An Implementation of ARM

Larry D. Wittorff
The Boeing Company

An open API for Application Response-time Measurement (ARM) was announced on June 24, 1996. This paper will show an implementation of ARM that is working and providing information that, in the author’s opinion, would not have been possible without it. A group of three application providers, a system management vendor, and the customer agreed to use ARM as the standard to instrument the applications and provide the source for service level information. Here is how it was done along with some things that should be considered when using the ARM specification.

1. Introduction

In late 1993, Boeing Commercial Airplanes initiated a process improvement activity to re-engineer the production of airplanes called DCAC/MRM (Define and Control Airplane Configuration / Manufacturing Resource Management).

It is important to note that the introduction of computing systems or hardware was not the objective of this initiative. The main objective was to get the configuration and production of airplanes streamlined to reduce cost, cycle time, and defects. The objective for the computing systems was to support the timely implementation of these process improvements.

The subject of this paper is how ARM was used to monitor this change and to provide service level information on the critical processes. The outcome of which was reduced cycle time. In the airline production industry cycle time is defined, as the time required producing an airplane.

2. Background

A decision was made to use COTS (Commercial Off The Shelf) products when possible and to follow the corporate direction of implementation of new applications on open computing platforms. Three application providers were chosen as well as the hardware platforms and the software to manage this environment. These providers were brought in as “partners” in change. Dedicated resources were assigned by each of these providers. These teams worked along with Boeing to make recommendations on how to achieve the system objectives.

One of the objectives was a service level agreement with the customer. A service level agreement existed for the proprietary systems that were being replaced and the requirement was considered to be even more important in the new environment. However, the single vendor architecture didn’t exist in the new systems.

Within the project team, interfaces were being designed specifically for this purpose when ARM was announced in 1996. ARM was determined to be the standard interface that would be used in the application code and picked up by the system management software. ARM would be the link between the “partners” to capture the response time taken to perform certain functions.

3. How ARM works

ARM, Application Response-time Measurement, is a specification, which if followed, allows applications to make very basic calls to stored routines which do not have to be coded. The application only has to adhere to the specific parameters allowed. These parameters provide a means to identify the application and transaction making the call, indicate when the process is starting, and when it has ended. In addition, there are other parameters that will pass data about the application or any application state information that might be available but the basic information is to describe, start, and stop.

The application includes calls to stored library routines that do not have to be written but the routines do need to exist in order to run. There are basically two choices: The application can be compiled with a set of “stub” routines that can be downloaded and are supplied with the ARM Software Development Kit or
system management vendors can provide these routines.

This allows application developers, whether in-house or outside, to include ARM calls and instrument their application without having to choose a specific system management vendor or having to have one at all if the “stub” routines are in place.

The following illustration shows some of these basic concepts. There are many more capabilities in ARM but its strength comes from the simple things it does.

**Figure A.**

**How does ARM work?**

![Diagram showing the flow of ARM operations](image)

Applications code and compile with the free “stub” procedures downloaded with the ARM Software Development Kit.

- **Application**
  - Describe
  - Start
  - Stop

- **ARM Standard Procedures**

- **System Management Software**

System Management Software Vendors replace the Standard procedures with ones which will feed their product but still look the same to the application.

Detail information about how the ARM specification works can be obtained at the following web sites.

http://cmg.org/regions/cmgarmw/index.html

http://www.opengroup.org/

**4. History of ARM**

ARM 1.0 was developed by Tivoli and Hewlett-Packard and released in June of 1996. An ARM Working Group was formed within CMG (Computer Measurement Group) and was made up of the ARM 1.0 partners as well as other system management software vendors and end-users interested in the promotion and advancement of ARM. This group issued ARM 2.0 in 1997.

Both ARM 1 and 2 were written in C. ARM 2.0 was also approved and published as a standard by The Open Group.

ARM 3.0 is an enhanced version written in Java and will be primarily published through The Open Group. There is a C version of ARM (with the 3.0 enhancements included) being worked on but not yet prototyped.

**5. Including ARM in applications**

Deciding what to instrument in an application is very important. If the application is developed in-house or on contract then the customer has more control on what is built into the application. Even when the application is being developed in-house, it is very difficult to plan and anticipate the eventual use of the
response time data and what might be the most critical transactions to the customer during day-to-day use.

When the application is being developed for customers not yet identified, it is even more difficult to decide what to instrument. Consider this, it is much easier to ignore unneeded data than to get data that doesn’t exist. Therefore, more data is better than not enough. As with anything though, instrumentation can be overdone.

The most important thing is to have some sort of method or reason on how the application is to be instrumented. It should be applied consistently throughout the application. In developing that method, try to anticipate the use of the data. Instrumentation of an application for debug purposes is very different from instrumentation for the purpose of a service level agreement.

One of the primary things the developer must watch out for is a break in the transaction for user intervention. If the main goal is to record the amount of time the computing environment spends processing a transaction, then make sure the time for a user to respond to a dialogue box or any kind of a pause for user interaction is recorded separately. The entire time to complete a business process might be a very valuable piece of information but the processing and user time should be recorded individually.

Another case to be considered is an error message from invalid user input or some other situation. ARM specifications provide a parameter to indicate whether the transaction finished normally or not. However, the developer still needs to identify the possible paths that could end a transaction and make sure they are covered. This is a place where having a consistent methodology will help significantly when attempting to use the data generated by the ARM calls.

The primary use of the ARM data in our application was to monitor the service levels of critical transactions in the system. To do this, the important business processes had to be matched up with the data available from the applications use of ARM. There was a huge effort to go back through the data to find the sum of transactions that would make up a business process.

Any of the ARM specifications beyond ARM 1.0 provide for a correlator that is generated by the ARM agent when the application indicates the start of a transaction. This correlator can be passed on from application to application to identify all the work necessary to complete a user request. These linkages are assembled by whatever ARM agent is chosen to “catch the calls”.

Our application was coded to the ARM 1.0 specification or the correlators were not passed. That is one of the things that made tying all the transactions involved in a business process together for service level reporting very difficult. Also, the same application function was used in several business processes along with other specific functions for that process. This made it very difficult to determine how to use the data from that common function in the service level actuals.

A clear purpose for the data, a methodology for instrumentation of the application and documentation of what was actually instrumented are very helpful when reaching the point of actually using the data.

6. Possible stopping point

Once the application has the ARM calls included, the key ingredient is in place. There is more to do and some costs to consider but the source data is in the bank. Remember also that there are no actual requirements beyond this point. There are no immediate benefits either, other than the application is set up to generate service level data for use in the future. As long as the application has the ARM calls in place and the stub procedures from the free ARM Software Development Kit are included, no additional work is necessary to the application program.

An important point for software developers: ARM can be built into the application and offered on the market. Customers that do want the ARM data can decide which system management software to buy in order to capture that data. Other customers may not care to go to that effort or expense. The original software product is absolutely useable in either case.

7. Software to catch the calls

If the true benefit of the ARMed application is to be realized, a decision will need to be made on a system management package. Technically it is feasible to write the procedures on the other end of the ARM calls; however there are products on the market to do this. Justifying the time and expense to re-create, maintain, and enhance this capability would be very difficult. If coding these capabilities is being considered, at least look at what is already available in the market before making a final decision.

There should be an overall list of requirements used to evaluate any decision to acquire a System Management Package. Make sure that the requirement to support ARM, and which versions of ARM are supported, is included in that list.

There are companies that market products that work with applications that are ARMed. One starting point
might be to contact the vendors who originally
developed ARM or one of the members of the ARM
Working Group. A list of its members may be found at:
http://cmg.org/regions/cmgarmw/members.html

The ARM Working Group was organized with the
sanction and cooperation of CMG. Their products
could at least be considered for the extent that they
handle ARMed applications

The capability to support ARM is just one of the criteria
for a system management product; and even though
one may handle and provide support for ARM data,
the best one will be the family of products that best
meets the overall needs.

Most companies have already chosen a primary
system management vendor for use over as many
systems as possible. In this case, the ARM capability
of that product needs to be evaluated to insure that it
is at least adequate. If not, this is an area that will
require a lot more research. The conflicts and
architecture of two or more system management
products and their agents would have to be thoroughly
understood. There might even be conflicts of multiple
agents to catch the ARM calls. That situation would
require careful planning and considerations to
implement.

8. The system management tool

Once a product has been chosen, the documentation
will be available for installing and setting up to handle
the ARMed applications properly.

Remember that ARM allows this complete flexibility to
choose whatever system management product is the
best fit for the specific environment being considered.

ARM allows all applications to be instrumented with
the same basic requirements that system
management products use. There are some amazing
things being done with this data in real time as well as
over time and this response time data is also being
integrated with other analysis tools.

9. Archiving the data

Even with all the graphing, reporting, and real time drill
down capability that the system management products
give, there always seems to be specific summaries
and formats that are required. There were a number
of these requirements in our application.

There was a process already set up to extract the
performance data accumulated by the system
management package into the data archiving product.
There was some additional effort to include the ARM
data. Once that was complete, access to response
time data was available through the same methods
and for the same timeframes that other performance
data was.

10. Information and capabilities that ARM enables

The following example shows one possible use of the
ARM data. It starts at the most summarized level
provided and then shows examples of further detail to
the point that transactions can be viewed in real time,
as they happen. This doesn't have to be done in this
order at all. The particular system management tool
used in our application allows alarms in real time on
certain ARM triggers so; analysis would not have to
wait until a chart is shown with an anomaly.

This is a little difficult though. Alarming on the
response time of one single transaction can get pretty
active. Even in some of the most controlled
environments, there are still a significant number of
transactions that hang or just take a long time for
whatever reason. There may be a lot of alarms to
answer but there will probably a lot learned by
being able to dig right into these situations.
11. Examples

Figure B. Four Week Service Level Summary

Wichita SLA Summary Report
Weekly Demonstrated Performance by Transaction
Code Instrumentation Data

<table>
<thead>
<tr>
<th>Transaction ID</th>
<th>Scenario/Transaction Name</th>
<th>26APR01</th>
<th>03MAY01</th>
<th>10MAY01</th>
<th>17MAY01</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCS001</td>
<td>Open NCR - Standard</td>
<td>98%</td>
<td>98%</td>
<td>97%</td>
<td>98%</td>
</tr>
<tr>
<td>NCS002</td>
<td>Import NCM Graphics</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>NCS003</td>
<td>Route Record Next Queue</td>
<td>99%</td>
<td>99%</td>
<td>99%</td>
<td>100%</td>
</tr>
<tr>
<td>NCS004</td>
<td>Retrieve NCM Work Queue</td>
<td>96%</td>
<td>95%</td>
<td>94%</td>
<td>98%</td>
</tr>
<tr>
<td>NCS005</td>
<td>Open Order - Update - Standard</td>
<td>92%</td>
<td>90%</td>
<td>90%</td>
<td>95%</td>
</tr>
<tr>
<td>NCS006</td>
<td>Open CAR - Update</td>
<td>97%</td>
<td>97%</td>
<td>94%</td>
<td>98%</td>
</tr>
<tr>
<td>NCS007</td>
<td>Open Action Plan - Update</td>
<td>86%</td>
<td>85%</td>
<td>88%</td>
<td>92%</td>
</tr>
<tr>
<td>NCS009</td>
<td>Open Code Page - Update</td>
<td>99%</td>
<td>99%</td>
<td>99%</td>
<td>99%</td>
</tr>
</tbody>
</table>

This is part of the main service level summary. It shows one “percent under target” value for each critical transaction at a location for a week along with the four week history of this metric. The green background (if black and white, values over 90%) means that at least 90% of the transactions met the service level established.
12. Conclusion

In the author’s opinion, ARM has provided response time data and performance tools that would not have been achievable without it.

More ARMed applications are needed.

If applications are being developed in-house or contracted for, the capability is within the company to convince the decision-makers to include ARM.

If COTS applications that have already been developed are being considered, ARM can be part of the decision criteria. It may not be important enough to swing the decision away from a product that doesn’t include ARM but at least it could be considered.

The situation could be improved if customers would push the requirement hard enough to where COTS developers took it for granted that ARM would need to be built into their applications. Then customers could make some consideration in their decision over who did it the best and what level of service the application has been able to provide in similar situations.

I had always thought that ARM could be considered a success when some day, a product would be brought in that had already been ARMed without having to specifically negotiate it with the vendor. Then I found out, it has already happened.
There are ARMed applications out there and they would become more prevalent if more customers would look for them and ask for the capability that ARM provides.

13. Summary

1) 1993 – A huge effort was undertaken to re-engineer the processes and systems used to manufacture airplanes at Boeing.
2) Application and hardware vendors were chosen and considered “partners” in change.
3) The ARM specification was decided on as the standard to provide the needed service level information.
4) Care needs to be taken on how ARM is built into the application.
a) User think time or dialog boxes should not be included or should at least be separated.
b) Error messages should be provided for and handled consistently.
c) Whatever ARM data is produced is the source for whatever else is done.
5) Once an application is ARMed, any of the system management products can be chosen that support the ARM standard and used to get amazing “out of the box” results.
6) Tailored reports, charts, or drill down capability can be developed to meet specific requirements.
7) Look for ARMed applications. It will help us all.

14. Acknowledgements

Anybody in The Boeing Company or the Boeing partners that had anything to do with the decision to use ARM, implement it, or work with the data produced.

Mark Johnson of Tivoli and Ron Carelli of Hewlett-Packard who have represented their company’s and our industry’s considerable efforts in ARM for as long as I’ve known anything about it.

15. Trademarks

To the best of my knowledge, the following are trademarks of companies referenced in this material.

CMG is a trademark of Computer Measurement Group, Inc.
Hewlett-Packard and HP are trademarks of Hewlett-Packard Company
Tivoli is a trademark of IBM Corporation
Netscape is a trademark of Netscape Communications Corporation
Java is a trademark of Sun Microsystems, Inc
SAS is a trademark of SAS Institute Inc.