating Ada 95 technology from scratch. All are building from a decade's worth of stable and mature technology.

Issue: How Does One Judge the Stability and Maturity of Ada Compilers?

Ada 95 compilers provide projects with new opportunities; however, PM Offices, their development staff, and/or their contractors must take steps to assess compiler maturity and usability on upcoming projects to ensure they are getting a quality compiler.

The most important techniques are:

- Benchmarking the compiler using a set of standard tests;
- Supplementing the standard benchmark tests with project-specific benchmarks reflecting areas of project concern (e.g., speed of compilation, execution speed of real-time features, size of generated code for generics, etc.); and
- Supporting a pilot project that then measures the compiler's usability under typical project conditions.

PEOs and PMs should use these techniques to determine whether it is too early, just right, or too late to adopt Ada 95. Also see Table 8 for additional information.

Many methods help measure the quality of a compiler. Most of the Ada 83 techniques will still be very useful for the evaluation of early Ada 95 compilers and may be used while these tests are being enhanced with Ada 95 specific additions. It is strongly recommended that projects benchmark and evaluate all potential compilers early. Some methods for evaluating Ada compilers may be found in the following sources:

- SEI's *Ada Adoption Handbook Volume II: Compiler Evaluation and Selection*. This guide to evaluating Ada 83 compilers is still relevant.
- The Ada Compiler Evaluation System (ACES), a comprehensive benchmark and quality testing suite available from the AdaIC.
- The PIWG Benchmarks. The Performance Issues Working Group (PIWG) of SIGAda has created a series of PIWG benchmarks to help evaluate compilers.

See Appendix A for more information.

PEOs and PMs should use these test suites to evaluate the quality of new Ada 95 systems with respect to known compiler baselines of Ada 83 and other languages.

Ada 95 Compiler Validation

Issue: What Is the Policy on Ada 95 Compiler Validation?

Ada compiler validation provides a measure of* minimal *quality. It assures the user that the compiler adheres to the standard as defined by ISO and ANSI. However, validation does* not *ensure that a compiler is "bug free". To determine "usability", Program Offices should sponsor compiler evaluation and benchmarking using suites of tests such as those described in the previous section.

In Chapter 2, the letter from the Assistant Secretary of Defense (C3I) stated that *unvalidated Ada 95 compilers may be used on projects.* However, non-R&D projects must upgrade to a validated Ada 95 compiler in time for the project's final delivery. A validated compiler is one that passes the Ada Compiler Validation Capability (ACVC) suite of tests, which is used to certify that a compiler implements the Ada language standard.

This suite is currently undergoing revision. The first Ada 95 version, ACVC Version 2.0, will be available in March 1995. The validation certificate will list the ACVC modules the compiler was validated against. This version of the ACVC will include tests covering all areas of the language arranged in a modular fashion to give compiler vendors the opportunity to implement those new features that their customer base wants first (e.g., OOP or real-time enhancements) and delay implementing less essential features. *Compilers will be validated under ACVC 2.0. This validation suite will be in use for two years.*

The second version, ACVC 2.1, will be released in 1996 and will include a full suite of tests for the Ada 95 core and all annexes. It will be put into use starting in 1997. The approach will be similar to the current ACVC, in that a compiler must pass all applicable tests to be validated. ACVC Version 2.1 will require vendors to implement the entire core language and will allow validation against each of the optional specialized needs annexes. This version's validation certificate will list the language annexes the compiler supports.

Issue: Under What Circumstances Can Subsets and Extensions of Ada 95 Exist?

PEOs and PMs must consider the portability of their Ada software. Validation of compilers increases portability and lowers risk by ensuring that compilers are measured against the same yardstick of conformity, the ACVC.

As with Ada 83, the ACVC will ensure that a *compiler* implements the language as it is described in the Ada standard. In this sense, a compiler must implement the full core language as specified in the standard. Unvalidated compilers may be a partial implementation of the language features.

Users, on the other hand, have always had the freedom to *use* only a subset of the language. This has historically been true when teaching the language to novices; they are encouraged to use only a subset of the full capabilities at first and add more concepts incrementally as they learn. Ada 95 also encourages the use of a subset of features in the Safety and Security Annex — where the use of certain features is discouraged to enhance program provability.

With regard to extensions of capabilities of the language (i.e., features outside the language definition), the standard gives limited permissions for implementations to add functionality through implementation-defined pragmas, attributes, and packages.

Other Ada 95 Tools

Support tools for Ada 95 are already available — to help programmers manage the development environment on large projects. Some of these tools are incremental upgrades to support tools that were available for Ada 83. Others are new tools for Ada 95 (e.g., class browsers, which help programmers examine the source code for object-oriented programs).

Issue: What Is the Status of Other Ada 95 Support Tools? What Is Their Projected Schedule of Availability?

Tools will be sold in many ways: some will be bundled with Ada compilers, others will be in the public domain (i.e., free), still others will be sold as third-party add-ons. Those available from compiler vendors will generally become available in increments along with the Ada 95 compilers. In many cases, third-party tools will be available sooner, because the compiler vendors will be concentrating on compilers before working on support tools. However, there will be exceptions. Table 6 gives examples of the kinds of tools and discusses their availability. The table lists general categories of tools rather than specific products. Some tools will be available from the compiler vendor, both bundled with the compiler and as extra add-on tools. Others will be sold by third-party vendors for multiple compilers and platforms.

PM Offices should have their contractors and/or development staff develop a tool integration matrix that shows the interactions between the following concepts: tool availability, platforms that the tool runs on, dependency on other tools, and support technology (and version) required (e.g., operating system and version or DBMS and version). The matrix will identify if a suite of tools must be coordinated for early handling.

Table 6: Example of Some Support Tools and Their Availability (as of Fall 1995)

Type of Tool	Source	Likely Timeframe
Ada Library Support Tools	Compiler Vendors	Immediate with Compiler
Run-Time Libraries	Compiler Vendors	Immediate with Compiler
Compilation Ordering Tools	Compiler Vendors	Immediate with Compiler
Documentation Generators	Third-Party & Software Environment Vendors	After Compilers
Test Case Generators	Third-Party Vendors	After Compilers
Quality and Style Checkers	Compiler Vendors and Third-Party Vendors	Both Immediately and Later
Class Browsers	Compiler Vendors and Third-Party Vendors	Both Immediately and Later
User Interface Builders	Third-Party Vendors	Both Immediately and Later

Check with the AdaIC for more information on these types of tools.

Issue: How Can the Cost of Tools for Ada 95 Adoption Be Justified and Financed by a Project?

Tools account for a small portion of overall life-cycle costs. Incremental adoption of Ada 95 will help ensure that the cost does not affect the project's budget in ways not accounted for. PEOs and PMs should exploit several of the creative options available to reduce the tool costs of Ada 95 adoption. Some compiler vendors are making inexpensive upgrades available (e.g., the cost is built into current maintenance agreements). Other vendors are charging prices equal to that of a new compiler. Pragmatic PEOs and PMs should re-evaluate their vendor sources and comparison shop for the best deal.

There are several *techniques* PEO and PM offices should employ to reduce the likelihood that tool costs will disrupt the project budget:

- *Conduct pilot efforts using low-cost tools and environments* — Some new sources of Ada 95 technology are free. One widely available resource is GNAT (**G**NU/**N**YU **A**da 95 **T**ranslator), a *free* compiler that is part of the Free Software Foundation software suite. Many groups are producing additional free software that fits in with and enhances the basic GNAT compiler. The availability of such free software enables PEOs and PMs to sponsor small pilot projects with minimum investment. (However, remember that free software comes without many of the things projects may want or need, such as printed documentation and hotline support.) Additionally, compiler vendors are making beta copies of their products available early and at a discount.

- *Take advantage of the upgrade and maintenance programs of Ada 83 vendors* — Some vendors are offering Ada 95 technology as part of their compiler's normal upgrade path. Projects that are currently under maintenance contracts will be able to transition to Ada 95 toolsets at a lower cost. PMs should talk to their compiler vendors to determine if their projects qualify.

- *Use PEO's cross-project view of multiple projects to negotiate quantity discounts* — Cost savings may be available when purchasing multiple copies of compilers (and other tools). PEOs, with their responsibility for multiple programs, can purchase such tools in larger quantities than the individual programs, allowing each program to benefit from the quantity discount and from making the environment both standardized and stable.

- *Work with vendors to evaluate new environments on pilot projects* — Many vendors will supply an evaluation copy of a compiler so that a project may conduct benchmarks and evaluate the product in a realistic setting. Small pilot projects to evaluate Ada 95 technology may be possible in cooperation with Ada 95 compiler vendors.

UPWARD COMPATIBILITY

Ada 95 is highly upward compatible with Ada 83; PEOs and PMs will be able to preserve their investment in Ada 83 software when transitioning to Ada 95. PEOs, PMs, and their developers and/or contractors need to be aware of certain issues associated with compatibility and act accordingly.

Upward Compatibility Prospects, Issues, and Actions	
This subsection looks at the opportunities and issues that surround the upward compatibility of existing Ada 83 software with new Ada 95 compilers. Information is given on how to assess the upward compatibility and prepare current Ada 83 projects for eventual Ada 95 migration.	
Prospects	• Ada 83 software can continue to be used — as is • Tools generating Ada 83 software can be used with Ada 95 • Ada 83 bindings to software can be used with Ada 95
Issues	• How upwardly compatible is Ada 95? • What should be done to assess the upward compatibility of existing Ada 83 software? • How do I preserve my investment in software written in Ada 83? • How do I maximize my existing investment in Ada 83 tools? • How do I preserve my investment in legacy software that was not written in Ada 83?
Actions	• Ensure that staff have fully investigated and understood the detailed technical issues of upward compatibility • Assess the degree of Ada 95 upward compatibility in existing Ada 83 software • Investigate the degree to which new Ada 95 tools will be able to switch between Ada 83 and Ada 95 modes • Continue to use non-Ada legacy software, using technology such as "wrappers" and re-engineering to ensure cost-effective use of legacy code

Assessment

Issue: How Upwardly Compatible Is Ada 95?

The brief answer to this question is — very compatible! Ada 95 is almost a strict superset of the Ada 83 language. However, it should be noted that there a few areas where upward incompatibilities cannot be avoided. For example, to add new capabilities such as object-oriented programming and improved real-time programming it was necessary to introduce new keywords into the language. These keywords will make some Ada 83 programs incompatible with Ada 95. Fortunately, it is possible to automatically detect this kind of upward incompatibility by running the source code through an Ada 95 compiler. Fixes can be made using automated tools.

Based on current analysis of Ada 83 and existing Ada 95 programs, it is expected that a majority of existing Ada 83 programs will* not *have to be significantly altered; the most common incompatibilities will be detected when the software is recompiled with an Ada 95 compiler. Based on early Ada 95 experiences, there are five upward incompatibilities that most organizations need to be aware of:

- New reserved words — Ada 95 has six new reserved words. Any Ada 83 program that uses these as identifiers is an illegal Ada 95 program. *All Ada 95 compilers will diagnose this problem*. It is straightforward to use an editor's Search and Replace function or to write a script to change all of the identifiers into ones that do not conflict with the keywords.

- Type Character has 256 values — Ada 95 will allow the use of international characters by using the ISO Latin-1 character set (a superset of the common ASCII characters). A few programs that depend on the fact that Ada 83's type character has 128 values will be affected by this change. *The ISO Working Group maintaining Ada has already allowed this change in Ada 83. An Ada 95 compiler will* not *be able to automatically detect* all *cases of this problem*. Peer inspection and automated tools should be used to help diagnose this situation.

- Generic Parameters — Ada 95 differentiates between two cases of generic instantiations; Ada 83 does not. This change will require that some Ada 95 programs add the characters"(<>)" to some code. *Ada 95 compilers will automatically detect this situation, and the change is very straightforward.* This change will detect some previously undetected errors and will increase the portability of some Ada 83 software.

- Library Package Bodies — In Ada 95, it is now illegal to supply a body to a package that *does not* require one. This change eliminates a class of subtle errors that can occur in Ada 83 where a package specification — but not the body — can be recompiled, and the error goes undetected. *This will automatically be detected by Ada 95 compilers.*

- Numeric Error — In Ada 95, this exception has been changed to be a renaming of Constraint_Error. This is consistent with several official clarifications of the standard since 1983. This will, however, create a few situations where old code must be slightly modified to ensure that both exceptions are used consistently. *An Ada 95 compiler will detect these situations automatically.*

Beyond these major upward incompatibilities there are approximately 25 "pathological cases" that, while technically upward incompatibilities, are highly unlikely to occur in actual systems. Most organizations will never encounter these and they can safely be the worry of only the chief technical leader for the development project and not the PEO or PM.

The most important upward compatibility issues have been summarized on a one-page flyer that is available from AdaIC. More comprehensive details can be found in the document Ada 95 Upward Compatibility Guide V6.0. *For more information, see* *Appendix A*.

Mitigation

Issue: What Should Be Done to Assess the Upward Compatibility of Existing Ada 83 Software?

One of the first actions that PEOs and PMs must take, even prior to first Ada 95 adoption, is to ensure that contractor/developers working on Ada 83 development activities produce software that is upwardly compatible with Ada 95. Existing Ada 83 code should also be evaluated. To accomplish this, PEOs and PMs should:

- Supplement the contractor/developer's internal peer reviews with a step to assess the upward compatibility of the project's software. (This review should consist of both manual and automated checks.)

- Supplement formal Government software reviews with a step that reviews the status of all code for Ada 95 upward compatibility.

Emphasis in these reviews should be on the cost-effective placement of all Ada 83 software currently under review into three categories. (Most Ada 83 software will fall into categories 1 or 2.)

1) *Ada 83/Ada 95 Compatible* — This software may be recompiled using an Ada 95 compiler as is.

2) *Potentially Compatible* — This software contains one or more simple incompatibilities but can be made Ada 83/Ada 95 compatible using a combination of automated and manual means.

3) *Vendor or Application Specific Implementation* — Software in this category is *not* Ada 95 compatible and it would *not* be cost effective to make this software Ada 83/Ada 95 compatible. For example, the software in question might make use of a vendor-specific run-time extension that has been replaced by a standard Ada 95 feature. Additionally, if the software

is maintained by another organization, then it may remain Ada 83-specific until its owners migrate it to Ada 95. Software in this category will typically be scheduled for future re-engineering into Ada 95. Projects may encounter similar problems when porting Ada 83 code from vendor to vendor.

Eventually, projects moving to Ada 95 will begin to encounter a fourth category of software:

4) *Ada 95 Only* — Software that makes use of new Ada 95 features (such as tagged types or hierarchical libraries). This software will not be backward compatible with older Ada 83 compilers. PEOs and PMs should be aware that creating this kind of software represents their move away from Ada 83 compilers.

Once the software under review has been categorized, the PEO and PM staff must then determine the disposition of "potentially compatible" software. If funds become available and the payback is judged sufficient, the software should be revised to make it upwardly compatible. Typically this will be done as part of the project that creates the "potentially compatible" Ada 83 software. However, if the current project is resource constrained, then the software should be scheduled for upgrade either as a separate project or as category 3, "Vendor or Application Specific Implementation".

Issue: How Do I Preserve My Investment in Software Written in Ada 83?

Ada 95 software needs no special effort to integrate, communicate with or call Ada 83 software. In most cases, by simply compiling the Ada 83 software along with new Ada 95 code, the Ada 83 code is immediately available for reuse in the new Ada 95 environment. ***PEOs and PMs should, therefore, ensure that the Ada 83 code is upwardly compatible*** *(see the previous section).* Ada 95 is *not* a new language, it is an incremental improvement on Ada 83.

Current evidence suggests that the majority of Ada 83 software will be upwardly compatible and will run unchanged when compiled by an Ada 95 compiler. Of the fraction that is incompatible, the majority can be made upwardly compatible by using simple automated tools such as text editors, shell scripts or ".com" and ".bat" files. ***PEOs and PMs should expect to continue using existing Ada 83 software both during and after their transition to Ada 95.***

Issue: How Do I Maximize My Existing Investment in Ada 83 Tools?

PEOs and PMs should expect to make use of their existing investment in Ada 83 tools. These tools will be useful in two different ways:

1. *Continue to Use Ada 83 Tools Where Appropriate* — During the process of Ada 95 adoption, PEOs and PMs must consider two kinds of software: *Ada 83/Ada 95 compatible* software and *Ada 95 only* software. *Ada 83/Ada 95 compatible* software is both upwardly and downwardly compatible and may be processed with both Ada 83 and Ada 95 tools. *Ada 95 only* software makes use of new Ada 95 features and may only be processed with Ada 95 tools. By maximizing the use of *Ada 83/Ada 95 compatible* software, PEOs and PMs can continue to make use of their existing Ada 83 tools during the transition period. As *Ada 95 only* software becomes dominant (i.e., as projects begin to make significant use of Ada 95), then projects can incrementally increase the Ada 95 toolsets and support.

2. *Upgrade to "next version" Ada 95 tools* — Since Ada 95 is an incremental upgrade and not a new language, tool vendors are building on their Ada 83 toolsets and experience. As a result, some tool vendors are offering Ada 95 capabilities in the new versions of their Ada 83 tools, making Ada 95

capabilities available under the standard maintenance license for the product. Therefore, projects that have paid for maintenance or support may be able to migrate to Ada 95 technology at a very low cost and with no loss of their investment in Ada 83 tools and environments. PEOs and PMs should contact tool vendors directly.

Issue: How Do I Preserve My Investment in Legacy Software That *Was Not* Written in Ada 83?

Ada 95 has been designed to help projects continue to get maximum use from their legacy software as they incrementally migrate to their new language, environment and/or platforms.

When transitioning from one language to another, it is necessary to preserve the organization's investment in software that was written in the previous language(s) — legacy software. *PEOs and PMs are* not *expected to convert all of their legacy software to Ada 95*, this is neither necessary nor cost effective. Developers should exploit Ada 95 features to interface their new Ada 95 software with the existing legacy code. This will permit incremental upgrades and enhancements to be written in Ada 95, thereby minimizing cost.

Ada 95 developers have several kinds of support for interfacing with legacy code:

- "Pragma Interface" has been enhanced in Ada 95 to allow developers greater control over the names generated by the programming environment.
- Ada 95 now defines standard interfaces to COBOL, C, and FORTRAN within the core standard. These packages help developers to more quickly create interfaces to language-specific code by supplying standard types and conversion routines. Compilers may provide any of the three packages. If provided, they must conform to the standards definition.
- Ada 95 also provides enhanced interfacing primitives to smooth the interface with other languages such as JOVIAL, CMS-2, and assembly languages.

PEOs and PMs should remember that it is still up to the vendors to provide support for each particular compiler's language calling conventions. To ensure that a project can interface with a specific language or compiler, project staff should check with their Ada compiler vendor.

To maximize the use of legacy code within new software development or re-engineering efforts, projects should:

- Examine the legacy software to determine the areas of maximum stability.
- Make these areas available to the new system through "**wrappers**". A wrapper is a layer of software (a binding) that allows the new Ada 95 code to call on the old software (regardless of the language it was written in or the platform on which it runs). This new software layer will allow the project to make use of the legacy system until it is replaced.
- When it is cost effective, replace the old legacy code with new software; the wrapper layer serves to insulate the new Ada 95 code and ensure stability. The old legacy code can be replaced with new Ada 95 code with little or no disruption.

Ada 95, with its language interfacing features, is ideally positioned as a technology to help preserve a group's investment in legacy software.

For more information see the section on COTS, Legacy Software, and Multi-Language Software Development.

TECHNOLOGY TRANSFER

To ensure that their staff and any contractor(s) are adequately trained in Ada 95 and related technologies, PEOs and PMs should:

- Assess the need for training and budget the cost of training either before or as a part of their first Ada 95 project.
- Ensure that the technology transfer is well integrated with prototypes and pilot project efforts to allow trained personnel to practice what they have learned.
- Seed participants of the first training efforts onto other projects to help train and mentor others.
- Incorporate the results of prototype and pilot projects back into the training efforts.

Technology Transfer
Prospects, Issues, and Actions

PEOs and PMs must ensure training for both their staff and any contractor's. This means that they must understand and manage the technology transfer process. This subsection deals with Ada 95 technology transfer issues.

Prospects	• Ada 95 training can be done incrementally • Amount of training to go from Ada 83 to Ada 95 is not a large investment
Issues	• What is the impact of Ada 95 education and training on the project? • What is the availability of Ada 95 training? What options are available to acquire an Ada 95-trained workforce? • What is the cost of Ada 95 training? How does it compare to the cost of training people in Ada 83 or in other languages?
Actions	• Ensure that PEO and PM staff are adequately trained early (and then retained) • Make sure that contractors either are already trained or have training provided • Acquire and disseminate technology transfer materials to staff and contractors • Seed trained people onto pilot projects • Seed pilot project personnel onto full projects

Assessment and Mitigation

Issue: What Is the Impact of Ada 95 Education and Training on the Project?

Technology transfer, which includes both education and training, is a vital link in the Ada 95 adoption process. The early availability of Ada 95 training helps. The PEO, PM, and the contractor/developer become aware of management and technical concepts associated with Ada 95.

Education and training can take many forms, including formal courses, computer and video training, and books and articles. *Acquiring and distributing technology transfer materials will be a key part of a manager's Ada 95 management strategy.*

To provide training, managers must ensure that:

1. Training is available from suppliers, and
2. Training is made available within each PEO's and PM's organization.

To deal with the latter, managers must budget early for Ada 95 training. The former will not be a large issue, because courses are already available from many sources (*see the next subsection*).

Ada 95 training differs from Ada 83 training. Ada 83 training was focused on learning the entire language, and courses were usually one to four weeks long. Ada 95 training focuses on incremental approaches to the language. Typically, so far, vendors are offering more specialized courses that are tailored to the audience's specific needs. Ada 83 prerequisites are, of course, essential to these shorter, specialized courses. Training for non-Ada programmers generally stresses a direct move to Ada 95 — teaching an integrated set of features both "old and new".

Issue: What Is the Availability of Ada 95 Training? What Options Are Available to Acquire an Ada 95-Trained Workforce?

Ada 95 training is available. Sources include:

- *DOD Ada Trainers and Defense Colleges* — Many organizations within the DOD provide Ada 83 training to their service's students and to other DOD organizations and are upgrading their courses to Ada 95. These include the Air Force's Ada training at Keesler AFB and the service academies, such as the Air Force Academy in Colorado Springs, among others.
- *Local Colleges and Universities* — Nearly 300 colleges and universities in the United States teach Ada at some point in their undergraduate curricula. In addition to producing graduates who are already proficient in Ada, many of these schools offer courses that can help train current workers in Ada. For the past three years, the Ada 95 Project Office has been working with these universities to produce Ada 95 curricula for colleges, ensuring the early availability of Ada 95 training.
- *Tutorials at Conferences* — Often the best source for introductory training. The first Ada 95 training was presented at the many Ada-specific and general software-oriented conferences. The tutorial reprints are typically available for a fee through conference organizers.
- *Industry Training Organizations and Consultants* — A number of organizations provide commercial training courses in Ada 95 and other software engineering topics. Among the most experienced are those that have been providing Ada 83 instruction over the past decade.
- *Basic Training Materials on the Internet* — One new and very positive aspect of the Ada 95 effort is the large amount of free, publicly available material stored on the Internet. Already several kinds of Ada 95 training material have been created and made available. These materials can form the nucleus for an in-house training group's efforts. Much of this same material has been archived on CD-ROM by the AdaIC.

The AdaIC maintains detailed information on specific training dates, course names and descriptions, and contact information. Also, Appendix A provides additional information on training sources.

PEOs and PMs have several ways to acquire an Ada 95-trained workforce:

- When acquiring software via a contractor, choose a contractor who has already adopted Ada 95 (no need to train);
- Acquire training from an outside vendor for the development staff or contractor just in time to support the project;
- Develop and present training using an in-house staff (only cost effective for large PEOs or projects);
- Train staff using self-paced training (this is usually a higher risk unless it is used to supplement classroom training only); and
- Provide mentors to work along with the project staff.

It is essential that PEOs and PMs assess technology insertion and technology transfer as part their initial examination of Ada 95 adoption. If at all possible, the PEO and PM need to look for and make use of a contractor with previous Ada 95 experience. However, at the beginning of the Ada 95 adoption effort this may be difficult. Therefore, it should be expected that both the Government and the contractor will need to acquire Ada 95 training as a part of the adoption process — and should do so jointly if possible. The PEO and PM Offices should plan and budget for Ada 95 training as part of their adoption process. PEOs and PMs should remember that Ada 83 experience is available and may be almost all that is needed.

Issue: What Is the Cost of Ada 95 Training? How Does It Compare to the Cost of Training People in Ada 83 or in Other Languages?

PEOs and PMs moving from other languages to Ada 95 will find that the cost of Ada 95 technology transfer will be approximately the same as any other language. Based on informal comparisons of commercially available courses, the cost of Ada 95 training is similar to that of Ada 83 training and that of other programming languages. One *very* positive note is the early availability of free training materials that organizations can use to augment their in-house training curriculum — typically a very cost-effective approach for training large numbers of developers. Two other factors influence the total cost of training:

- *The cost of each class* — This cost seems to be well within the typical range of prices charged for training in Ada 83 or languages such as C++ or Smalltalk.
- *The total number of classes needed* — Based on the currently available courses, Ada 95 training is more extensive than that needed for a language such as assembly or FORTRAN, but composed of the same number or fewer training days than a language such as C++ requires.

***PEOs and PMs making the incremental upgrade from Ada 83 will find that their cost is even* lower.** PEOs and PMs need only provide supplemental training to make Ada 83 programmers proficient in Ada 95.

It is not necessary to get training in object-oriented technology or object-oriented programming to move to Ada 95. *Not* conducting OOT training may lower the costs of the transition and the risks associated with a multiple technology transition. The project may simply use Ada 95 as a slightly better Ada 83. However, to unleash the full power of Ada 95, projects may want to make use of the object-oriented programming technology available with the new upgraded language.

SOFTWARE DEVELOPMENT METHODOLOGIES

Ada 83 has always supported software development using object-oriented analysis (OOA) and object-oriented design (OOD) methods. In addition, many projects have used Ada 83 in combination with functional decomposition methods (e.g., Structured Analysis and Structured Design or Information Engineering). Ada 95 adds new support for **object-oriented programming,** further encouraging the use of object-oriented methods.

Software Development Methodologies
Prospects, Issues, and Actions

PEOs and PMs must be concerned not only about the programming language that the contractor uses on the project, but about the software development method used to support development. This subsection details the prospects, issues and actions associated with software methods and Ada 95

Prospects

- Ada 95 supports functional decomposition and object-oriented methods
- Ada 95 can more easily map to the notations and methods found in newer object-oriented CASE tools and methods

Issues

- What issues are introduced by using various software development methods with Ada 95?
- What should be done when simultaneously changing both language and development method on a new project?
- How do CASE Tools support Ada 95?

Actions

- Examine the bidder's proposal for compatibility between methodology choice and Ada 95 adoption
- Ensure that multiple software development methods are *not* being used on the project
- Begin a dialogue with CASE tool vendors to ensure that the tools support Ada 95 in the manner and timeframe appropriate to the project's needs
- Ensure that both staff and contractors are properly trained in a software development method appropriate to Ada 95
- Hire a consultant to mentor the project on software development methods if they are being used for the first time
- Change either the language or the method, but not both simultaneously
- Offer incentives to the contractor to change both the method and the programming language
- Employ a SETA contractor to monitor technical issues during development

Ada 95 and Software Development Methods

Issue: What Issues Are Introduced by Using Various Software Development Methods with Ada 95?

As an important early assessment step, PEO and PM Offices should require an assessment of the impact of using various software development methods in combination with Ada 95 on each project. The use of object-oriented methods provides technical advantages when used with an OOPL such as Ada 95. However, if the contractor is not experienced in object-oriented technology, then its use raises certain issues which must be considered.

When a group is unfamiliar with one or more technologies, such as Ada 95 and OOA, then trying to apply both simultaneously on a project is more difficult. Therefore, *it is very important that the contractor be fully trained in the selected methodology and have a well-defined process and tools to support the methodology.* With Ada 95, several options might be considered for a project:

1. *The Use of Object-Oriented Methods* — A very positive selection for an Ada 95 project. There is a distinct trend toward using object-oriented technology (OOT), and Ada has had a 10-year history of successful OOT use. Ada 95, with its support for OOP, will make this an even more positive selection. However, a set of issues accompanies this choice. An informal effort with no well-defined method is a high-risk solution, even though it is object oriented. The use of objects simply as an acknowledgment of a "trend" is not likely to lead to well-engineered software. In addition, further risk can be experienced if the OO method is not tailored to Ada, because the details mean different things in different languages although basic OOP concepts are the same.

2. *The Use of Functional Decomposition Methods* — This can be done in Ada 95, although it is likely that the full power of the language will *not* be exploited. However, many projects have used methods such as Structured Analysis and Structured Design or Information Engineering with Ada 83, and projects may continue to use these methods equally well with Ada 95.

3. *The Use of Mixed Methods* — Historically, attempting to use two diverse approaches has proved difficult for projects. The PEO and PM must look very carefully at any project that suggests such technology for Ada 95 use.

Issue: What Should Be Done When Simultaneously Changing Both Language and Development Method on a New Project?

Simultaneously changing two technologies on a project is not only a common event, but is often necessary for the new language's benefits to emerge. Management must take active steps to counter the implications of simultaneous introduction of two technologies. These steps fall into four categories:

1. *Information Buying Actions:* PEO and PM staff and the contractor/developer should procure training that emphasizes not only the methodology and language, but also the interactions between them. PEOs and PMs should then hire an experienced consultant to help mentor the development team. The consultant should focus on the effect that the changes will have on other software development processes and products such as documentation or testing. It is especially important that the consultant focus on those areas where the language and the methodology suggest conflicting changes. The PM staff should make use of the Internet, which provides mechanisms such as electronic mailing lists, retrieval of files via ftp and gopher, newsgroups and discussion forums such as comp.lang.ada, and information browsers such as World Wide Web (WWW) and Mosaic. The Program Office and contractor/developer will find Internet a cost-effective source of information.

2. *Complexity Avoidance Actions:* To avoid complexity, change only a single variable (either method or language) and keep the other constant. Since this may not be practical and may, in fact, eliminate other benefits of Ada 95 adoption, a good alternative is to initiate pilot projects that explore the impact of one of the two changes. These pilot projects will provide information about the adoption of each new technology in "relative" isolation. Then, the technologies may be combined for the development effort. This will minimize the complexity, even if it does not entirely avoid it.

3. *Complexity Transfer Actions:* Program Managers may transfer the transition complexity to the contractor by offering certain incentives in compensation or training for the increased complexity.

4 *Complexity Reduction Actions:* PEOs and PMs may reduce technical complexity (from changing both method and language) through the use of

additional experts. An Independent Verification and Validation (IV&V) organization may be used to evaluate the project's progress during development.

Tool Support for Methods

In the 1990s, the use of Computer Aided Software Engineering (CASE) tools on projects has become more common. Recently the DOD has sponsored a CASE technology adoption project called I-CASE. However, CASE has been recognized as a supporting technology and not as a silver bullet.

PEOs and PMs should ensure that the tools contractors use on the project support the software engineering methods and training that the project has received. PM Offices and contractors/developers must work with CASE vendors to ensure that Ada 95-support versions of their tool will be available in concert with project schedules.

Issue: How Do CASE Tools Support Ada 95?

As of this writing [Fall 1995] CASE tool support for Ada 95 has been announced. Upper CASE tools (which cover the early phases of the software life-cycle) are typically not language-dependent. Lower CASE tools (which cover the later phases of the software life-cycle) are language-dependent and need to be upgraded to support Ada 95 effectively. Many of these kinds of CASE tools will either be bundled with the compilers or sold as a part of vendor-sponsored Software Engineering Environments (SEEs). Of these vendor-supplied tools, some (e.g., class browsers) are already available for Ada 95. Third-party tools are also becoming available. Table 6 (*in the Tools and Environments subsection*) shows the projected availability for some kinds of tools. There are many CASE tools that work with Ada 83. These tools can continue to be used with Ada 95; they simply won't take advantage of the new features of the language. Several Lower CASE tool vendors have announced plans to have their tools generate Ada 95 code from diagrams.

PEOs and PMs should talk to the vendors of CASE tools — to tell the vendors the kinds of applications and platforms that they are using and provide a schedule indicating when they will be needing these tools to support Ada 95. The sooner the vendors hear from their customers, the sooner they will be able to determine which platforms and tools should be upgraded first. By taking a proactive approach, PEOs and PMs help to ensure that a tool will be in place when needed.

COTS, Legacy Software, and Multi-Language Software Development

Ada 95 will allow PEOs and PMs to incorporate existing software of many kinds: reusable software written in both Ada and other languages, COTS and GOTS software written in many languages, and a project's existing legacy code. This will increase quality and reliability and decrease project cost.

Of the many improvements that Ada 95 offers over Ada 83, Ada 95's ability to work more easily with other programming languages is possibly the most beneficial on a daily basis. Software development efforts in the next few years will continue to demonstrate the recent trend toward the integration and interfacing of multiple kinds of software, including legacy code from previous development efforts, bindings to new COTS software, and the incorporation of reusable software components. Invariably, this software will be written in multiple languages, causing the modern developer to rely heavily on Ada's ability to interface with other languages.

COTS, Legacy and Multi-Language Prospects, Issues and Actions	
PEOs and PMs can use the Ada 95 capability to interface with software written in other languages. This will allow projects to take advantage of a variety of software that is available including legacy code.	
Prospects	• Ada 95 can more effectively support interfacing to legacy and COTS software than did Ada 83 • Ada 95 can be called by other languages and act as custom code for GUI builders or 4GLs • Ada 95 can more effectively support windowing technology and other common "mixed-language" tasks • Ability to support other project goals such as "open systems"
Issues	• What bindings are available for Ada 95 development? • How easy is it to interface Ada 95 with other languages? • How easy is it to interface other languages with Ada 95? • How can Ada 95 be used in an "open systems" development effort?
Actions	• Identify all applicable standards, COTS and legacy software • Make use of existing legacy code within new Ada 95 code, if applicable • Based on knowledge of the current status of Ada 95 bindings, initiate an effort to determine the project's bindings needs and risk areas • On a systems upgrade, with legacy code involved, make sure to have the contractor investigate current wrapper technology to support the continued use of the legacy code during migration of the system • Ensure that the project's architecture is mapped to an "open systems" standard such as the NIST APP or DISA's TAFIM

COTS Bindings

Issue: What Bindings Are Available for Ada 95 Development?

PEOs and PMs should initiate an effort to track their project's current support services (e.g., database, GUI or operating system) and the bindings that these services require. This information and the bindings should be supplied to the contractor as a part of the desired development environment.

Table 7 provides a list of Ada 95 binding development efforts underway or completed.

Since Ada 95 provides additional facilities that make bindings simpler and easier to build, a large effort is under way to upgrade existing Ada 83 bindings and create new ones in Ada 95. Some of these efforts have been directly sponsored by the Ada 95 Project Office as a part of their language update. Other efforts are the work of compiler vendors, third-party vendors, or user groups. Many of these bindings will be available with Ada 95 compilers, while others will be produced and sold or given away shortly thereafter. To gain these reuse benefits, most projects will want to be users of these bindings, since significant investment may be required in the creation effort.

The use of such common Ada 95 bindings will ensure access to large amounts of COTS software. Reuse of COTS software increases the software's portability and reliability, creating a positive net effect on the project's cost and schedule.

The Ada Information Clearinghouse publishes a report on available Ada bindings and regularly updates its database of such products. ACM SIGAda's Ada Bindings Working Group (ABWG) is the primary source for technical interchange on bindings issues. For more information, see Appendix A.

Table 7: Current Ada 95 Binding Development Efforts

BINDING TO:	PLATFORM
Flex LM	Workstation
Fresco	Workstation
ODBC 2.0	Workstation/PC
Xlib	Workstation/PC
ASIS 95	Workstation/PC
IEEE 488 GPIB	PC
IEEE 1003.1g (protocol-independent communication — sockets)	Workstation/PC
IEEE 1003.1b-1993 and 1003.1c-1994 (real-time extensions and threads)	Workstation
IEEE 1295.1 (Motif) draft	Workstation
Mail Application Programming Interface (MAPI)	PC
OLE-2	PC
Win32	PC
Winsock	PC
OS2 Presentation Manager	PC
MACH kernel API	Workstation
DWARF API	Workstation
Persistent Object Base	Workstation
Xt, Xlib (X11R6)	Workstation
Common Gateway Interface (CGI)	Workstation/PC
Catalogue of Interface Features and Options for Ada Run-time Environments (CIFO)	Run-time Specific
OMG CORBA	Workstation/PC
SQL2	Workstation/PC
OSF DCE	Workstation
ODMG-93	Workstation/PC
Distributed Interactive Simulation (DIS) Layer	Workstation
Ada Bind Gen (Tool-Ada 95 to C/C++ Binding Generator)	Workstation

Multi-Language Software Development

PEOs and PMs should ensure that their development staff exploit Ada 95's improved abilities to interface with software written in multiple languages. This will allow for the reuse of existing legacy software and COTS software — thereby reducing project cost and accelerating the schedule.

Issue: How Easy Is It to Interface Ada 95 with Other Languages?

Among the technological issues that must be considered in supporting multiple language development is getting two specific implementations of two different languages to talk together, and preventing the loss of portability that accompanies most solutions. Ada 95 facilitates interoperability by defining standard interfaces to C, FORTRAN and COBOL within the language and by providing enhanced primitives and support packages to smooth interfacing with other languages. These newly redefined primitives capture 10 years of experience in mixed language development and standardize the best practices of the Ada industry.

Issue: How Easy Is It to Interface Other Languages with Ada 95?

Ada 95 directly supports the use of Ada routines as supporting elements that can be called by other programming languages. It is now possible to write single Ada 95 subprograms that may be called from other languages.

This feature is used to provide specific extended functionality to COTS products. Ada 83 was easily used to glue together other COTS products and to call them. Ada 95 can now also be used to create the supporting routines that tailor an application framework. This fits easily into the model defined by 4GLs or by languages such as Visual Basic.

Issue: How Can Ada 95 Be Used in an "Open Systems" Development Effort?

For many PEOs and PMs, portability, vendor independence, and interoperability goals will be achieved through the use of "open systems" standards such as the Defense Information Systems Agency (DISA) Technical Architecture For Information Management (TAFIM).

Ada 95 provides excellent technical support for open systems. The enhanced Ada 95 support for making use of bindings to other language Application Programming Interfaces (APIs), and the existing strong support in Ada for separating the interface to services from their implementation, contribute to Ada's ability to support the open systems vision. Ada 95 will make it easy for projects that wish to support "open systems" to do so.

To effectively use Ada 95 in an "open systems" effort, PM Offices should determine the project's bindings needs. These needed bindings should be mapped to the contents of the TAFIM and to the project's use of open systems services.

The term "open systems" has become one of the more popular buzzwords of the 1990s, and it has come to mean many things to many people. In this context, it is used to indicate the use, by software developers, of accepted, standard interfaces to software services (e.g., GUIs, operating systems and databases). This use helps to ensure that the project software is portable and modifiable, because it is built using standard services. The underlying services may be procured from more than one vendor, each of whom meets the interface; may be changed without disrupting the software project; and allow for the separation of project-specific software from general support services.

Each set of system services (e.g., GUI or operating system) is packaged together. The project's software accesses this packaged set of services through an API. If the set of services is written in a different programming language, then an *Ada binding* to the API is needed to access the services.

For more information about preserving one's investment in legacy software, see the section on Upward Compatibility.

The Adoption Process and Personnel

In addition to the new opportunities presented by Ada 95, *any* technology adoption effort has associated issues. To address the issues, the PEOs, PMs and their staffs must understand them. This section describes the issues surrounding the adoption process and personnel areas — and discusses strategies to handle them.

Adoption Process and Personnel Prospects, Issues, and Actions

This subsection looks at the opportunities and issues that should be a PEO or PM's top priorities — the common technology adoption issues that can sabotage the entire transition effort. The PEO or PM must address these issues during the initial transition planning and adoption decision process.

Prospects

- Ada 95 can help an organization to introduce *other* modern software engineering technologies
- Introduction of Ada 95 may help engineers learn more about other technology areas

Issues

- What are the necessary steps to assess the impact of the new technology?
- What are the necessary steps to minimize the possibility of adopting Ada 95 technology and tools before their readiness?
- What steps are necessary to avoid adopting the technology (and tools) too late to enjoy a competitive advantage?
- What can be done to avoid unanticipated social issues arising from the transition?
- How best can the issues of paradigm and mindset shifts from one language (culture) to another be managed?
- How best can the issues of secondary technology be managed?
- What are the best sources of first-hand information on Ada 95?
- Who can I contact with questions, comments and lessons learned?

Actions

- Use risk management technology early
- Gain awareness of other Ada 95 adoption efforts through information buying
- Invest in training and consulting and then seed the early adopters onto second-generation projects
- Involve staff, contractors and support contractors together in the adoption
- Benchmark and evaluate the tools to assess readiness
- Employ incremental transition strategies
- Ensure that the project's success doesn't rest on the use of the new technology
- Use the new technology to enhance your organization's competitiveness
- Amortize the risk of adoption over several programs within a PEO

The major issues associated with successful adoption of Ada 95 are common to most new technology adoptions, especially when moving from one programming language to another. This subsection looks at those technology adoption issues that can sabotage the entire transition effort and therefore should be a PEO's and a PM's priority. ***Successfully addressing these issues reduces the likelihood of project failure; not attending to these critical issues can impair the entire transition effort.***

Transitioning from Ada 83 to Ada 95

Most of the issues associated with transitioning from one programming language to another are simplified or eliminated, because Ada 95 is an incremental upgrade to Ada 83. Among the remaining issues that the PEO and PM must manage, the ones dealing with the adoption process and personnel are:

1. Failing to fully assess the impact of the new technology,
2. Adopting the technology (and tools) *prior* to their readiness,
3. Adopting the technology (and tools) too late and missing out on their competitive advantage, and
4. Unanticipated social and environmental issues from the transition.

Issue: What Are the Necessary Steps to Assess the Impact of the New Technology?

Preparing an* Ada 95 adoption impact analysis *is the first order of business when planning to move to Ada 95. By failing to fully assess the impact of the new technology, the PEO or PM may underestimate the nature of the transition and assume that the new technology can be incorporated with *no* impact to budget or schedule. If this happens, there is significant risk of the project coming in late and/or over budget. Although it *is* possible to transition to a new technology (such as Ada 95) and maintain budget and schedule, this can only happen if the issues are identified early and managed. PEOs and PMs must remember that changing programming languages will affect the design, test, integration and support phases as well as the coding phase of the software life-cycle. The conversion can be smooth, but only when it is an informed adoption.

The *impact analysis should include* the following project- and organization-specific information:

- What parts of the project's development process are affected when moving from Ada 83 to Ada 95?
- What is the effect on the project's development environment?
- What impact will the learning curve and training needs have on the project schedule?
- What is the effect of Ada 95 on project planning?

Information buying — the acquisition of information about the issue in order to be able to plan better — is the primary management technique if this situation occurs. By becoming aware of others' efforts to adopt Ada 95 and their results, PEOs and PMs will be able to capitalize on the successess and avoid making the mistakes that the early Ada pioneers have made. PEOs and PMs should take the following steps:

- Invest in training and consulting,
- Use local Ada meetings, conferences, magazines and the Internet to learn about others' Ada 95 experiences,
- Hire Ada-experienced people, and
- Seed the experienced people on transition projects.

Issue: What Are the Necessary Steps to Minimize the Possibility of Adopting Ada 95 Technology and Tools Before Their Readiness?

When adopted too early, the technology may fail to live up to its advertised promises. PEOs and PMs should have their staff, Research, Development and Engineering Center (RDEC), or contractors assess the technology's readiness to control the timing of the Ada 95 adoption. PEOs and PMs should:

- Sponsor pilot project efforts and prototypes to evaluate the new technology and tools in a *realistic* usage scenario.
- Ensure that the PM Office is highly informed about the current state of Ada 95 technology. To manage an Ada 95 adoption effort well, the PM office should become highly involved in the Ada 95 community.
- Understand the impact of Ada 95 on their software engineering environment. Other tools may need to be upgraded in concert with the adoption of Ada 95. The lead-time to coordinate with vendors' schedules must be accounted for in the project's schedule.
- Buffer their projects from changes in tools and technology by acquiring tools incrementally.

These pilot efforts and technology studies will reveal many characteristics of the technology. Based on these characteristics, PEOs and PMs can judge if it is too early or too late to begin the technology adoption. Table 8 provides PEOs and PMs with information on which characteristics indicate that it is the *right time for* ***their*** *organization* to adopt Ada 95.

Table 8: Balancing the Issues of Adopting Ada 95 Too Early and Too Late

Too Early	Right Time	Too Late
Staff is uninformed	Staff is informed, eager and ready to adopt	Staff has changed jobs to adopt Ada 95
Unable to estimate impact on schedule and budget	Impact on project planning is understood	Other projects are already using Ada 95 to compress their schedule and budget
Tools not yet evaluated	Tools are available and mature (i.e., "production ready")	Funds being spent on support for older versions of tools rather than upgrades
Ada 83 software not yet evaluated for upward compatibility	Ada 83 software made upward compatible	Money spent on writing "work arounds" for features Ada 95 has
No training or mentoring	Initial training/mentoring available and acquired	Most other projects have already been trained
Software methodology use is inconsistent	Impact of changing methods assessed	Productivity loss from use of older software development methods
Have not determined how strict the project deadlines are	***At the start of a project or major support enhancement***	In the middle of an existing development project with strict deadlines

The key to minimizing the possibility of adopting too early is to use information buying techniques. PEOs and PMs should use the following techniques:

- Benchmarking and quality evaluation of tools to assess the readiness of the technology and tools;
- Measuring the upward compatibility of existing Ada 83 software to become informed about the impact of Ada 95 on existing code;
- Ensuring that the project's success does not depend on the new technology being adopted (i.e., the project can still be accomplished with the previous technology, even though the old technology may increase budget or schedule);
- Working to measure and monitor pilot projects so that project estimation tools can be refined to reflect early Ada 95 experience;
- Coordinating the impact of upgrading the language technology with the support tools found in the software development environment so that the entire set continues to work together; and
- Incrementally adopting the tools and technology across parts of the project. This will ensure that problems are dealt with locally so that they don't ripple across the entire project

Issue: What Steps Are Necessary to Avoid Adopting the Technology (and Tools) Too Late to Enjoy a Competitive Advantage?

Failing to adopt competitive technologies in a timely manner may be riskier than switching to the new technology. In an era of migration to single common systems, downsizing defense budgets, and fee-for-service programs, the defense industry is becoming increasingly competitive. It is no longer enough for a project to come in on time and on budget. The program may find itself competing for funding against similar systems in other departments as the DOD migrates toward fewer standard systems. Technology innovations are spurred on by the constant need for people to do more with fewer resources. If PEOs or PMs fail to keep up with technological advances, they risk being unable to perform their mission. This may lead to other groups being chosen to provide the same services — possibly faster or on a smaller budget. It is important for organizations competing for limited resources to use new technology to enhance their competitive edge.

Timing is essential to determine when the technology is mature enough for use, but is not yet in use by all competitors (maximizing the competitive gain from the technology). One of the more suitable management techniques consists of a series of studies that helps to determine the intersection point of these two technology curves. Based on the characteristics defined in Table 8, PEOs and PMs, in cooperation with RDECs and Federally Funded Research & Development Centers (FFRDCs), should monitor the maturity of Ada 95 with respect to their own project needs. Items to monitor include:

- The readiness of the technology and tools,
- The upward compatibility of existing Ada 83 software (to become informed on the impact of Ada 95 on existing code),
- The adaptation of project estimation tools (to reflect early Ada 95 experience), and
- The impact of upgrading the language technology with the support tools found in the software development environment.

Within the DOD, when "competitors" may actually be part of the same overall organization, it is also possible cooperate and share lessons learned. ***Issues may be addressed by several organizations (e.g., within a PEO) by joint Ada 95 adoption efforts in which the costs and lessons learned are borne jointly by two or more groups.***

Issue: What Can Be Done to Avoid Unanticipated Social Issues Arising from the Transition?

Remember that people are involved in the adoption process. Any change can be difficult for an organization. PEOs and PMs should make sure that the adoption of Ada 95 is a participatory process and that their software developers are prepared for and actively involved in the transition planning as well as implementation phases. *While this may seem to be a "soft" issue compared with the availability of compilers or the performance of Ada run-time kernels, one of the most important lessons of previous technology adoptions is that the social and environmental changes are often the most important ones.*

Risk transfer will be a useful technique to mitigate the risk of social disruption. In this case, the PEO and PM must actively involve everyone (the staff, development contractors and support contractors) in the adoption of Ada 95. Ensuring that everyone participates in the process will also serve to ensure that the adoption is driven by the team and *not* imposed from above. This will result in the team owning the adoption effort and tailoring it to their current culture.

The Software Engineering Institute (SEI) can provide detailed information about handling change within an organization. Much of its work in this area has helped many organizations institute process improvement. ***PEOs and PMs should contact the SEI for additional information and help with this aspect of Ada 95 adoption.***

Transitioning from Another Language to Ada 95

PEOs and PMs should consider the adoption of Ada 95 to include the adoption of Ada 83, since Ada 83 is the core of Ada 95. In addition to the issues discussed in the previous subsection, there are also the following:

1) The impact of the paradigm and mindset shifts from one language (culture) to another, and

2) The impact that changing programming languages has on secondary technology (e.g., tools, environments, training, development methods, and standards).

Issue: How Best Can the Issues of Paradigm and Mindset Shifts from One Language (Culture) to Another Be Managed?

When transitioning from another language to Ada 95, the paradigm and mindset shifts are greater than they are when moving from Ada 83 to Ada 95. Each programming language brings with it a large collection of techniques and methods that reflect "how software ought to be built" using that language. Changing from one language to another is not as simple as switching from one make of automobile to another. Rather, it is more akin to switching from driving a car to piloting a jet.

PEOs and PMs should take the following steps to make sure that their staff and contractors/developers successfully adopt the Ada 95 culture:

- Support technology transfer efforts that educate the developers not only to the syntax of the new language but also to the way it should be used (i.e., its culture).

- Provide mentors with Ada experience who can help guide the development team.

- Hire experienced Ada personnel and seed them on the development team as group leaders who are responsible for showing the entire team how to create software "with an Ada mindset".

Issue: How Best Can the Issues of Secondary Technology Be Managed?

***PEOs and PMs should have their staff, RDEC or contractors/developers assess* the impact that changing programming languages has on secondary technology**. The areas to be assessed include:

- *The design method* — For example, Ada 95's support for OOP may cause the development team to examine an object-oriented method for design.
- *Technology transfer* — Many of the courses taught have programming language-specific elements, even though the courses are not about a particular language; courses on testing are a good example.
- *Documentation and process standards* — For example, MIL-STD-2167A or MIL-STD-498 may describe concepts such as "modules" in programming language-specific ways; these must be tailored to Ada 95.
- *Software Engineering Environment* — Coordinate the impact of upgrading the language technology with the support tools found in the software development environment so that the entire set continues to work together. Many tools including CASE tools have language-specific elements.

Becoming Informed about Ada 95 Adoption

Issue: What Are the Best Sources of First-Hand Information on Ada 95?

PEOs and PMs may find many sources of information at little or no cost, including those listed below. Fortunately for those transitioning to Ada 95, "buying" does not always involve spending large sums of money. Information is often free, although some time must be spent acquiring it. Some information sources provide direct first-hand information, others provide lists and pointers to direct information. PEOs and PMs should avail themselves of the following sources of information:

- *Government Information Sources* — The AdaIC and the Ada 95 Project Office are good sources. The AdaIC provides pointers to many direct sources of information; it should be the first item on any PEO's or PM's shopping list. The AdaIC can provide its information by phone, fax, mail, e-mail, bulletin board, or via the Internet. The Ada 95 Project Office can also provide first-hand information on Ada 95 and direct adopters to other sources.
- *User Groups* — A number of national, local and international user groups meet regularly to share information on Ada usage. These organizations typically offer free membership or charge only a nominal membership fee. Local groups meet monthly or bimonthly with speakers and discussions on Ada-related topics. *PEOs and PMs should make sure that they and/or their staff regularly attend such meetings to exchange ideas with other project managers.* ACM SIGAda is one such international organization with local chapters in many cities.
- *Conferences* — There are both Ada-specific conferences and general software conferences that feature Ada as one part of a larger theme. *PEOs and PMs should make sure that they or their staff attend at least one major Ada conference each year to exchange information with other managers, tap into the experience of professional consultants, and acquire up-to-date information from tool vendors.* Tri-Ada, sponsored by ACM SIGAda, is the largest Ada conference held in the U.S. each year. Other notable Ada conferences include the Washington Ada Symposium (WAdaS), the Annual Conference on Ada Technology (ANCOAT) and the Ada-Europe conference, which rotates each summer to a different European city.

- *Magazines* — The most well-known Ada-specific magazine is *Ada Letters*. This bimonthly publication features articles on technical and non-technical topics. Ada is also featured in many other magazines devoted to software issues. *PEOs and PMs should have their staff prepare summaries of information on Ada found in software magazines. They should also ensure that their technical staff monitors the technical articles in these magazines on a regular basis.*

- *Electronic Sources* — Often the most immediate and up-to-date source of Ada 95 information will be found on-line (especially via the Internet). Several kinds of information sources exist: *files of information*, which may be downloaded from several repositories; *newsgroups* or *discussion groups*, where questions may be asked and answered; *electronic mailing lists* for discussions on specialized topics such a bindings or reuse; *information browsers*, such as the World Wide Web (and its browser Mosaic) and gopher, which provide interfaces to large collections of interrelated information; and *electronic user groups* such as Team-Ada, which are a collection of volunteers who assist organizations using Ada via electronic mail.

- *Ada Tool Vendors* — Compiler and other tool vendors will provide detailed information on their products. They also are a good source for detailed tool performance measurements.

For detailed contact information on these sources, see Appendix A.

Issue: Who Can I Contact with Questions, Comments and Lessons Learned?

PEOs and PMs will find a large amount of support within the Department of Defense for all Government Ada 95 users. The Ada Joint Program Office (AJPO) is the central group within DOD that manages Ada technology. The program office sponsors the AdaIC, which is specifically set up to answer questions and collect lessons learned about Ada 95 adoption. Additionally, PEOs and PMs may draw on the experience of FFRDCs to assist in the transition to Ada 95.

Each source of first-hand information listed in this last section of Chapter 3 is also a good place to receive answers to questions. These sources also may act as forums where comments and lessons learned may be distributed. For contact information on these sources, see Appendix A.