DO-178B Certified Software: A Formal Reuse Analysis Approach

Hoyt Lougee
Foliage Software Systems

In these lean economic times, avionics manufacturers have a heightened interest in certifiable software reuse as an alternative to developing design-from-scratch software for next-generation systems. When appropriate, software reuse can provide significant return on investment and time-to-market advantages.

As indicated in the December 2004 CrossTalk article "Reuse and DO-178B Certified Software: Beginning With the Basics," [1] reuse is defined simply as "using previously existing software artifacts." These artifacts may or may not have been designed for reuseability. In fact, these artifacts may be proposed artifacts intended for future reuseability.

Artifacts include all products of a certification development process: planning data, requirements data, design data, source code, configuration management records, quality assurance records, and verification data. Artifacts extend over all functional areas of software. A sample breakdown is illustrated in Figure 1.

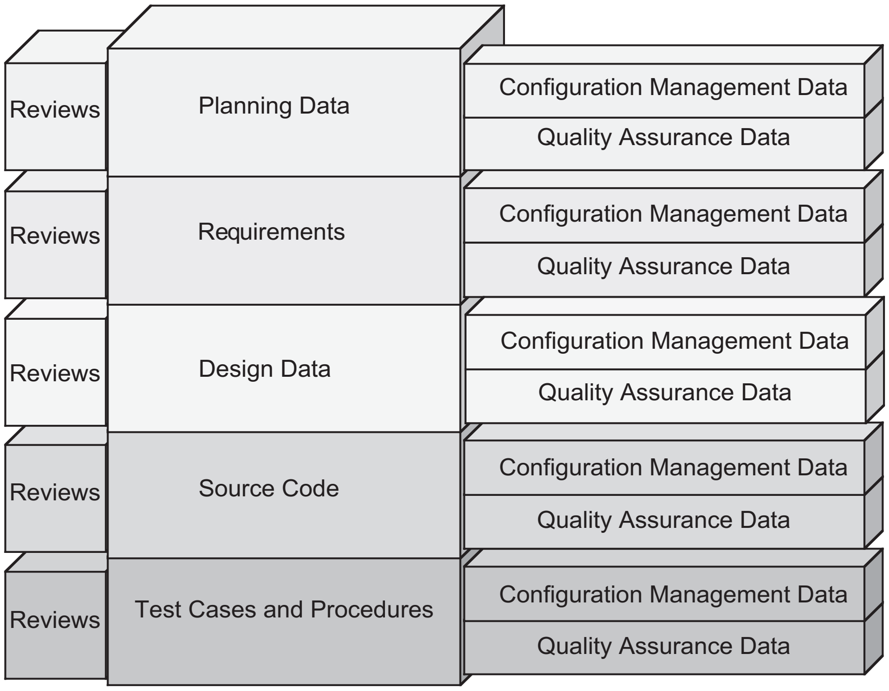
Reuse Stakeholders

In identifying the purpose and goals, and selecting the optimal approach, the reuse analysis must take into account the views of the various stakeholders. Typical stakeholders are described in Table 1. Each of these stakeholder groups normally has different expectations for reuse, as well as different insight into the pros and cons of a particular approach.

First-tier sources[1], for example, are concerned with cost, schedule, functionality, and safety. The reuse strategy particulars that provide these benefits are typically of little importance to these first-tier sources. Senior management usually represents the interests of the external customer stakeholders.

On the other hand, cost and schedule are not of paramount importance to the Federal Aviation Administration/ Designated Engineering Representative (FAA/DER). Safety is the primary driver of the certification authorities (CA). The reuse strategy selected, however, can make the CA's job easy or complex. A properly designed and executed reuse strategy will make it easier for the FAA/DER to trace verification among configurations to provide more confidence and reduce the amount of review to be performed.

Figure 1: *Reusable Artifacts*



Although cost, schedule, functionality, and safety are also important to senior management, the long-term efficiency and profitability of their product lines are also a priority, as are the competing short-term budget limitations. Senior management drives reuse strategies that accommodate the *big picture*.

Cost, schedule, functionality, safety, and long-term efficiency and profitability goals are flowed to project management. At this level, however, concerns about short-term feasibility, effects on staffing, and the lower-level tactical implementation gain importance.

System engineers drive the hardware/software functionality of the system and are often the source of system-level requirements allocated to software. Since many software strategies must be supported by or aligned to candidate hardware configurations, systems engineers are critical participants in the reuse analysis process. Cost, schedule, functionality, safety, long-term efficiency and profitability goals drive the systems engineer across both the hardware and software domains.

Development engineers and test engineers have the cost, schedule, functionality, safety, long-term efficiency and profitability goals, as well as the expectation for creating good software: software that is flexible, extensible, and sustainable. The short-term feasibility and the actual *nuts and bolts* of the implementation are also placed on their doorstep.

Quality engineers must verify the software configurations produced and must be efficient when doing so. Efficient testability is a challenge for the quality engineer in two ways:

1. Reusing tests and test results for reuseable components can reduce the amount of testing performed on new platforms, as well as reduce the overall number of tests to be created.
2. Ensuring that end-to-end functionality is verified to make certain that reused components are used correctly.

Clearly, these two goals can contradict one another. If reusable components are verified at the component level, end-to-end verification testing may redo much of the testing. Moreover, the FAA/DER typically insists that significant end-to-end testing be performed to ensure all software components integrate properly. Obviously, compromises are in order.

Finally, the configuration managers have the considerable task of configuring the reuseable elements to allow controlled, traceable software releases. This task must be performed early in the process or the reuse effort suffers accordingly. The viability of the reuse strategy depends upon the ability to configure the elements appropriately. This ability depends upon a coherent and reasonable configuration strategy, a suitable toolset, and appropriate training and staffing.

Reuse Analysis and Incorporation Process

As discussed in [1], reuse types vary according to the reasons for reuse, as well as the character of the elements to be reused; however, the reuse analysis and incorporation process is the same. Figure 2 describes a standard reuse analysis and incorporation process.

Clearly Identify Purpose and Goals

The first step in our reuse analysis and incorporation process is to clearly identify the purpose and goals of the reuse effort. The reuse purpose is the business problem to be addressed. Reuse goals, on the other hand, are the competing benefits that different reuse strategies yield.

To effectively identify purpose and goals, you must rid yourself of the idea that you can have everything: Tradeoffs must be made. Optimal purpose identification often entails initial purposes that grow, shrink, divide, or combine as the goals are examined. Therefore, involving higher levels of management is often crucial in the give and take of the purpose and goal identification.

Further tradeoffs must be made in prioritizing goals. The relative importance of potential reuse benefits will dictate the degree to which they will be considered in selecting the reuse strategy. That is not to say, however, that one benefit will completely exclude any of the other benefits; the identification of purpose and goals is based upon a relative prioritization of these factors.

Potential short-term and long-term benefits must also be addressed when identifying goals and purpose. For example, flexibility, extensibility, and sustainability typically correlate with increased long-term benefits, higher short-term cost, and longer schedule. To lower short-term cost and shorten schedules, these factors might be given lower priority.

The identification of purpose and goals entails the five steps detailed below.

Identify Purpose

As indicated above, the reuse purpose is the business problem to be addressed. The purpose is the project description over which the reuse effort will be applied and describes the scope and domain of the effort. For example, the need to introduce a new product line is a reuse purpose. Another purpose might entail creating a family of controllers – the family representing a single *project* over which reuse analysis will be performed. Yet another purpose could include functional modifications necessary to enhance an existing product line.

Note that these purposes do not and must not relate to the reuse strategy. An example of a poorly defined purpose would be "to create a common reuseable library." This purpose presupposes the results of the analysis: a reuseable library. The *correct* strategy has already been decided before the analysis has been performed. You must clearly define your purpose to allow bounding of the problem to be addressed.

Identify and Quantify Goals

Next, you must determine the relative importance of potential reuse benefits, especially in terms of long-term versus short-term benefits.

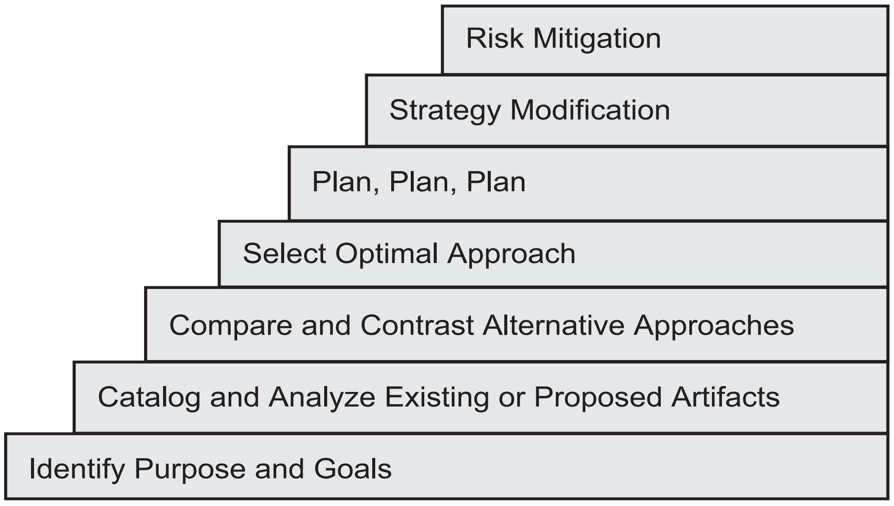
To identify goals, you must catalog (list and describe) the expected potential reuse benefits. A variety of competing reuse benefits typically present themselves. For example, all organizations want to reduce cost, shorten schedules, and lower risk. Moreover, flexibility, extensibility, and sustainability are typically considered a necessary part of every good design. Other less-tangible goals are also important. For example, customer satisfaction and market share often outweigh other cost and schedule considerations.

Next, you must quantify the identified goals. How do you quantify? To drive the return-on-investment analysis, the recommended strategy is to quantify goals in terms of cost and schedule. Cost savings are quantified by investment dollars over a specified timeframe. Schedule, of course, is quantified in calendar time – often with respect to specific milestones. Although a variety of measures can be applied to flexibility, extensibility, and sustainability, ultimately a cost and schedule impact can be identified. What is the resulting impact of alternative levels of flexibility, extensibility, and sustainability? Risks can also be addressed with cost. How much are you willing to pay to mitigate specific risks?

Stakeholder	Internal/ External
Airframers	External
First-Tier Sources (engine manufacturers, etc.)[1]	External
FAA/DERs	External
Senior Management	Internal
Project Management	Internal
Systems Engineers	Internal
Development Engineers	Internal
Test Engineers	Internal
Quality Engineers	Internal
Configuration Managers	Internal

Table 1: *Stakeholders*

Figure 2: *Standard Reuse Analysis and Incorporation Process*


Risk Mitigation
Strategy Modification
Plan, Plan, Plan
Select Optimal Approach
Compare and Contrast Alternative Approaches
Catalog and Analyze Existing or Proposed Artifacts
Identify Purpose and Goals

Although less tangible goals such as customer satisfaction and market share can also be quantified in terms of cost, much more guessing as opposed to estimating is typically involved. You may or may not want to apply costs to these factors; often, the tangible goal costs are simply compared with the understood need for these elements.

While other measures can also be effective, performing the additional analysis and estimation to assign costs facilitates the prioritization and ultimately the selection of competing strategies based on return on investment.

Prioritize Goals

After quantifying the goals, a relative priority must be given to each. Goals that have been quantified in terms of cost lend themselves well to prioritization. Intangible goals not assigned costs can be compared with this initial prioritization. For example, the lowest-cost, barely sustainable system may eventually lead to a poor marketplace reputation that must be avoided at all costs. Comparing the cost of creating the eminently sustainable system may not be worth the investment when compared to the few defects that will ultimately be addressed – except in the customer's eyes. A *simple fix* taking an *unseemly* amount of time can destroy confidence in the software and the credibility of the manufacturer.

Update Purpose and Goals

Now that the goals have been quantified and prioritized, the purpose should be re-addressed. Often, the initial goals and purpose identified should change as the tradeoffs become apparent. In fact, as the analysis continues, the goals and purpose are often adjusted again and again. Furthermore, even though the goals and purpose are assigned a baseline when the optimal approach is selected, they may change with any strategy changes or changes in business conditions.

Document Results

The final step in the goal and purpose identification is to document the analysis results:

- **Purpose:** Describe the included and excluded scope for the reuse effort.
- **Goal Priority:** Describe the priority sequence.
- **Goal Quantification:** Describe the quantification and associated rationale.
- **Long-Term and Short-Term Rationale:** Discuss the long-term versus short-term tradeoffs.
- **Assumptions:** Describe the assumptions upon which the analysis was performed, including business climate, staffing considerations, and so forth.

The relative quantifications and rationales are particularly important, so do not omit them. Often, as strategies are developed, goal prioritizations may change based on the relative ease or difficulty in implementing alternative strategies. For example, if enhanced extensibility will result in minimal cost impact but is prioritized low, revisiting the goals and prioritization may be warranted.

This documentation should be made available to all involved in developing the reuse strategy. To avoid the reuse effort taking on a life of its own (as often happens), the data should be made available throughout the reuse strategy implementation.

As indicated above, the goals and purposes may change. The update of the documentation describing the goals and priorities is crucial and is often neglected.

Catalog and Analyze Existing or Proposed Artifacts

After the purpose and goals are identified, existing or proposed artifacts must be cataloged and analyzed, and the results documented. Note that artifacts to be cataloged and analyzed may already exist or may simply be proposed. Existing artifacts are products of previous development. Sometimes, no applicable previous development is pertinent.

Cataloging Artifacts

Cataloging artifacts allows the identification of the areas of potential reuse and the relationships among them. Any data associated with previous or ongoing development are fair game for reuse. This data may include the following:

- Planning data, including the Plan for Software Aspects of Certification (PSAC), software development plan, software configuration management plan, software quality assurance plan, software verification plan, and tool qualification plan.
- Requirements data, including systems and high-level software requirements.
- Design data, including the software architecture and low-level software requirements.
- Verification data, including test cases and procedures, analysis, review records, tool qualification data, and problem reports.
- Configuration data, including software configuration index, the software life-cycle configuration index, and the software accomplishment summary.

The software configuration indexes and the software life-cycle environment configuration indexes for the potential reuse sources are excellent places to start gathering information. These documents serve as a *central clearinghouse* of information and describe directly and indirectly most of the configurable data associated with the potential source. Be aware, however, that not all reusable data are included in the configuration indexes: Configuration management and quality assurance records may not be identified in these indexes and may still be targeted for reuse.

The cataloging effort entails documenting the possible reusable artifacts and the relationships between them. Depending upon the complexity, a spreadsheet or a small database may be appropriate. You should include the following:

- **Artifact Identification:** Identify the artifacts at a useful level of abstraction. Describing each individual code file in a configuration may not be useful; groupings of files may be more appropriate (for example, low-level discrete input/output [I/O]). Although the level of abstraction should be related to the potential reuseability, ultimately the individual configurable items must be identifiable.
- **Artifact Version:** Identify the appropriate version of each artifact. When artifacts are grouped, a version identifier appropriate to the grouping should be used; for example, low-level discrete I/O associated with Release 2.2. Note that different versions of the same artifact may be applicable to the reuse analysis. This is because, over time, different versions with different desirable functionality may have been produced.
- **Related/Traced Artifacts:** The related/traced artifacts are central to certification reuse. In certifiable configurations, most artifacts are specifically related/traced by version. The ability to reuse artifacts both horizontally and vertically increases reuse benefit. Relations are described by traceability documentation, association in configuration documentation (configuration indexes), and/or applicability.
- **Configuration Location:** The physical location of the configured elements must be documented. If a configuration management system such as Clearcase or SourceSafe is used, the

location within the hierarchical configuration structure should be identified. This is particularly important when artifact groupings are used; often the path within the configuration management system is used to identify the grouped artifacts.

Artifact Analysis

Now that the artifacts have been cataloged, they must be analyzed in terms of the following:

- Functional Alignment.
- Requirements Volatility.
- Previous Development Rigor.
- Maturity of Existing Artifacts.
- Targeted Platform Changes.
- Criticality Distribution.

This analysis is discussed in depth in the companion article [1]. The results of the analysis will extend the information gathered in the cataloging stage. For each artifact (or artifact grouping), the impact of these characteristics must be described.

The specific measure used in describing the characteristic levels (including percentage, high/medium/low, relevant/not relevant, and so forth) will depend upon the needs for gradation. These measures must be selected before the analysis begins and must be consistently applied. For example, for the I/O low-level drivers, the functional alignment might be 100 percent, the requirements volatility low, the previous development rigor high (as would be expected in a previously certified product), the maturity of the existing artifacts high, no targeted platform change effects, and no partitioning for criticality distribution appropriate.

Compare and Contrast Alternative Approaches

After the artifacts are analyzed, the target software architecture must be addressed. The goal of the architecture analysis is to develop and document a number of what-if architectural candidates. To achieve this goal, the existing software architecture must be analyzed to determine how easy or difficult the incorporation into a new application will be. Two formal software evaluation processes are considered below.

Software Architecture Analysis Method

The Software Architecture Analysis Method (SAAM) [2] is simple, easy to learn, and does not require a great deal of training. The stakeholders generate a number of scenarios that describe possible future system modifications that can address the purpose and goals identified.

Description	Direct/Indirect[2]	Required Changes	Number of Changed/Added Components	Effort for Changes (estimate)
Establish lower-criticality partition for maintenance monitoring functionality.	Indirect.	Additional system safety assessment effort. Updates to architecture to enforce partitioning. Updates to PSAC. Restructuring test documentation.	Plans, requirements, design description, implementation (15 modules) test cases and procedures (25 test cases, 57 procedures) updated. Formal final testing included with overall release.	4 person months.

Table 2: *SAAM Scenario Evaluation*

Scenarios are short statements describing the interaction of one of the stakeholders with the system. Partitioning maintenance monitoring from flight-critical functionality is an example of a SAAM change. The associated SAAM scenario evaluation is illustrated in Table 2.

A number of inputs and outputs are required to perform the SAAM. One or more documented architectures are used and modified to describe potential scenarios. These documented scenarios provide a context within which the accommodation of the purpose and goals are evaluated. The primary SAAM outputs are the adjusted scenario architectures and the estimates of the anticipated costs and associated schedule. In addition, the analysis provides a greater understanding of the system functionality. Finally, the SAAM-based evaluation provides social benefits, as stakeholders come together and gain a common understanding of the costs and benefits of competing approaches.

Architecture Tradeoff Analysis Method

The Architecture Tradeoff Analysis Method (ATAM) [2] reveals both how well an architecture satisfies particular quality goals and also provides insight into how those quality goals interact with each other.

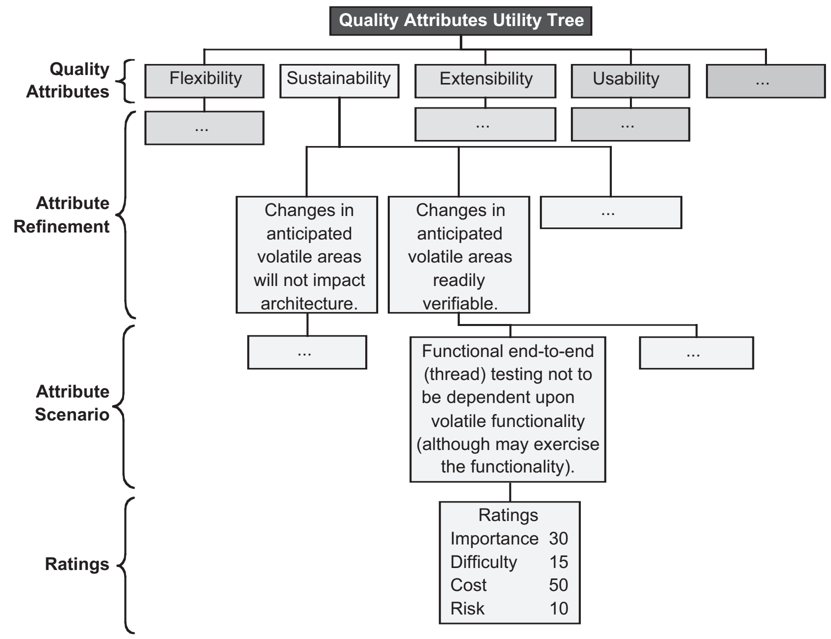
In contrast to the SAAM method, which focuses primarily on modifiability, the ATAM method provides greater insight into the quality goals: the *ilities* (flexibility, extensibility, sustainability, and so forth). The quality attribute utility tree allows a prioritization of quality attributes realized as scenarios. This tree focuses the analysis on the scenarios that address the quality attributes and ensures that the scenarios that are important to the stakeholders are addressed.

A quality attribute utility tree is illustrated in Figure 3. Typically, however, the quality attribute tree is documented with a matrix rather than with the tree format shown.

Also vital to the ATAM method are the types of scenarios addressed in the following:

- Use-case scenarios that address typical current stakeholder usage scenarios.
- Growth scenarios that address antici-

Figure 3: *Quality Attributes Utility Tree*


Quality Attributes Utility Tree
Quality Attributes
Flexibility
Sustainability
Extensibility
Usability
...
Attribute Refinement
Changes in anticipated volatile areas will not impact architecture.
Changes in anticipated volatile areas readily verifiable.
Attribute Scenario
Functional end-to-end (thread) testing not to be dependent upon volatile functionality (although may exercise the functionality).
Ratings
Ratings
Importance 30
Difficulty 15
Cost 50
Risk 10

pated changes to the system.
- Exploratory scenarios that address extreme changes expected to stress the system.

Addressing these scenarios allows changes to the system to be considered up to and beyond the current boundary conditions of the current design.

Selecting the Appropriate Architectural Method

So which approach should you choose? Clearly, SAAM is a smaller, simpler, less expensive approach, whereas, ATAM is larger, more complex, and more expensive. Three factors should drive the selection: the complexity of the existing architecture, the potential complexity of the resulting architecture, and the approximate size of the effort. If the architectures are complex and the size of the anticipated effort is large, then the cost of ATAM is well worth the additional reduction in risk. On the other hand, if the architectures are not especially complex and the effort is small (compared to a two-week effort with a dozen people), the reduction in risk may not warrant the cost.

Select Optimal Approach

At this point, the selection of the optimal reuse approach should be made, selecting from the number of *what-if* architectural candidates that have been created based on the analysis method used from the earlier section, "Compare and Contrast Alternative Approaches." Keep in mind that the entire point of reuse is to increase the return on investment (ROI) while ensuring continued business viability. You should now have documented alternative architectural approaches that can be assessed in terms of the purpose defined and the prioritized goals. As with the initial identification of goals and purpose, all stakeholders should participate in the approach selection process. Moreover, the goal of approach selection is agreement and buy-in – all the more reason to ensure that all stakeholders are involved.

The selection of the optimal approach and the associated ROI analysis considers the following:
- Determining cost incurred over time is perhaps the most difficult part of the ROI analysis, especially accurately determining the long-term versus short-term benefits – and then making the appropriate tradeoffs. As a rule of thumb, the more work performed *up front* in emphasizing maintenance, extensibility, and flexibility, the less follow-on work will be required. The best way to model this comparison is to plot anticipated expenditures over time and compare those expenditures to the company's financial capabilities and the overall business schedule considerations.
- Schedule considerations are typically driven by either the schedule imposed by customers, or by market conditions and the need to field products before your competitors. Although schedules imposed by customers tend to be immutable, more schedule flexibility is typically available with marketing strategy tradeoffs. In both cases, the earlier the reuse strategy is considered and implemented, the more schedule flexibility will be available in both cases. Clearly, development of a new reuse library with elements designed for reuse is inappropriate for a slight change to an existing certified configuration on a tight schedule. On the other hand, if the change is slight but the schedule is not critical *and* there are other uses for a reusable library (for example, future families of similar products that would benefit from the effort), the migration of a known application may be the ideal means to introduce the reuse.
- If the reuse strategy implementation is to be funded internally by company investment, the work can be performed to align to projected needs. Otherwise, the work must be performed within the schedule permitted by the customer. Note that the risk both increases and decreases with investment-funded reuse implementation. Clearly, schedule risk decreases; however, the risk of losing funding partway through the reuse implementation and resorting to *fast-and-dirty* development increases.
- The costs associated with reuse must be carefully estimated and documented – especially with design for reuse. Too often, manufacturers either underestimate or shift funding away from design-for-reuse activities. As a result, many of these efforts fail. The software that was intended for reuse turns out to be only applicable for a single use, but that software was more expensive because of the aborted *reusability* work. In fact, scavenge reuse is often the product of aborted design-for-reuse efforts. Unfortunately, when the reuse library does not materialize and the costs associated with each scavenge reuse instance exceeds the initial estimates based on design for reuse, reuse is given a bad name.
- Effects on intangible factors must be considered. Is the best always the lowest cost? Intangible benefits often cause more expensive approaches to be selected over less expensive approaches. Company limitations on available funding often stand in the way of the most efficient approaches as well. Therefore, the short-versus-long-term analysis is crucial to the approach selection. You must also consider the lifespan and anticipated breadth of applicability of the reuse elements. The lifespan of an application concerns the longevity, with changes, for a particular configuration. The breadth of applicability concerns the number of different applications that will make up the reuse target family.
- Plausibility and associated risk are a concern. Selecting the optimal approach is to judge the various approaches in terms of the identified purpose and the prioritized goals. When it is not possible to select among the candidate scenarios and still attain the purpose and goals, the goals and purpose must be adjusted or new scenarios generated. Risk also must be considered when selecting among competing architectures and may require the adjustment of the goals and purpose. An informed decision must be made on how much risk the organization is willing to carry. You must ensure that reasonable expectations are set.

"Determining cost incurred over time is perhaps the most difficult part of the ROI analysis, especially accurately determining the long-term versus short-term benefits – and then making the appropriate tradeoffs."

Plan, Plan, Plan

Reuse planning is key to the success of

reuse. Effective reuse planning always includes hardware considerations; therefore, a considerable amount of planning and analysis will occur *outside the realm of traditional software plans*.

Two types of reuse planning are necessary: certification planning and project planning. Certification planning encompasses creating the planning data required by the certification authorities (typically to meet RTCA DO-178B data guidelines). The project planning concerns the internal schedules, budgets, and staffing considerations.

The key certification-planning document for reducing reuse risk is the PSAC. The strategy by which reuse is to occur, including especially partitioning considerations, should be provided to the FAA in the PSAC as early as possible to prevent costly project missteps. Another critical document for reducing reuse risk is the configuration management plan. The configuration of reusable components and the tracking of changes among different reuse instantiations are often neglected and can impact both cost and schedule (as well as embarrassment).

An additional certification-planning consideration concerns the FAA/DER used. When incorporating reuse, try to use the same DER for all reuse instantiations to allow him/her to become comfortable with the reuse data and process.

Because much of the analysis data will flow directly into the detailed project planning, the data must be realistic. Tracking against unrealistic expectations ensures failure. Many successful reuse efforts can be viewed as failures because the documented expectations were unrealistically high. Other reuse efforts fail because they are abandoned when early tracking data deviates from unrealistic plans.

Strategy Modification

As indicated above, reuse strategies can be changed mid-stream. Changes can occur based on changes in business conditions (especially funding), progress not matching plan, and so forth. When strategies change, the documented reuse analysis data is key in determining the next best or appropriate alternative path. All stakeholders should be involved in the decision to change strategies to first determine if strategy should change and then, if necessary, to determine how the strategy should change. The same consideration applied to initial strategy selection should be applied to strategy changes.

When strategies are modified, the revised purpose and goal expectations must be documented and communicated. Reuse must be tracked against the appropriate plan; otherwise, even though reuse provided meaningful savings (albeit less than initially expected), the reuse process will lose credibility.

Risk Mitigation

Reuse risk mitigation includes reigning in the scope of the reuse activity, overcoming the *not-invented-here* mindset, and avoiding (where practicable) *bleeding-edge* technology. Furthermore, a clear purpose and goals, a solid analysis, careful planning, and stakeholder buy-in mitigate the risk of reuse.

These pitfalls represent the common major challenges that you will face when implementing reuse. A myriad of other programmatic and technical risks specific to the particulars of the product and companies will plague the reuse effort. These risks include both normal development risks plus risks associated with the reusable aspect of the development.

The target of the risk process may be different, spanning many products instead of a single product. Schedules and milestones must be coordinated: Risk of delay in one product instantiation affecting the schedule and milestones of another product instantiation must be addressed. Moreover, you must address the misalignment risks associated with planning and executing the reuse-specific tasks that could include separate plans, requirement documents, design documents, and so forth.

Conclusion

Cost and schedule time can be saved and safety can be enhanced with reuse for DO-178B certifiable software. To attain maximum reuse benefits, however, you must be rigorous in your approach to the planning, analyzing, execution, and tracking of reuse. This article outlines a rigorous reuse process that provides a road map to reuse success. Keys to reuse success include the following:

- Involving the appropriate stakeholders throughout the reuse analysis and incorporation process.
- Identifying clearly the reuse goals and purpose.
- Performing a rigorous architectural analysis.
- Planning and tracking in detail reuse execution.
- Reviewing and documenting any necessary mid-stream strategy changes.
- Applying risk mitigation to the reusable aspects of development.

Addressing these key issues will allow you to take control of the success or failure of your reuse effort and, ultimately, to control your company's bottom line and continued competitiveness.◆

References

1. Lougee, Hoyt. "Reuse and DO-178B Certified Software: Beginning With the Basics." CrossTalk Dec. 2004 <www.stsc.hill.af.mil/crosstalk>.
2. Clements, Paul, Rick Kaman, and Mark Klein. Evaluating Software Architectures. 1st ed. Addison-Wesley, 15 Jan. 2002.

Notes

1. In this example of stakeholder breakdown, major aircraft sub-system suppliers (for example, engine manufacturers) are the first-tier suppliers to the airframers. The software organization discussed would be part of a second-tier supplier, supplying components directly to the first-tier suppliers. Often, software supply sources populate even lower tiers in the supply chain.
2. Direct scenarios are currently satisfied by the system architecture; indirect scenarios require a modification of the architecture.

About the Author

Hoyt Lougee is the engineering manager, Aerospace Division, at Foliage Software Systems. Foliage delivers DO-178B process and technology consulting, custom software development, and independent verification and validation. Lougee's responsibilities include program management and software process improvement. Previously with AlliedSignal/Honeywell, Lougee has more than 13 years of experience with both military (DOD-STD-2167A) and commercial (RTCA DO-178B) aviation software development and certification efforts. Lougee has authored a number of white papers and presented at the 2002 Digital Avionics Systems Conference.

Foliage Software Systems
168 Middlesex TPKE
Burlington, MA 01803
Phone: (781) 993-5500
Fax: (781) 993-5501
E-mail: hlougee@foliage.com