

MES Augmentation- An Alternative to MES Replacement in a Semiconductor Fab

Nancy Glenn
Delphi Automotive Systems
Kokomo, IN 46904-9005

David Braun
R&R Visual Systems
Rochester, IN

Abstract:

Imagine if you can-a manufacturing environment with a well engrained legacy MES. Now add on increased pressure to do prior step verification, recipe verification and download, and an ever increasing push for scrap reduction and process control. Already familiar to some of you ? Our solution? Lacking the resources to replace our MES, we took the approach of augmenting. We built an infrastructure framework which could bridge the gaps between equipment, operator, MES and management requirements. Stepping outside some of the current manufacturing trends, we architected an n-tiered, Java-based system which provides us high availability, flexibility and scalability. We will address the pros and cons of this choice, and the implications for other manufacturing environments. No new technology comes without an infrastructure cost. We will also discuss the layers of IS/IT glue necessary to tie together existing workflow, multiple databases, new and legacy data systems while still presenting the floor operator with a common interface across all types of equipment. No technology solution can be successful without the cooperation of operations, manufacturing engineering and IS/IT staff. An implementation in an operational Semiconductor Fab with a unionized workforce presented a multitude of challenges and opportunities, and we will share the lessons learned -both positive and painful- on this project from the people aspect of implementing new technologies.

Introduction

In any manufacturing setting, the decision to implement new technology can be a difficult one. In the current mentality of cost cutting, any new system will have to quickly justify itself financially and significantly contribute to the enterprise. Faced with the pressures of manufacturing in an internet-based electronic business environment, small and medium sized manufacturers are looking to add Manufacturing Execution Systems (MESs) in order to compete. Recent advances in technology have larger manufacturers which may have implemented MESs as long as 20 years ago rethinking the capabilities of their systems. MESs are once again a hot implementation in Manufacturing IT. In a new manufacturing facility, once the decision to use an MES has been made, it is merely a choice between competitors and finding the right software package to fit the facility. But what about a facility that made the leap 10 or 15 years ago? The system they installed at that time, although possibly leading edge technology *then*, is most likely outdated and not serving all of the needs of the enterprise. Is it time to replace the system, or is there some way to eek a few more years out of the current investment ? Fifteen years ago having a daily report on yesterday's wip was considered state of the art and assisted managers in

decision making. Today knowing yesterday's WIP is about 24 hours too late. With the move to lean manufacturing, or with a JIT (Just In Time) pull from suppliers in place, managers often need a real time view of their facility. They need advanced scheduling and dispatch with a view of machine up and down time to track and move the manufacturing flow around bottlenecks. Often, they also need to provide visibility to their customers, or to the corporate boardroom. They may also need to extend data collection out to suppliers for quality feedback, or to subcontractors to track offsite manufacturing. Some manufacturing facilities have moved to a strong 24X7 schedule and need systems that are fault tolerant and fail over cleanly. Many of the legacy MES systems did not provide the flexibility or stability demanded by today's enterprises. In these cases, where the technology demand exceeds the capability of the current system, a decision must be made to replace or to augment the MES. Many add on packages can be bought that provide some of the additional functionality, but tying them together can be a tricky and costly process.

Evaluating Your MES and your business needs

As you work through the decision to buy new or build on to what you have, there are several points you should consider.

What do you need out of your MES ?

Before you start any technology implementation, it is important to have a clear definition of your goals. What are your biggest trouble areas? What information do you need to run your enterprise and in what format? Where will your cost justifications come from? Do you need to reduce scrap? Do you need to increase throughput? Better manage recipes on equipment? Are you looking to track static routes, or to dynamically assign product to routes according to changing bottlenecks?

Do you need to change the core functionality of the system ?

If you already have a system in place, does it- at the core- do what you need? If not, building on to your existing system will only expand your problems, not solve them. If your legacy MES relies on statically defined routes to dispatch, and you want to do real time dispatching it does not matter if you add an equipment monitoring function, the dispatch will still be static. If you have new business requirements for traceability and require that your systems all be transactionally safe, but your legacy MES is a socket and string based technology, no amount of enhancement is going to change that core problem issue. There are, in fact, times to move on and invest in a new system.

How are you growing?

It is important to have a good sense of where the expected areas of growth are, both as a business and as an IT Infrastructure. What you don't want to end up with is an MES with so many custom patches and add-ons and custom fixes that it becomes unmanageable. Nor do you want to plan a custom module around an Oracle database, only to find out that IS is desupporting Oracle at your site in 6 months. Be sure that your code will not have to be "fixed" in 12 months because of a planned business process change that was not communicated downstream.

How old is your legacy system ?

Although adding code to your MES is less costly than replacing the entire system, it is not a small investment. Be sure you are comfortable that you will be able to get about another 5 years of life out of your MES, before making an investment of this size. Be sure to consider both hardware and software issues. Look not only at the MES software itself, but does it require almost obsolete hardware or an OC that is about to be desupported by the manufacturer? What is the message bus or database infrastructure used in the MES? Do those companies still support the revision used in your system? Don't spend several hundred thousand dollars on a system that can not be supported in 6 months.

Understand System Connections

In order to be sure that you are making the right architectural decisions, it is important to understand the numbers, types and frequency of connection to the system today, and how those numbers are expected to change with the additional functionality. As the number of connections increases, it drives you to an architecture that uses a common bus type. It is also a good idea to separate the connections made for core functionality(product movement and tracking on the floor) from those made for auxiliary functions (reporting, equipment monitoring, etc.). This separation can be done with either software or hardware re-directs, or some combination of the two.

Supporting your changes

If you have made a careful evaluation and it makes good business and technological sense to build onto your MES, be sure you understand how you will support this new codebase. Do you have internal staff who can support this, or will you outsource the support? Can your current MES support team handle the extra work load, or are they already short staffed? In order to help this project succeed, it is critical that you have a support plan in place BEFORE you start any development work.

What is MES Augmentation?

[1]augment

\Aug*ment", v. i. To increase; to grow larger, stronger, or more intense; as, a stream augments by rain.

Source: *Webster's Revised Unabridged Dictionary*, © 1996, 1998 MICRA, Inc.[2]

Augmentation can be a double edged sword. By Webster's definition, augmenting something is to make it grow larger or stronger. If we are discussing a flood, it is hard to imagine those not going together. Can floodwaters get larger and  get stronger? But in an IT project, that is not always the case. Larger can also be like an over-inflated balloon, much more susceptible to disruption. It is important that any plans to make an existing system larger are carefully considered and implemented in a way that will avoid bringing the manufacturing floor to its knees.

Done correctly, augmenting an MES can help you to avoid vendor lock-
The traditional leaning toward floor to ceiling implementation by one vendor is currently being questioned as the impact of changes is too costly and painful in the

future. However, if you write your new code only to a specific vendor API, then you only be augmenting the impact of a vendor change. Be clear about your goals for this new custom code- will you just throw it all away 5 years from now when you *do* finally upgrade? Or will you want to migrate this functionality to your new MES? One way to evaluate this is to look at why you are augmenting. Are you adding functionality to your legacy system that is currently available in newer MESs on the market, or are you adding functionality because you can not buy it anywhere else? If you are replicating functionality that is common today, most likely you will be throwing out your custom code once you migrate MESs. Be sure to plan your architecture accordingly.

MES Augmentation Methodologies

There are several methods and levels at which augmentations can be made. Depending on the reason you are augmenting and the lifespan of your custom code, not all of these will be a best choice.

Code level changes

This is a change of the actual MES. It requires either that your MES be open source, or that your MES has an API for modifications. In some cases this can be done by in-house staff, in other cases the vendor will require that their technical staff make any modifications. This is rarely a best choice, and works best for you as an end user if the MES is modular and designed for user modifications. Common modifications at this level are things like custom screen exit functions, custom algorithms for calculations, custom data lookups and verifications. In almost all cases, it is best to avoid a case where the vendor is branching their actual code in order to make you a "custom-built" MES. If they are doing it for you, they are doing it for other customers, and if the MES is not modular enough, they will not be able to support it long term, nor will upgrades be an easy or cheap prospect. This is not a scaleable solution by nature, and if your system is growing enough to become more distributed, the complexity involved in these modifications can increase the complexity of your system significantly.

Client wrapping approach

If your client users normally pull data from many sources, you may want to consider a client wrapping approach, where your end users are running a custom application which then interfaces with many other systems and applications. This client application could be stand-alone or web based. It is an approach that puts more business logic in the client code, and if clients have to connect in a peer to peer way, it can quickly become unmanageable. This approach works well for situations where the number of clients is small. As the client base increases, or as the number of connections a client has to make increases, it can grow out of hand. This is a solution that can be custom built, or purchased.

Server wrapping approach

A server wrapping approach is one in which data and services are wrapped in a commonly available interface. This is a good approach if you have legacy systems which you would like to have incorporated together. This is a loosely coupled solution, which allows some components of the system to change without breaking the entire application. Depending on the legacy systems you are behind the server

this implementation results in a common interface to several system where the clients have no knowledge about how exactly the connections are made to the system. The advantage of this type of approach is that it gives you an opportunity to swap out the legacy system without interruption.

Common Connections Method.

The common connection approach to augmentation is similar to that of the server wrapping method but it uses a common connection between components. An example of these types of connections is Publish/Subscribe. There is a connection made to the legacy system that allows a message to be published to it. This would require code changes in the system unless it already supports it. Data for this that system is sent from the clients to the system and back. The advantage of this system is that there is a common connection language. The disadvantage is that there still is not a common data model for the systems -thus clients would have to be smart about what message to publish and would require some sort of wrapping that is also found in the server wrapping method. The big buy from this approach is that you are using a middle tier to isolate your clients from changes in backend- things like database drivers, message bus, etc. Most common applications are N tiered server solutions in which the users are isolated from changes to the back end. Putting together a framework to which the client interface connects, rather than having the client interface talk directly to the back end system allows you to change back end systems and not have to change your user interface significantly. Also isolating your business logic into the middle tier allows you to change the rules by which you run your business without redeploying or upgrading client interfaces and interrupting the manufacturing flow.

What is a framework ?

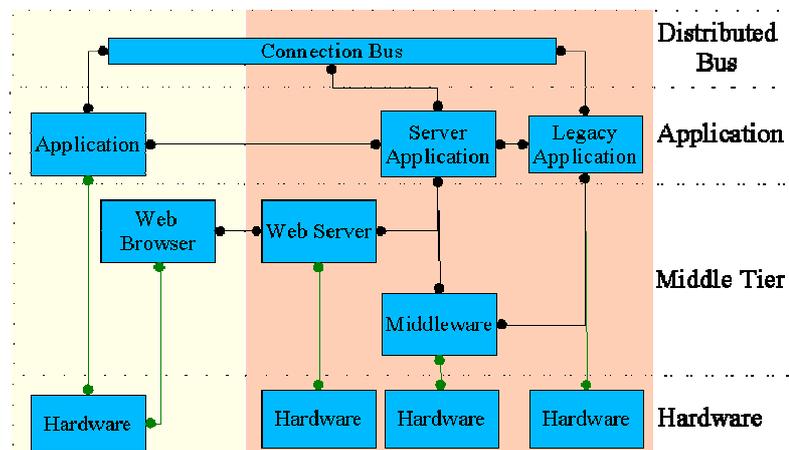
When you use hardware and software to build a technology scaffolding upon which functional modules can be "hung", you are building a framework. Be sure you are just putting in the plumbing and wiring, make it capable to plug in other software packages- *not* writing all the other add-ons. In Figure one you can see an example of how all the possible pieces of your framework could fit together. All the functionality is expensive to upkeep and maintain- there is a potential cost in programmer heads. If possible, you want to push that off to vendors designed to do software support. If there are pieces you have to build in house, try and partner with vendors who are interested in marketing and maintaining it. In most situations, your framework will have some basic required functionality. This includes the following:

database caching- allows for the most efficient use of resulting database queries . It should be noted that many application servers or middleware software packages do this already- it may not have to be coded into the framework itself.

Figure one: Framework Example

user authentication – how will you handle security ? Again, some middleware servers handle this portion of it. This is an area to be particularly sensitive to the future growth and direction of your IT department. Be sure this can be flexibly changed without re-writing the whole framework. User authentication methodologies are in a flux, with LDAP cornering an ever growing portion of the market. So even though you may be using DNS authentication currently, LDAP may be on your horizon.

client connection management- having a single API into which the client calls the middle ware, then having this distributed out inside the middle tier allows you to change how you do functionality within your middle tier. For example- database calls. Never have the client do a database call directly, they should call into a common point of the middleware and pass in a request for a database call. This interface can then remain the same, even if you change your backend database or change from a rdbms to an ordbms. If the client called directly to the database pool in the middle tier, this would have to be changed with backend changes. This directly



follows the primary principle of having a common connection point that you are isolating through interfaces. Although there are many purchasable application servers or messaging servers that provide a large portion of this infrastructure, you will have to do some adaptation for your particular environment.

Distributed Bus/Web Services Approach

If you do not want the client to have to be wrapped or smart about how to connect to the servers, you can go with a distributed bus or a web services approach. This is an approach where the services are each individually "wrapped" in a way that the clients can dynamically "discover" them. This generally requires much more of an infrastructure investment as the bus must be standard within the company.

Issues to ensure a successful MES Augmentation

There are several factors that can bring down the best intentioned augmentation. Be sure to use this list as a way of helping to ensure your success.

Resources:

Hardware-

Servers. Your servers will need some horsepower, fast i/o, fast CPU, and lots of memory. Many middleware application servers need memory to achieve the performance levels expected. Although these are powerful pieces of software, they can expect a high-end machine to perform acceptably.

Clients- keep them as thin and fast as you can. Look at how you will be creating your user interfaces. Java (or VB, or any other language) GUIs are pretty, but eat memory and require more out of the client. That does not make them a bad choice, just be sure your client is scaled appropriately. Web interfaces have a lower demand, but browsers can be less stable. If you are going with a web front end, make sure your end users know how to restart the browser and that the client is not storing any necessary logic within a transaction. Be sure you consider the implementation on the floor- what does it take to put a new client out on the floor? Putting PCs on the floor requires that someone understand PC support and installs. We use a linux based client that auto pulls it's configuration on first bootup. No user interaction required, except for a barscan of the location marker, (or hand typing in the number, if wanted).

Networking- adding the extra communication to servers requires a good network in place in the manufacturing site. Stand alone systems that are RS232 connected to equipment avoid the impact of networking and network management, but make it nearly impossible to interlink data, trace full process flow, achieve part traceability or easily maintain the client software

Software- Are you taking the middleware approach? Which application server/web server or messaging software will you be using? What are your evaluation needs? Are there any message bus installs or upgrades that have to happen? Do you have all the necessary database drivers to connect to your core datasources? There are a million little show stoppers that can grind a project to a standstill if they are not in place here.

Human – there are multiple groups of people who will be involved in a large augmentation, and it is important that you have all of them at the correct level of awareness/ability.

Information Systems- If you do not have IS/IT staff with *deep* knowledge of middleware systems available and on staff already, track down consultants who can help you through the implementation and train existing staff. This is a tricky thing to learn by the seat of your pants, and distributed systems have

been around long enough that many consultants are available

Operations – Ops has to buy in and work with the team to clearly define requirements, manufacturing flow and a reality check. They will be deeply involved in user testing, approvals and verification of software. If you do not have sufficient Ops support, it can bring the project to a stop.

Engineering- This group needs to assist with process flow definitions, best practices, validation checks, etc. Again, without sufficient domain expertise from engineers, the project could fail.

It is critical that all three of these groups are prepared and have a commitment to move forward before you start your project. Have a well defined method of communication and keep the lines wide open. This will help to ensure your project goes smoothly.

Planning:

The best intentions and grouping of resources will not translate to a successful project if your planning is not complete.

Timing- This is the most critical planning factor, as this will inevitably result in down time in your facility. Downtime will be incurred both for testing and implementation. Do not make the mistake of putting code into production without doing some production testing first, it will inevitably You will have to look at whether a phased or big bang switch over makes more sense, again size and complexity of the facility are big factors here.

Costing, ROI - Although an augmentation is often significantly less than the cost of the new MES install and configure, it is still an expensive undertaking. This must be a business case decision. In small enterprise where a new MES install only costs 500 K, spending 1 Million dollars to put in a framework makes no sense. But consider all factors before deciding. Maybe in a small enterprise, you could scale back the framework and make it more affordable. Is a five 9s(99.999%) uptime really necessary for your shop? In a larger enterprise where a new MES may cost 3 Mill to implement (or more), this may be a winning situation without even considering the other factors.

Costing, hardware costs- adding a tier to your system can sometimes add to cost of initial development, but it is an investment in infrastructure to avoid costs later. It will be cheaper in the long run to have your database connections and drivers centrally located than to have multiply licensed copies installed on many clients. Especially when the time for a database upgrade with a driver change comes. We have seen a case where a database driver upgrade on clients took several months in a semiconductor fab. Compare this with a one-time switch on a server. Although more expensive, the extra hardware is sometimes necessary to handle load of all users or additional system functionality.

Costs, ongoing development- If you are putting a framework in place, will the framework be static or need upgrades? Remember that static software= dead software. And eventually dead software is de-supported. Changes in programming language growth can also make previous versions unsupported, some functionality may be deprecated over time, new functionality may be required. Any system you put in place, whether purchased or built, requires some ongoing programming. Be sure you are planning for this, and the resources to keep it a healthy system.

Technology lifespan- Part of any good IT plan takes into account the technology lifespan. Realize that you are not going to extend the life of your legacy

system forever. At some point in time the system you are augmenting *will* have to be replaced. Be sure that your augmentation is appropriately system specific as discussed above. On the other hand, just putting in an application server will not solve all of your problems. No matter which technology approach you take, you will have to have some code written. You need to write some "code glue" that can specifically interact with your existing systems and pass through the data and business logic that will be necessary for day to day operations. The task then becomes to write this code in such a way that it is "plug and play" or reusable even if components are replaced or upgraded.

Choice of interfaces- In any case possible, avoid creating new, proprietary interfaces and use well defined interfaces and industry standards. It has been repeatedly documented that doing so will lower your costs, both for implementation and on-going.

Total Cost of MES Ownership

If you decide to augment your MES, this now becomes a part of the total cost of your MES ownership, and has to be factored into many other decisions. Be sure that you have identified those spots and appropriately accounted for them. People from the affected systems and groups will have to be notified and involved in the changes.

What is your IT plan?

If you are a division, be sure to check with corporate on direction and impending corporate implementations that may conflict with your decisions. This is especially important if you are planning on adding on other packages which are not included in this round of functionality and will want them to communicate in the future. Verify that you are moving in a common IT direction, and not moving farther from corporate standards, and be sure that they have a good understanding of the further functionality you will offering the corporation as a whole. There may be other higher level reports or applications which can take advantage of the added functionality you are providing. If this is the case, you may be able to get them to pay for part of the project.

How will growth affect the Cost of Ownership?

If you have a legacy MES that is either licensed or supported according to connections or data storage, you will want to consider the impact you will be seeing here. Any added functionality will see growth in 2 main areas- data storage and consumption and system connections. An increase in either of these areas could have an impact on the cost of your licensing and support. If your additional functionality also depends on the core MES, the added load could also impact performance and require upgrades to maintain user expectations.

Failure and uptime .

The importance of these depends on the manufacturing uptime of your facility and the expense a downtime might cause. Getting relatively high uptime is easy and cheap these days, it is getting that last one or two percent of the uptime which can add significantly to the implementation and ongoing maintenance costs. You need to analyze the potential business loss if the system is down and decide on

your baseline requirements.

Example of a MES Augmentation project.

At the Delphi Automotive Systems, Delphi D Semiconductor Fabrication Facilities, (also known as "the fab") we had implemented Computer Integrated Manufacturing (CIM) in a variety of ways, for a variety of reasons ,over a period of approximately 8 years. These ranged from point solutions that use proprietary languages (Wonderware) to programs written in c or Visual Basic. This was a little more object oriented, but still resulted in different applications for every client. However, the long term support was killing any attempts at doing ongoing development. Doing the necessary ongoing changes to keep up with changing business rules was preventing us from doing new deploys. A change in tactics was necessary.

After evaluating the economic and business factors, it was soon apparent that an MES upgrade (the best solution) was out of reach for the time being. This meant finding a way to augment the existing WorkStream MES to handle increased demand for recipe verification, prior- step verification and a more common user interface. The road to developing a middle tier, common point of connection system was rocky and bumpy and full of pluses and pains.

By putting together a middle tier solution- we learned that we could create a common GUI interface to a widely disparate group of equipment. The operators on the floor liked it, there was a low training impact, easy to move operators from one piece of equipment to another/ This was a flexible Java- based GUI that auto configured out of a backend Oracle database, rather than hard coding individual equipment differences in client. It allowed for common client across the manufacturing floor.

Using an application server made it easy to manage middle tier functions such as connection pooling, fail over, security, user authentication, hot deploy, etc. This was *significantly* easier than hand coding the functionality.

As in any undertaking like this, we learned that the tools/ scripts used for deployment and testing were critical to save time and headaches. Spend some up time effort to purchase or build these and it will pay off in the long run.

However, we did find that distributed programing is much trickier to debug, and it is hard for some procedural coders to change their mind set and do the full system debugging. This was a bigger stumbling block than we expected, and the human resources used in your project should be carefully selected. The learning curve was much steeper than we had planned for, and why we are recommending that you bring in experienced consultants. No matter how bright your programmers are, don't assume or expect them to learn this by the seat of their pants.

Not surprisingly, when your resources are pool together, small changes in one part of the system can have huge impact in other parts. This is not a problem if test protocol covers this and protocol is followed carefully. This is a big problem when protocols are incomplete, or schedule pressures are allowed to overwhelm proper testing . No matter how boring or tedious, do not allow the pressures of the production floor to push you away from a well documented test procedure. Spend the time up front to carefully develop test procedures, make sure that both engineering and operations are involved in testing, and the developers without domain knowledge are not verifying your system.

Our framework was internally developed and although we thought we had an adequate support plan, this later developed into an issue. Be sure that your support

plans are accepted by all parties before starting your development. If head count for support is an issue for you, you will want to be sure that you are choosing technologies and methodologies that can be easily outsourced for support.

Conclusion

The decision to augment versus replace/upgrade needs to be made on both a business and a technical case. If you can get 5 or more years out of your legacy system, but it does not do everything you need it to do, it is a good candidate for an augmentation. If it is going to be replaced in less than 3- 5 years, carefully consider all the other factors and their impact before deciding to just patch the system. If you are going to be replacing the system in the near future, go look for an MES that does everything you want your system to do, with a little growing room. Be sure your next choice has an open API, so that it will lessen the impact of future system changes on the enterprise.

A well designed framework will allow you to choose best of breed modules and change them out without impacting user interface or floor training. It allows for flexibility in keeping up with technology changes in the future, and can allow you the freedom to implement functionality on an "as needed basis", lessening the initial investment for your facility.

Which approach is best? In general the best approach is the middleware/server wrapping approach because it supplies the most flexibility without modifying the original systems. The web presentation approach is good when you are really just interested in some sort of common interface to all the components but this approach requires that each of the components have the ability to present to the web in some fashion.

We are based out of a semiconductor fab, but what other manufacturing sites make lots of sense for using a middleware framework and augmenting existing systems, rather than a clean replacement?

- *Anyplace where manufacturing time is 24X7*- if your enterprise is closed in the evening or over a weekend, doing a replacement will not be so expensive in terms of lost manufacturing time. Also making code changes to clients, making upgrades to business logic, etc. will not be so much a consideration in terms of timeliness and cost savings.

- *Where business logic changes often*- if you find yourself often having to make code changes to your applications because the business logic in how you run your enterprise changes often (manufacturing rules, process controls or EDI requirements from customers...) then you are a candidate for a framework with isolated business logic. This makes it easy to centrally change business logic with less impact on system availability. If you have been running under the same business rules for the last 3 years, and your customer base has changed little, if at all, you are likely to not consider this so strongly. (of course you may also be considering if you will still be in business next year....)

- *If you have a large number of legacy systems* that have islands of data, making it hard for you to make well informed decisions. Having a common interface where data from multiple sources can be easily combined and viewed will empower you to make more realistic decisions.

- *anywhere you like*- if the business and technical cases seem to sell it, this is an approach that can be used almost anywhere, even in non-manufacturing systems.