Reducing software development cost, schedule and risk using AGI software

Kevin M. Flood
Analytical Graphics, Inc.
220 Valley Creek Blvd.
Exton, PA 19341
www.agi.com

August 19, 2009

Abstract

This paper summarizes the cost, schedule and risk advantages of using Analytical Graphics, Inc. (AGI) software for relevant development activities as compared to custom development and "freeware" options. The results are based on metrics generated using established cost and risk models and benchmark development projects that used AGI software. AGI offers an approach for conducting such comparisons using standard cost models, recognizing results will vary based on assumptions and program-risk settings.

1.0 Overview

Using a widely accepted cost-assessment tool, COCOMO (COnstructive COst MOdel),[1,2] AGI conducted a comparison of three software development alternatives for a hypothetical project:

- Custom development
- "Freeware"[3] integration
- AGI software integration

Based on nominal settings for the custom-development and "freeware" cases, and worst-case licensing assumptions for AGI software, AGI found the integration of its software to be the lowest-cost and lowest-risk approach. On a relative basis, the lifetime costs and overall development schedules compared as shown in Figure 1.

---

[1] COnstructive COst Model (COCOMO) II, University of Southern California Center for Systems and Software Engineering, *http://csse.usc.edu/csse/research/COCOMOII/cocomo_main.html*.

[2] Abts, C. M., "Extending the COCOMO II Software Cost Model to Estimate Effort and Schedule for Software Systems Using Commercial-off-the-Shelf (COTS) Software Components: The COCOTS Model," University of Southern California, PhD Dissertation (May 2004).

[3] For the purposes of this analysis, "Freeware" refers to any reused software that requires no additional cost to acquire for the development activity. This would include open source, Government-Off-The-Shelf (GOTS) or internally developed and reused code.
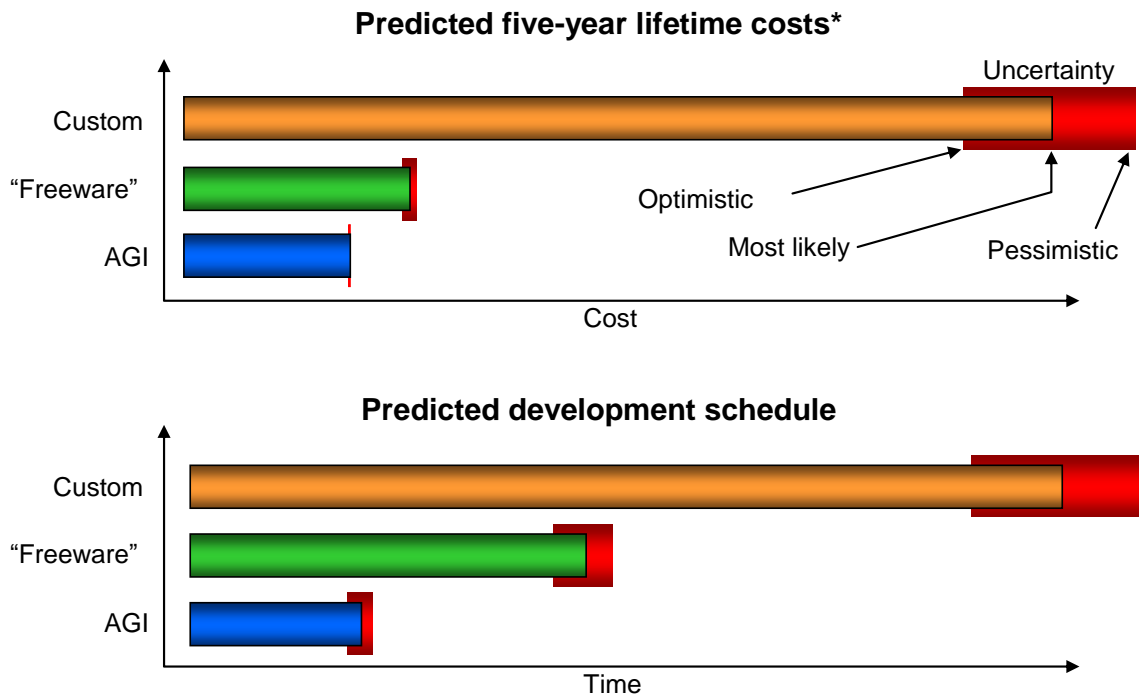
Predicted five-year lifetime costs*

Predicted development schedule

Figure 1: COCOMO model predictions for lifetime software costs and development schedule for comparable custom-development projects, "freeware" integration, and AGI-software integration. *Cost estimates include only software development and maintenance and do not include operational savings because of ease-of-use advantages of end products. For information about operational cost savings, see the 2008 Frost & Sullivan report on the use of AGI software.[4]

2.0 Background

Much has been published by independent researchers, industry associations and the U.S. government regarding the use of Commercial-Off-The-Shelf (COTS) software, even for mission-critical aerospace and defense applications.[5] Despite well-constructed and researched evaluation criteria, the adoption of COTS for mission-critical activities remains slow because of perceived program risks. In this paper, AGI conducts an objective assessment using the COCOMO cost-estimating tool and benchmarks from its customers' integration projects to compare the predicted cost, schedule and risks of

---

[4] Fishering, David, "An Assessment of the Benefits Associated with Software by Analytical Graphics, Inc.," Frost & Sullivan, 2008. http://www.agi.com/downloads/products/by-capability/FrostSullivan_ROI_AGI_Software.pdf

[5] See, for example, American Institute of Aeronautics and Astronautics, "Managing the Use of Commercial-Off-The-Shelf (COTS) Software Components for Mission-Critical Systems," AIAA G-118-2006, October, 2006.

several development approaches. We expect that other cost-estimating tools, such as TruePlanning® by PRICE® Systems or SEER® by Galorath, Inc., would generate comparable results using the inputs and assumptions. We have not validated our results using those products, however we do identify all of the key assumptions so others may replicate our results, repeat them for different project parameters or validate them using other cost-estimating products.

The rationale for conducting this study was reinforced by the continued trend of cost overruns and delivery delays for major software development activities in the aerospace and defense industries. Such issues are widely recognized, and have prompted guidance from the U.S. General Accountability Office[6] that recommends the use of parametric tools for estimating software development effort and schedule. Figure 2 shows representative trends reported by The Standish Group[7] from 1994 through 2008 for software development project success and failure rates. These trends illustrate the challenges associated with software development activities. Despite better project-management approaches, such as the use of iterative development, overall improvements have been slow during the reported 14-year period.
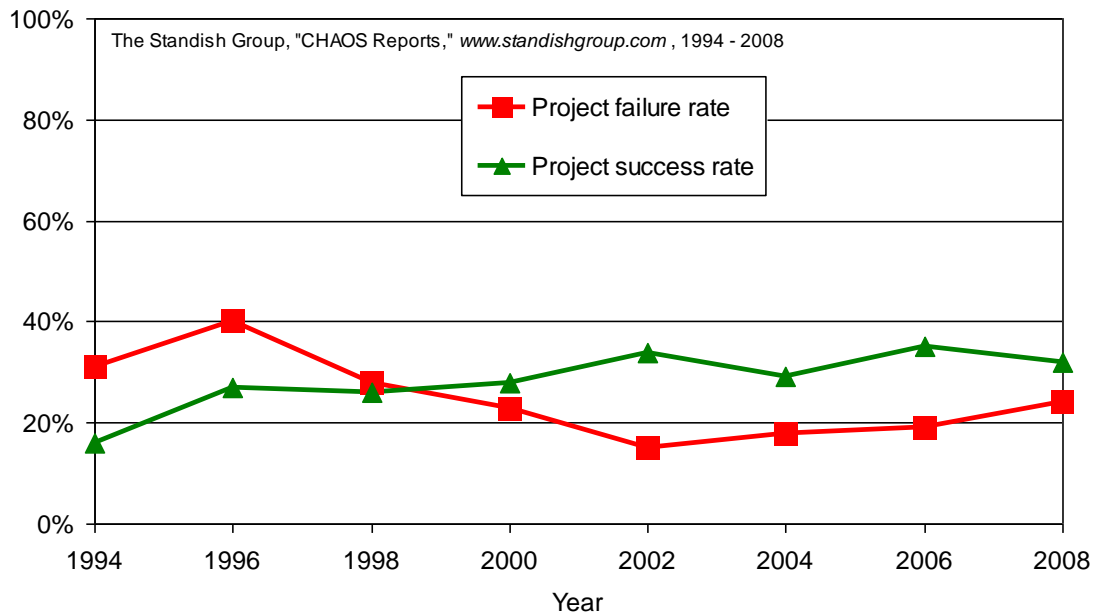

Figure 2: Success- and failure-rate statistics reported by The Standish Group from 1994 through 2008. "Failure" is defined as project cancellation prior to completion and "success" is defined as on-time, on-budget delivery with all requirements met. The remaining "challenged"

---

[6] United States Government Accountability Office (GAO), "GAO Cost Estimating and Assessment Guide: Best Practices for Developing and Managing Capital Program Costs." GAO-09-3SP, March, 2009. Specifically, see Chapter 12, "Estimating Software Costs."

[7] The Standish Group International, Inc., "The Standish Group CHAOS Reports," http://www.standishgroup.com, 1994 – 2008.

projects, not shown here, completed with some elements of cost, schedule or technical requirements unsatisfied.

3.0 Analysis assumptions

The analysis reported here used the following assumptions for the COCOMO model for all cases:

Scope – The assumed scope for this hypothetical project is constrained to:
- Replicating only the capabilities existing in the AGI products.
- Additional integration effort for "freeware" and AGI integration to create glue code or otherwise attach the APIs to a larger application.

The project is assumed to be a development activity for a new capability or application that will have a lifetime of at least five years after completion of the initial software development.

Software size – AGI used a baseline of 250,000 source lines of code (KSLOC) defined in a manner consistent with the Software Engineering Institute.[8] This size was chosen because it was within the range of calibration of the COCOMO model, but still represented a relatively simple development that could be classified as a "single module" within the scope of both the COCOMO model and AGI's software.

Labor rates – The labor rate used was an average of a "Junior Analyst" and "Senior Analyst" as defined on typical Government Services Administration (GSA) contracts. For the purposes of this analysis, AGI used an average of all the published rates for these labor categories for the "Millennia Government wide Acquisition Contract (GWAC)" since it incorporated a range of contracting organizations for large-scale system integration.[9] A schedule of 152 labor hours per month (the COCOMO default) was assumed for schedule estimating.

Project phase – AGI conducted the analysis for a "post-architecture" phase; that is, the results include only costs incurred after the engineering is completed and the software architecture is determined. This choice was made to focus the analysis on the software-development portion of the project. However, it is likely that even greater advantages exist for AGI software integration (where applicable) when the entire life cycle is considered, as considerable effort is required for both the development of computational algorithms and the software architecture.

---

[8] Park, Robert E. "Software Size Measurement: An Architecture for Counting Source Statements (CMU/SEI-92-TR-20, ADA258304)." Software Engineering Institute, Carnegie Mellon University, September 1992.

[9] See the U. S. General Services Administration Web site for reference – *http://www.gsa.gov/millennia*.

Throw-away code – The COCOMO model recognizes that some code developed never makes it into the countable code base and includes a "throw-away" factor (REVL) to account for that reality. AGI set this value to 0, which gives an advantage to the competing approaches because, in general, the percentage of throw-away code increases with increasing development scope, and utilization of AGI's software represents the lowest level of new-capability development scope.

Programming language – AGI uses several object-oriented approaches for its software products. Therefore, this option was set to "Object-oriented general" for the purposes of the analysis.

Effort Adjustment Factors (EAF) – The COCOMO model incorporates factors to account for requirement attributes of the software product, delivery platform and the overall project. It also incorporates experience and capability attributes for the development organization, and the requirements, the state of the development team and project attributes. This analysis used all "nominal." Again, this likely gave an advantage to the development and "freeware" cases, as the class of functionality under consideration for many aerospace and defense applications probably has a complexity factor greater than nominal.

All other parameters were set to their default or nominal values. Settings unique for each of the three approaches are defined in Table 1.

4.0 Analysis results

This section summarizes the results of the cost analysis and provides a brief description of the observed cost and schedule drivers.

4.1 Cost

The cost analyses are broken into initial development and integration and the maintenance required for a five-year program lifetime. Together, those elements compose the life-cycle cost for the hypothetical project. The results are summarized in the normalized graphs of Figure 2.

As implied in the graphs, the uncertainty in the development and integration costs is driven by the overall labor effort. Both the AGI and "freeware" reuse cases reduce the labor costs and therefore the project cost uncertainties. Labor for these cases includes the assessment and assimilation costs to determine the applicability of the software, which is a cost borne by AGI as part of its commercial practice. For the AGI case, the development cost is dominated by the license fee, which for the analysis represents the maximum possible program license fee.[10] AGI's cost uncertainty is the smallest of the three cases.

---

[10] The maximum license fee is the bounding case for a license to a single development program and its subsequent deployment. See *http://www.agi.com/licensing* for more information.

Table 1: Approach-unique settings for the COCOMO software

	Parameter		Setting	Rationale
Custom	Code base		250 KSLOC	
	Reused code	Design change	NA	
		Code change	NA	
		Integration	NA	
		Assimilation	NA	
	Maintenance	Code modified per year	10% of code base	Moderate value for evolution of a new "product."
		New code per year	10% of code base	Moderate value for evolution of a new "product."
"Freeware"	Code base		250 KSLOC	
	Reused code	Design change	5% of code base; glue code ignored	Assuming a source-code-based integration with only 5% of the code base requiring rework for the new capability.
		Code change	5% of code base; glue code ignored	Assuming a source-code-based integration with only 5% of the code base requiring rework for the new capability.
		Integration	5% of code base; glue code ignored	Assuming a source-code-based integration with only 5% of the code base requiring rework for the new capability.
		Assimilation factor	4	Based on a scale of 0-8, this factor represents the effort needed to determine if the reused software is appropriate. The COCOMO definitions identify a factor of 4 as, "Some module test and evaluation (T&E), documentation."
		Unfamiliarity	0.4	Corresponds to a development team that is "somewhat familiar" with the "freeware" being integrated.
	Maintenance	Code modified per year	5% of code base; glue code ignored	Half of the source-code maintenance for a full new development, where the other half is assumed to occur through other means (related projects, open-source community, etc.).
		New code per year	5% of code base; glue code ignored	Half of the source-code maintenance for a full new development, where the other half is assumed to occur through other means (related projects, open-source community, etc.).
AGI	Code base		250 KSLOC	
	Reused code	Design change	0	Included in license.
		Code change	0	Included in license.
		Integration	3%	Yields slightly longer schedule than the AGI benchmark of 6 months for a project to integrate a single module.
		Assimilation	0	Included in AGI business practice.
		Unfamiliarity	0.4	Corresponds to a development team that is "somewhat familiar" with the AGI software.
		Software license	Program	Worst-case cost and upper bound for the purposes of this study.
	Maintenance	Code modified per year	10% of glue code	Cost for modifications to code base included in maintenance fees.
		New code per year	10% of glue code	Cost for modifications to code base included in maintenance fees.
		Maintenance fees	10% of license fee	AGI standard fee for development software.

When considering the maintenance cost, the COCOMO model bases estimates on the assumptions about the amount of the base code requiring modification and new code added to the code base. For commercial software providers, such as AGI, these costs are borne as part of their commercial practice and included in maintenance fees. The model also predicts the amount of integration code ("glue code") based on AGI's assumption of 3% integration effort relative to the custom development. This assumption comes from prior AGI projects that resulted in deliveries of initial operational software for one or more AGI modules in 3-6 months.[11, 12] A 3% integration setting estimates a duration of 6.4 to 7.4 months, which is a conservative estimate in line with AGI's findings. Labor efforts for maintenance of the AGI glue code are included in the results.

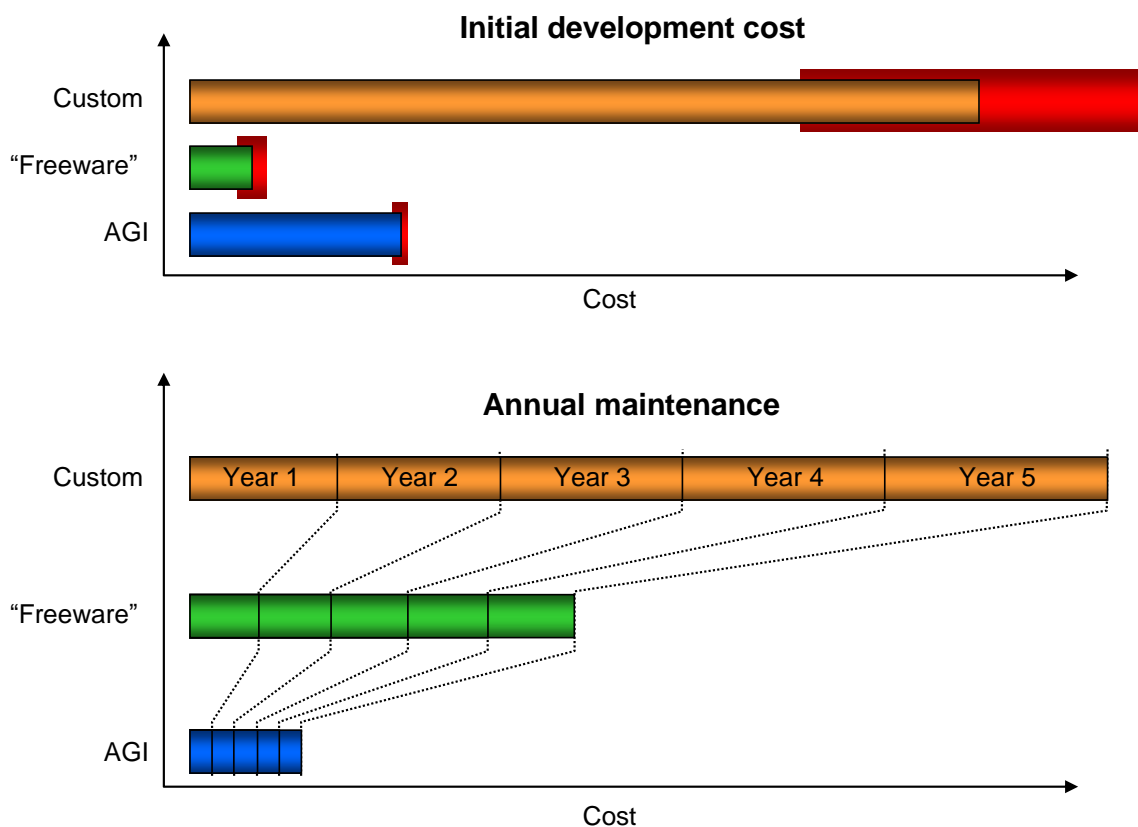**Initial development cost**

Annual maintenance

Figure 2: Normalized costs for initial development and integration (upper graph) and ongoing maintenance (lower graph) using the assumptions in Section 3.0 and Table 1. The uncertainty for the development projects is illustrated in the upper graph by the bounding boxes at the end of the bar.

---

[11] "A Case Study: RAF Fylingdales – Space Situational Awareness," Analytical Graphics, Inc, 2006.

[12] Krause, Adam – ITT Space Systems Division, "AMMP – Airborne MASINT Mission Planning," AGI Users' Conference Presentation (http://www.agi.com/events/2008-users-conference-resources/), October, 2008.

The center bar represents the likely outcome as predicted by the COCOMO software.

As summarized in Table 1, AGI assumed a nominal code modification of approximately 10% per year and development of 10% additional code each year. Similarly, the analysis assumed that the "freeware" code is maintained by the developer at a source code level. Since some "freeware" sources may generate upgrades that are pertinent to the project without project funding, the analysis assumes that the program is responsible for updating about half the code base relative to the full development project, which translates to modifying 5% of the code per year and increasing the base code by 5%. The labor for this base-code maintenance dominates the costs for the "freeware," and the additional maintenance for the glue code is ignored as it is quite small on a relative basis.

An important cost and risk tradeoff for commercial software is the scope of an initial software license. Thus far the analysis has assumed a license that bounds the cost for all users. However, the reason software licensing exists in its current form is because it attempts to match cost to measurable value metrics (such as products, number of users, duration of use, etc.). To illustrate the cost-risk tradeoff associated with conventional licensing, consider a software license that accommodates 1,000 users compared to one with unbounded access. Such a comparison is shown in Figure 3 for the AGI software. As illustrated in the figure, a license for 1,000 users has the lowest development and lifetime costs of the three options evaluated.
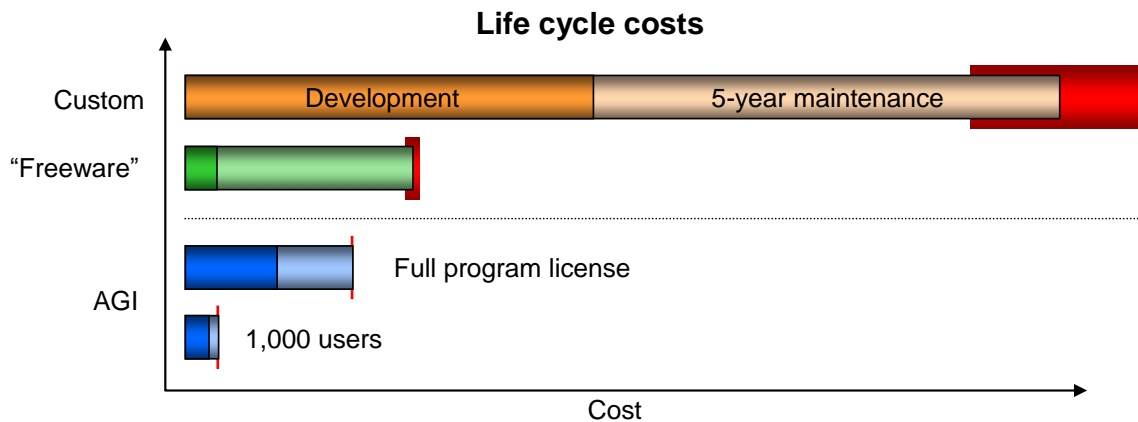

Figure 3: Normalized lifetime costs combining the results from Figure 2. The uncertainty for the development project is shown at the end of each bar by the bounding boxes. The graph shows the lifetime costs for the bounding AGI case of a program license and compares that with a license for 1,000 perpetual users. Development costs correspond to the darkly shaded, left portion of the bars and the maintenance costs correspond to the lightly shaded, right side of the bars.

The cost savings illustrated in Figure 3 suggests that commercial software providers consider offering programs bounding license fees in order to control risk and support meaningful cost trades. Likewise, programs implementing COTS should develop means

to estimate and budget for actual usage parameters (such as number of users) during program lifetimes. The uncertainty in cost is really not a risk factor, as the bounding case still results in a lower overall program cost than the alternative options. And failing to take advantage of commercial software licensing metrics may increase the cost well beyond that required to meet the needs of the program.

4.2 Schedule

The COCOMO model also predicts program duration based on the level of development effort required and the parameters associated with the program's requirements. Based on the settings described earlier, the relative development timelines compare as shown in Figure 4. In line with the results for the cost analysis, the overall labor effort correlates directly with the uncertainty in the development time.
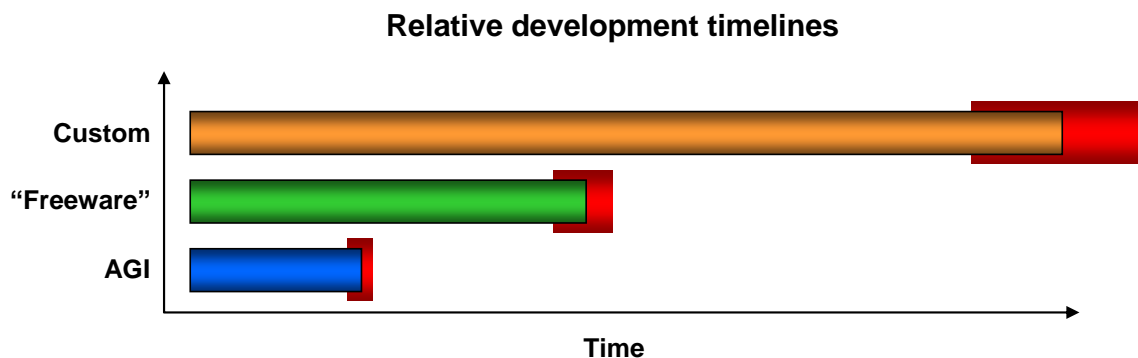
Relative development timelines

Figure 4: Normalized development timelines. The uncertainty for the development project is shown at the end of each bar by the bounding boxes.

4.3 Risk

Much documented evidence exists regarding the advantages and risks of COTS-based development projects.[13,14,15,16] A common theme among the documented risks is that they frequently reinforce perceptions of rigid COTS business practices that do not map easily into development and integration life cycles. While the risk assessment

[13] Galorath, Daniel D. and Evans, Michael W., Software Sizing, Estimation, and Risk Management. Boca Raton, FL: Aurbach Publications, 2006.

[14] Yang, Y., Boehm, B., and Clark, B., "Assessing COTS Integration Risk Using Cost Estimation Inputs", ICSE 2006.

[15] Mikiewicz, A. F., "The Real Costs of Developing COTS Software," IEEEAC paper #1159, Version 3, 2003.

[16] Yang, Y., Boehm, B. and Wu, D., "COCOTS Risk Analyzer," Proceedings of the Fifth International Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems, IEEE Computer Society, 2006.

recommendations incorporate consideration for these practices, it is can be difficult and time consuming for programs to evaluate these factors among various vendors, encouraging them to default their thinking to the most rigid preconceptions. Table 2 summarizes the primary concerns that are raised throughout the literature regarding COTS risk trade offs. The following paragraphs describe technical approaches and business practices that AGI employs specifically to maximize COTS advantages and minimize risks.

Table2: Relevant COTS advantages and associated risks[17]

COTS Advantages	COTS Risks
Avoid expensive development and maintenance	Up front license fees and recurring maintenance fees
Rich functionality	Unnecessary features that compromise usability
Upgrades anticipate future requirements	Dependency on vendor
Keeps pace with technology	Synchronizing upgrades

4.3.1 License and maintenance fees

Sections 4.1 and 4.2 illustrate the cost and maintenance advantages cited by researchers. However, the license fees, often viewed as a risk factor, can actually be used to the advantage of development programs to minimize cost risks throughout the program life cycle. Of course, this requires cooperation of the vendor. AGI implements three licensing elements that actually reduce risks compare with other options that are driven by labor costs:

1. **Barrier-Free Software Development Licenses** – AGI uses multiple models for licensing its development kits and the runtime applications that are produced. The spectrum ranges from completely free development kits with runtime-license fees to licensing of the complete software stack with unlimited distribution rights.

2. **Milestone-based license fees** – AGI supports milestone-based licensing fees. Milestones may be event-based or, for programs committing to multi-year projects, time-based. Figure 5 shows a five-year time-based cost schedule for the hypothetical example used throughout this analysis.

---

[17] Adapted from 13.

3. **Risk sharing** – Because the predominant costs for using AGI's software are for licensing and not labor, there is flexibility for risk sharing that does not exist for new development. AGI licenses its software at a fraction of the full cost for a risk reduction phase of a program provided the project commits to the license fees should the project proceed to full deployment. If the project should be cancelled prior to deployment, the program saves considerable expense as a result.

In AGI's case, license fees may be associated with the program's timeline or milestones. For example, a program's commitment to a five-year program would support a distributed license deployment during that lifetime.
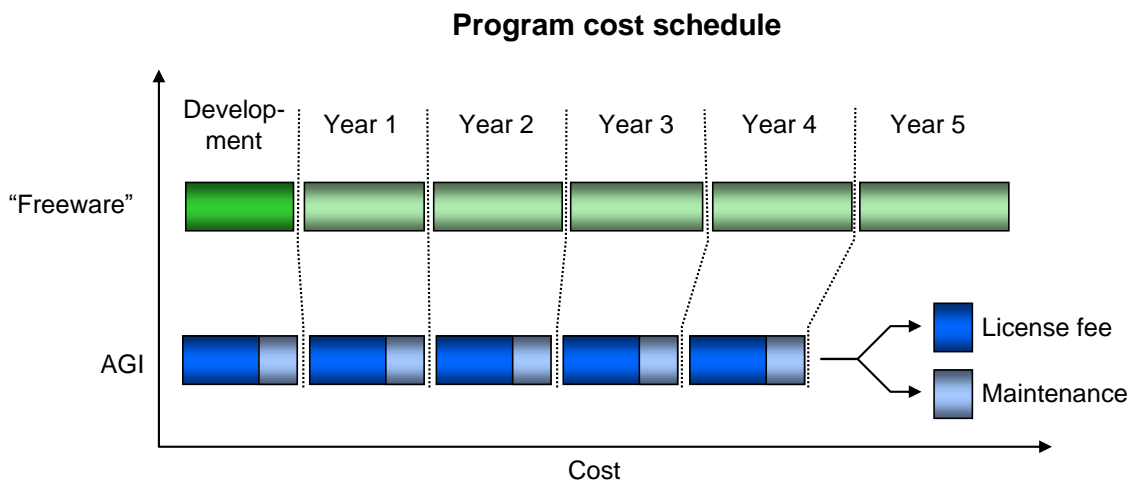
Program cost schedule

Figure 5: Illustration of a five-year cost schedule comparing a "freeware" development with one completed using AGI software. Note that the license fees are divided equally among the development phase and first four years of the five-year program. Note that the maintenance fees, paid in advance, cover maintenance for the successive year (i.e., there is not a missing maintenance cost in year 5).

Further, the analysis illustrated in the graph in Figure 2 shows that bearing the full labor cost of maintaining a code base (custom development, and to a lesser degree "freeware") is always more expensive than the AGI commercial maintenance rate. The common counterpoint is that COTS software providers deliver and maintain more capabilities in their products than are required for the specific applications. However, AGI licenses its software on a capability basis for development projects, which invalidates the above argument. And, when taken in the aggregate, commercial software upgrades are almost always ready sooner since the manufacturer does not wait for specific requirements before implementing enhanced functionality, performance improvements or greater platform flexibility.

4.3.2 Usability

As it pertains to software development activities, "usability" includes platform support, API flexibility, documentation, support and training. A number of COTS providers deliver their software in limited form factors, constraining usability and limiting flexibility for adapting the software's use as a program evolves. Conversely, AGI delivers its software in application form, embeddable application engines and low-level libraries to provide the greatest flexibility for developers. All forms of the software, including the desktop applications, may be extended by adding user-defined functionality and work flows. A majority of AGI's development software runs on Windows, UNIX and Linux platforms and supports Microsoft-standard (COM and .Net) and Java interfaces.

4.3.3 Vendor dependence

COTS vendors prepare software to address the needs of a marketplace and frequently look ahead to future requirements to ensure that their products not only meet current requirements but also provide capabilities for unspecified needs, anomaly situations or unexpected cases. [18] The commonly perceived risk associated with COTS software is that the development program reduces its options for future modifications because of commitments to the vendor's software platform.

In reality, any development activity of any consequence accepts significant risk when it commits to its baseline platform and/or primary development contractor. In this regard, commercial software can deliver advantages for the end customer because vendors build their software and business practices (training, support, services and resources) for all developers. Therefore, the end customer has more flexibility in choosing initial or future development contractors than they would for a custom-developed project.

Further, many commercial platforms are designed for extensive reuse across multiple product lines. Additionally, they commonly come with built-in interoperability with software from other vendors, which, again, provide a significant advantage for a development program, particularly during the out-year spirals of an iterative development when interoperation requirements commonly evolve. Quantitative cost and schedule benefits of COTS can be shown using the COCOMO model.

Within AGI's product line, for example, there are reusability features embodied by APIs for user-interface extension, Web integration, control embedding, general automation, data integration and functionality extension. Furthermore, there is built-in support for interoperability with other commercial software product lines including ESRI's ArcGIS, MathWorks' MATLAB, MAK Technologies' VR-Forces, Scalable Network Technologies' QualNet, Simulyze's Flight Control, Google Earth and Microsoft Bing. This level of interoperability is not reflected in the preceding cost and schedule tradeoffs,

---

[18] See, for example, Landers, Jamie. "AGI White Paper: Leveraging Commercial Off-the-Shelf Software to Identify and Mitigate Risk during Launch Vehicle Operations," *http://www.agi.com/whitepapers*, 2006.

as the required interoperability for the COCOMO analysis was "nominal." However, by adjusting the COCOMO product reusability parameter (RUSE) for "reuse across multiple product lines," one can estimate the value of this level of interoperability. Figure 6 shows the cost and schedule impacts of requiring commercial-grade reusability for custom development.
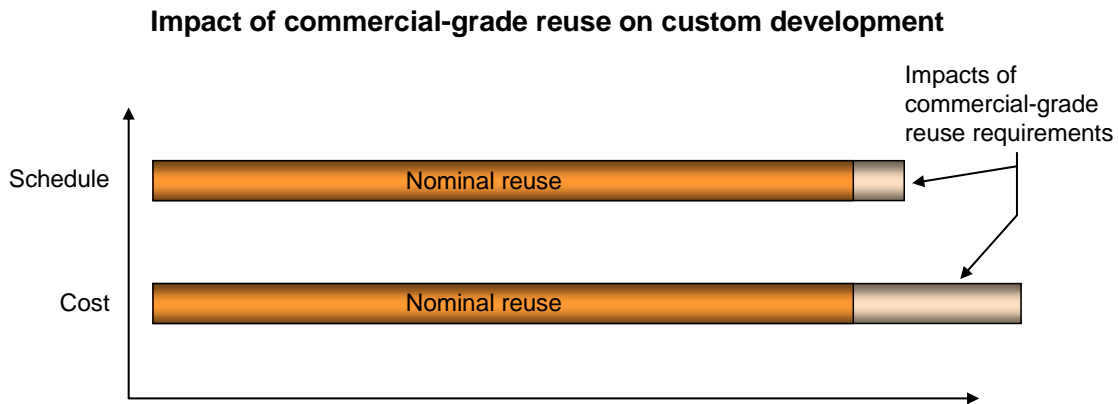
Impact of commercial-grade reuse on custom development

Figure 6: Relative impacts of requiring commercial-grade reuse on schedule and cost for a custom-development project. For reuse "across multiple product lines," a COCOMO RUSE setting of "extra" high is used.

4.3.4 Synchronizing upgrades

Each development program has a unique deployment schedule that includes a process for integrating software updates. A repeated concern when using commercial software is that the program is, to some degree, at the mercy of the vendor's upgrade schedule. While no vendor could possibly adjust its release schedule to suit every program that uses its software, in most cases the vendor can accommodate the program's update schedule if coordinated in advance. Both the vendor and integrator have responsibilities for ensuring effective upgrading. Two required actions include:

1. **Test integration** – AGI runs functional tests with each of its daily software builds. Incorporating the functional calls and call sequences of the developed system into the test procedures ensures the detection and correction of potential integration issues prior to the release of a software upgrade.

2. **Pre-release testing** – Potential issues related to specific user-interface work flows or not-yet-integrated functionality are difficult to detect via automated testing. Incorporation of pre-released software into the developer's or integrator's update processes further mitigates integration risks.

Summary

This paper presents a methodology for comparing software-development options to deliver capabilities currently present in AGI's product line. The alternatives included: 1) the integration of AGI's software, 2) reusing and integrating "freeware" and 3) custom development.   We used a standard cost-analysis model, COCOMO, to determine the development and life-cycle cost estimates, and to evaluate the relative development timelines. For the hypothetical case analyzed here, the option of integrating AGI's software represented the lowest-cost and fastest-turnaround alternative. Furthermore, a qualitative risk analysis indicated that AGI's software can also be the lowest-risk alternative. Key findings include:

1. **Lowest life-cycle costs** – Even for the worst-case licensing fees, AGI's software is less expensive than alternatives when evaluated during a five-year life cycle.

2. **Minimized cost and schedule uncertainty** – The cost model illustrates that the AGI solution delivers the lowest cost and schedule uncertainty due to the reduced labor requirements for software integration. Findings are supported by documented programs that have implement AGI software into their solutions.

3. **Reduced risk due to licensed software** – A timeline analysis also shows that AGI's licensed software delivers advantages for cost containment and risk mitigation as the licensing fees may be linked to program milestones or timelines. Comparable options do not exist for other alternatives.

4. **Lowest risk for iterative development** – With iterative development, functional and integration requirements evolve over time. Because AGI's software delivers capabilities for downstream requirements in advance, and because of the degree of built-in interoperability and multiple software form factors, programs experience less downstream cost and greater integration flexibility. Quantitative analysis of cost and schedule impacts bears this out when compared against custom-developed alternatives.

5. **Reduced risk for software sizing** – One of the values of this analysis is that it is based on a known software size, as measured by source lines of code. Accurate software sizing is a requisite for any meaningful cost and risk analysis. When a software capability exists, it should be used as the basis for analysis like this. AGI is willing to provide information about the applicable source lines of code to support independent cost analyses for software development activities.