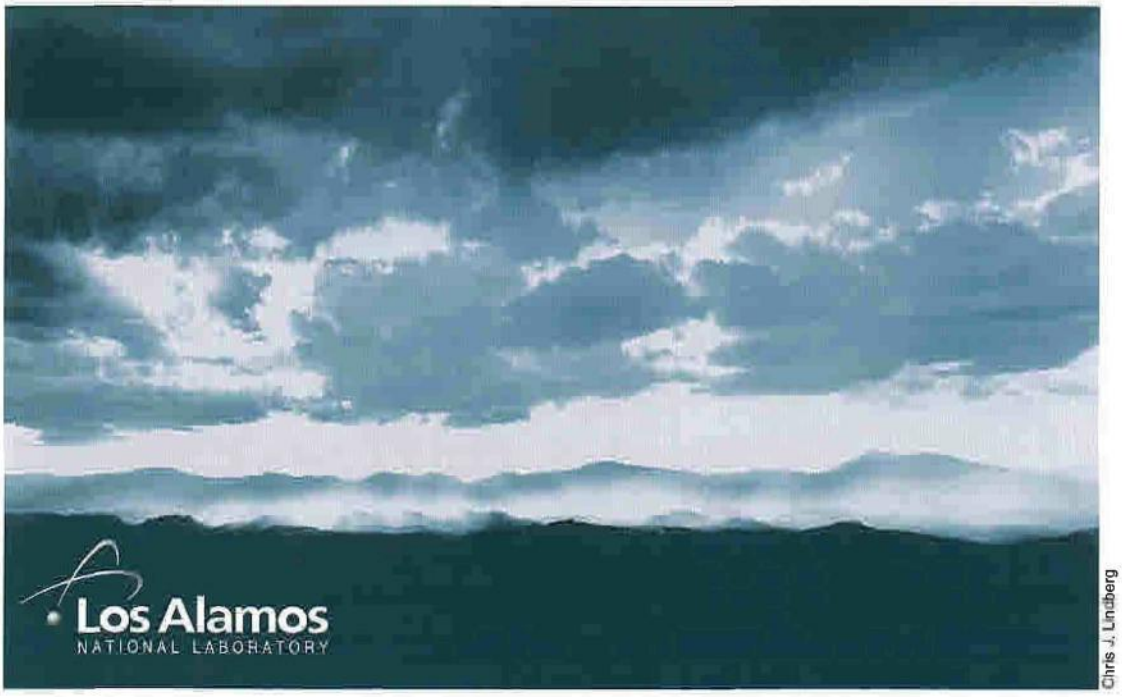
LA-UR-05-4046

USE OF COMMERCIALLY AVAILABLE SOFTWARE IN AN ATTRIBUTE MEASUREMENT SYSTEM

Duncan W. MacArthur, David S. Bracken, Louis A. Carrillo, Timothy H. Elmont, Katherine C. Frame, and Karen L. Hirsch
Los Alamos National Laboratory
Los Alamos, NM 87545

Presented at the
Institute of Nuclear Material Management
46th Annual Meeting
Phoenix, Arizona
July 10-14, 2005



This is a preprint of a paper intended for publication in a journal or proceedings. Because changes may be made before publication, this preprint is made available with the understanding that it will not be cited or reproduced without the permission of the author.

USE OF COMMERCIALLY AVAILABLE SOFTWARE IN AN ATTRIBUTE MEASUREMENT SYSTEM

Duncan MacArthur*, David Bracken*, Louis Carrillo*, Tim Elmont*,
Katherine Frame*, Karen Hirsch†, Philip Hypes*, Jozef Kuzminski*,
Robert Landry**, Douglas R. Mayo*, Morag Smith*, Duc Vo*
*Safeguards Science and Technology, Nuclear Nonproliferation Division
†Advanced Nuclear Technology, Applied Physics Division
**Safeguards Systems, Nuclear Nonproliferation Division
Los Alamos National Laboratory, Los Alamos, NM 87545

Abstract

A major issue in international safeguards of nuclear materials is the ability to verify that processes and materials in nuclear facilities are consistent with declaration without revealing sensitive information. An attribute measurement system (AMS) is a non-destructive assay (NDA) system that utilizes an information barrier to protect potentially sensitive information about the measurement item. A key component is the software utilized for operator interface, data collection, analysis, and attribute determination, as well as the operating system under which they are implemented. Historically, custom software has been used almost exclusively in transparency applications, and it is unavoidable that some amount of custom software is needed. The focus of this paper is to explore the extent to which commercially available software may be used and the relative merits.

Introduction

We consider the relative merits of using commercial, off-the-shelf (COTS) software versus the traditional use of custom-designed software in an attribute measurement system (AMS). An AMS is a nondestructive assay (NDA) system that utilizes an information barrier (IB) to protect potentially sensitive information about the measured item[1][2]. A concept of operations for an AMS is described in other documents[3].

One or more computing modules are needed to provide an interface between detector hardware and the user, as well as to analyze data (Figure 1). The operating system and application software installed in the computing module(s) will be used to interpret operator input, interface with the detector hardware, analyze data, verify that an item meets the declared attributes, and indicate the system status to the operator. It is unavoidable that some custom software will be required in any AMS, particularly for attribute comparison and the user interface.

This paper explores the ramifications of using COTS software whenever possible as opposed to the exclusive use of custom-designed software in the computing module(s). Here we will focus on the software used for the detector hardware interface and data analysis as well as on the operating system under which this software runs.

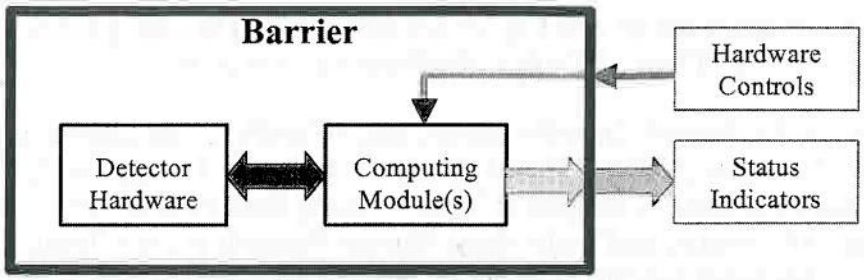


Figure 1. Simplified schematic representation of an AMS system. Detector hardware includes both radiation detectors and their associated analyzing electronics. One or more computing modules are used to prompt the detector(s) to start collecting data, interpret the data, compare it to the attribute thresholds or ranges, and send an unambiguous signal to the status indicator panel. The hardware control panel and status indicators provide the interface between both the user and the AMS. The information barrier is shown in blue, modules containing potentially sensitive information are red, modules accessible outside of the barrier are green, and wiring that is protected by virtue of its location is orange.

The Options

All AMSs require some custom software. This software is necessary both to run a command script and to compare the results of the data analysis to the acceptable range for each of the attributes. In either software option, a simple "hardware only" interface is employed to allow the operator to start data acquisition. The remainder of this paper focuses on the data acquisition and analysis software.

Option 1: Custom-Designed Software

An AMS incorporating custom-designed software would use a source-code-available operating system (e.g. Linux or ROM-DOS™), compiler, detector hardware interface and analysis software (e.g. NMC, Pu600). This has been the traditional approach used in transparency demonstrations[4]. A block diagram illustrating this approach is shown in Figure 2. Strictly speaking, the operating system used in this approach is not necessarily "custom," however, the availability of the source codes used allows any portion of the system to be modified in order eliminate extraneous functionality.

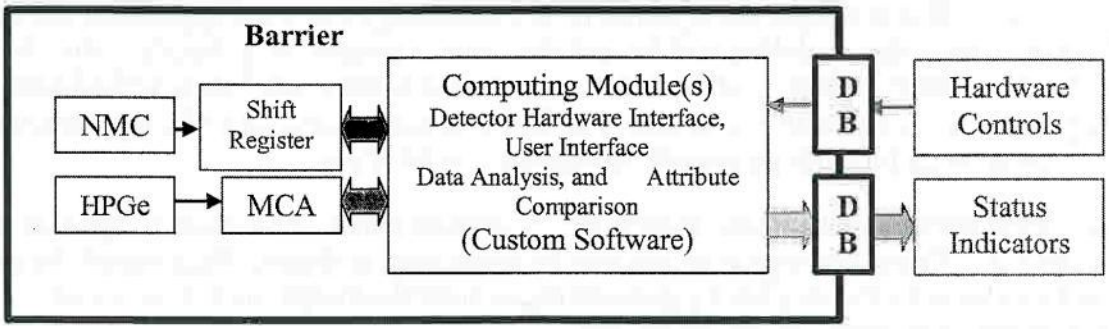


Figure 2. A simple representation of how custom software might be implemented in an AMS. The information barrier components (including two data barriers) are shown in blue, modules containing potentially sensitive information are red, modules accessible outside of the barrier are green, and wiring that is protected by virtue of its location is orange.

Option 2: COTS Software

A measurement system using industry-standard COTS software would include a high-level operating system (e.g., Windows-XP) and would run commercially available NDA data acquisition and analysis software (e.g., INCC, Maestro, FRAM, or Genie 2000). This option, illustrated in Figure 3. depends as much as possible on COTS software, using minimal command scripting to direct the COTS software.

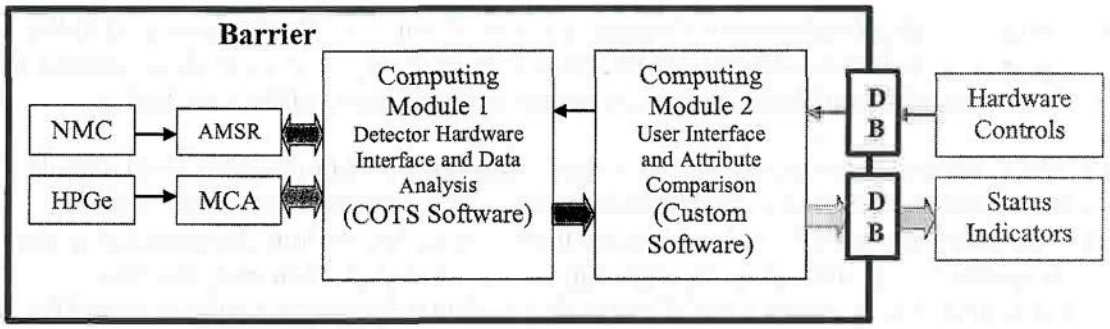


Figure 3. An implementation of an AMS using COTS software for the detector interface and data analysis of both neutron and gamma signals. The COTS software is quarantined within its own computer(s). The information barrier components (including two data barriers) are shown in blue, modules containing potentially sensitive information are red, modules accessible outside of the barrier are green, and wiring that is protected by virtue of its location is orange.

Requirements and Issues

At the highest level, there are two sets of concerns that the AMS must satisfy:

(1) The host facility must be able to certify the AMS for operation within that facility. This will include verifying that sensitive information is protected and that the AMS meets all applicable safety and security requirements. Sensitive information is well defined and, therefore, the security concerns of the host are fairly well defined. Safety considerations, although critical, are beyond the scope of this document.

(2) The inspecting party must be able to draw independent conclusions from the results generated by the AMS. Confidence in the authenticity of the results can be harder to assess, and historically, certifying a system is more straightforward than confidence building. The approach to confidence building has been to take actions that will increase confidence to as high a level as possible while still meeting the host's certification requirements.

In order to simultaneously satisfy both concerns, there are several issues that any software option will need to address:

- All parties must be able to trust the results.
- The system must be reliable.
- The required resources must be available.

The remainder of this paper examines these issues and how they relate to each of the software design options.

All parties must be able to trust the results of the attribute comparison.

In general, this means that the software must meet the computational requirements needed to provide an unambiguous comparison with the agreed-upon thresholds. All parties must be confident that there are no hidden or unknown functionalities or data storage capabilities.

The inspecting party may require more stringent scrutiny of software that is developed in the host country in order to build confidence in the results. Conversely, it must be demonstrated that software developed outside of the host country preserves the security of the host facility.

In either case, if custom software (option 1) is used, trust can be established by supplying all parties with the complete source code (although it is possible that the source code will be sufficiently complex to render complete scrutiny impractical). In addition, the availability of the source code means the possibility of exposure can act as a deterrent. However, the "bus" communication required by modern MCA's and shift registers does open up the potential for modification to custom software by changing the detector settings.

In option 2, the use of a widely available operating system, such as Windows XP, lends itself to the use of blind buys and random selection as means of establishing trust in the software. For example, multiple copies of standard, shrink-wrapped packages of the operating system can be purchased from any vendor. One copy can be used for installation and the other copies can be given to all involved parties for whatever level of examination they require.

The use of blind buys and random selection can also apply to the choice of detector hardware interface and data analysis COTS software, such as Maestro, PC-FRAM, or INCC. While much of this standard software was written in the US, this software has been used and rigorously tested worldwide. During the years that this software has undergone quality assurance, commercialization, and years of field use, any hidden or otherwise erroneous functionality should have been discovered. Because the software is licensed to commercial vendors whose prime motive is making a profit, there is strong incentive for releasing an unflawed product.

The system must be reliable.

System reliability is particularly important in transparency regimes. Significant numbers of people travel long distances to participate in these measurements. These measurement visits will have to be scheduled many months in advance, limiting operational flexibility. Poor system reliability will, at the very least, make inefficient use of everyone's time. At worst, poor reliability could lead to reevaluation of the technical viability of the entire program. Although any measurement system can fail, every effort must be made to ensure that failures in the system are as infrequent and as easily resolved as possible.

System reliability is defined as the probability that the system will accurately perform attribute measurements on any anticipated or reasonably similar item under a set of defined conditions for a defined duration. The exact reliability standard that will become the design target will be determined by the cost of developing a system capable of meeting the design goal and by how well the system is expected to tolerate items that are not reasonably similar to the anticipated

items. If the AMS simply responds with a system error report when the system cannot accurately perform an attribute measurement on an item that is different from what is anticipated, a lower reliability value can be tolerated. If the difference leads to a system crash that requires the rebooting of the system—or worse, requires expert diagnosis—the system's reliability will need to be much higher.

In option 1, the software would be developed specifically for a particular Concept of Operations and for the types of items that were predetermined under a particular agreement. Development, integration, and testing costs will increase proportionately with increased reliability testing and measurement flexibility. In order to obtain the best possible reliability and flexibility, custom software would have to be designed and tested to the same level as a commercial product that was intended for general NDA. This would be a significant project in its own right; the development of such software has typically involved millions of dollars and years to decades of design, testing, and quality assurance work. This level of quality assurance is not practical within the framework of an AMS project. Custom software would be developed to be as flexible as possible, within the time and budget constraints, but its flexibility could not be expected to approach that of commercially available software. In addition, the software must be designed to fail gracefully, instead of crashing. Yet, this graceful failure mode could only be proven within the confines of time and the budget that was available for testing. Testing would also be complicated by the need to test items that are at or beyond the limits of the agreement.

Option 2 makes use of available COTS software. The COTS software packages that are appropriate for an AMS have been in common use worldwide for many years, and broad testing and use demonstrate the reliability of the software. Although not all COTS software has the same level of reliability, carefully selected commercial software will generally have greater reliability than custom-developed software that is application specific. Using COTS software reduces the developers' control over whether the system fails gracefully or catastrophically, but codes that regularly have catastrophic failures either get upgraded or fail to be commercially viable in the long run.

The required resources must be available.

The success of an AMS project depends heavily on the availability of the required resources for development. These resources include both funding and personnel time. Additionally, they should include not only the development costs for each individual software component but also the costs to integrate the components into a system, test the full system, document the software components, and inspect the system (during confidence building and certification). This requirement is closely related to the above-mentioned requirements for system reliability. A user's guide and other software documentation will undoubtedly be required for either software option.

Option 1, the use of custom-designed software, has been prone to application and system crashes in the past. The frequency and seriousness of these crashes are typically inversely proportional to the amount of time and money spent developing and testing the software components. Thus, the costs for developing reliable custom software modules are highest for this option. However, after the software modules are completed, the costs for the integration of the individual components would be the lower of the two options because each component has been specifically designed for an AMS. In our discussions, we have assumed that the required resources are available.

Unfortunately, if the software author(s) has retired in the interim or otherwise moved on, this assumption may not hold true.

The costs associated with testing the completed system and writing documentation for the individual software components are highest in option 1. As with the development of the software, more time and money spent on testing and documentation result in a proportionately better system. Since there is no other use for the software, there are no existing test results or documentation that can be substituted to save time or money.

The strengths of option 1 are that the source code can be made available for inspection by all parties and it has no extraneous functions. Certification and confidence building can proceed from the inspection of the code. The time and resources required to develop, adequately test, document, and inspect the modules, may be prohibitively high.

Option 2 uses industry-standard COTS software and is, therefore, less costly and time consuming to develop. The software modules can be expected to work reliably "out of the box." Costs associated with integration of the individual modules for the COTS software will be higher; however, based on prior experience scripting COTS software, the additional integration costs will be small compared to the cost of developing the entire data acquisition and analysis software.

If testing the COTS software is limited to its functionality used within the AMS, testing costs will be lower for this option. Testing only the functions used by the system should be acceptable if they are accompanied by administrative, engineering, and software controls to prevent access to unused software functionality. Any additional testing costs should also be minimal for this option because of the manufacturer's prior field testing and its use in applications unrelated to AMSs. The functionality of the complete system will still need to be tested at the same level as that of any other option.

The individual components of the COTS software are already well documented, thus reducing the costs to document a complete AMS system built from COTS modules. The only documentation costs will be those associated with the small amount of custom software and with documenting the complete system. However, these costs would also be incurred with option 1.

The relative resource requirements for testing also apply to the resource requirements for inspection. If inspection is only required for the software functions that are actually used, costs will be minimized. It is hoped that the need for inspection of each individual software component can be reduced by building confidence early in the process, when the software is purchased, by using blind buys and random selection. Any extraneous functionality of the software should be less of an issue since the rest of the software will have been developed for more generalized (and widely documented) uses. Additionally, since hundreds or thousands of others use all parts of the software, any unintended or hidden functionality would most likely have been found.

Finally, future support of custom-designed software relies heavily on access to the developers of the code. Thus, support may be limited because the number of these developers is limited and they are often not available to help support an aging code. The simpler this custom-designed software is, the more likely it is that any problems with the system can be resolved.

Conclusions

Option 1, a purely custom-designed (open source) solution, is the traditional approach. It relies on the deterrent effect of making the source code available to all parties. This deterrent is limited, however, by the feasibility of inspecting all of the source code necessary to implement the detector hardware interface and data analysis software as well as that of the required operating system. Furthermore, developing a reliable, thoroughly tested and well-documented code requires a substantial investment of resources, which may not be available. While this option has been used in previous attribute measurement systems[4[5]], the most substantial argument for its use may be that it can be certified for use on sensitive items by the host country because of adequate administrative and engineering controls. However, neither the ability to authenticate such a system nor its reliability has been fully demonstrated.

Option 2 relies more heavily on COTS software, but it has not yet been tried. It relies on the wide availability of the proposed COTS software to make it unlikely that "backdoors" have been inserted or exploited by either side. The reliability of the system is supported by extensive field testing by users in a wide range of activities as well as by the available documentation and vendor support. By using existing, well-tested software packages, this option will be less costly in time and money while still providing the required level of data protection.

A potential concern with using option 2 is the role that the US weapons laboratories have had in the development of the detector hardware interface and data analysis software. An alternative to using a COTS analysis code is to develop the data acquisition and analysis software in a very high-level programming language such as LabVIEW™ or Matlab. This would help avoid the problem of relying on closed-source software developed at the US weapons laboratories without requiring that developers start from scratch with respect to the data acquisition and analysis codes. A simple program that essentially connects a few commercial library functions could be loaded into the AMS computer module by the host under inspector supervision. However, the library functions that are needed are not currently available commercially, making a simple implementation of the gamma analysis algorithm improbable at this time.

The competing needs of software that can be trusted, is reliable, and has acceptable resource requirements has led us to believe that rejecting all commercial, closed-source software out of hand is not reasonable. For the LANL AMS project, in portions of the system where the software is not required to be the primary provider of information protection, preference is given to commercial software on the basis of its demonstrated reliability and cost effectiveness.

References

1. D. Bracken, L. Carrillo, T. Elmont, K. Frame, K. Hirsch, P. Hypes, J. Kuzminski, R. Landry, D. MacArthur, D. Mayo, M. Smith, D. Vo "Attribute Measurement System: General Technical Requirements," Los Alamos National Laboratory Report LA-CP-05-0638, 2005.

2. J. Shergur, D. Bracken, L. Carrillo, T. Elmont, K. Frame, K. Hirsch, P. Hypes, J. Kuzminski, R. Landry, D. MacArthur, D. Mayo, M. Smith, D. Vo, "An Overview of the Design of a Next Generation Attribute Measurement System," presented at the INMM 46th Annual Meeting, Phoenix, AZ, July 11-14, 2005.

3. T. Elmont, D. Bracken, L. Carrillo, K. Frame, K. Hirsch, P. Hypes, J. Kuzminski, R. Landry, D. MacArthur, D. Mayo, M. Smith, D. Vo "Attribute Measurement System: Concept of Operations," Los Alamos National Laboratory Report LA-CP-05-0634, 2005.

4. J. K. Wolford, Jr, B. D. Geelhood, V.A. Hamilton, J. Ingraham, D.W. MacArthur, D.J. Mitchell, J.A. Mullens, P.E. Vanier, G.K. White, R. Whiteson, (members of the Authentication Taskforce, Software Working Group) "Software Authentication," presented at the *INMM 42nd Annual Meeting*, Indian Wells, CA, July 15-19, 2001.

5. D.W. MacArthur etal., "Fissile Material Transparancy Technology Demo", March 17, 2001, Los Alamos National Laboratory , Lawrence Livermore National Laboratory, and Pacific Northwest National Laboratory. May 25, 2005 <http://www-safeguards.lanl.gov/FMTT/index_main.htm>, Los Alamos National Laboratory Report LA-UR-01-1091, 2005.