output.industries

Whitepaper.

5 Keys to Successful MES Development.

“72% of manufacturers are still using paper and spreadsheets to manage their production floor.”

Overview of a Manufacturing Execution System

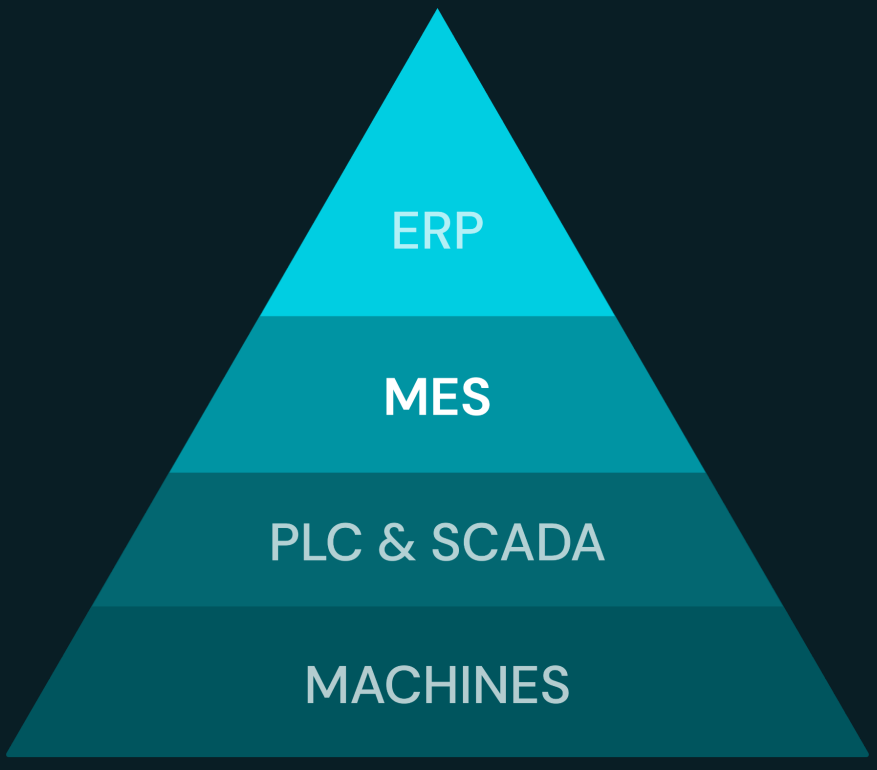
A Manufacturing Execution System (MES) is a software solution that helps manufacturers manage and optimise their production processes.

It provides real-time visibility into production activities and enables manufacturers to track and control every aspect of the manufacturing process, from raw materials to finished products.

MES solutions typically include features such as work order management, quality control, inventory management, material movement, and full tracking and genealogy reporting.

These features help manufacturers improve efficiency, reduce costs, and increase the overall quality of their products.

MES solutions can be integrated with other manufacturing software, such as Enterprise Resource Planning (ERP) systems, to provide a comprehensive view of the manufacturing process.



5 keys to successful MES development

1. Work collaboratively

Domain experts aren't always your software experts (and vice versa).

Finding a way to allow team members with different skills and experience of your shop floor operations to all contribute to the design of the system is a key to success.

Various tools and techniques can be used to help with this:

- **Workshops with a modern twist.** There is no denying that being able to run a productive workshop is a skill worth having. Tools like Jamboard from Google, and Miro, allow team members to contribute to whiteboard sessions remotely (and asynchronously).

- **Careful separation of requirement and solution.** It is all too easy to solutionize without properly defining the ultimate need. This can lead to design decisions that are hard to review and verify.

- **Visually Represent Processes.** There is a myriad of software available in 2022 that allows people (even with questionable graphic design skills) to draw requirements and processes. Miro.com is a good example. Have these diagrams online, versioned and able to be iterated on.

- **Clickable prototypes.** Tools such as Figma make prototyping interfaces easy. By linking these screens together, it is possible to create 'clickable prototypes' that allow the team to collaboratively identify issues that may not have been picked up earlier. It is much easier to fix issues in the design phase than in software.

2. Model your domain

Take time at the start of all new packages of work to define and agree the domain model. Domain modelling is an excellent way of addressing complexity early on in the project.

A domain model incorporates the actions that will be performed on the data as well as the data model itself.

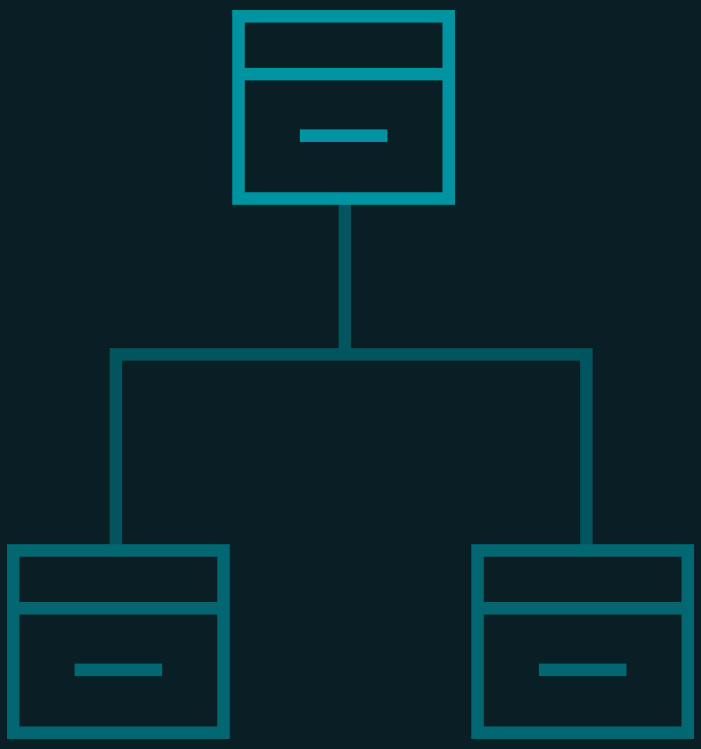
As you go through the requirements identified by stakeholders, build the domain model as you go. The team will quickly identify issues at this stage that can be discussed and rectified relatively quickly (compared to having these issues causing issues during the development phase).

If the data model is correct, it severely limits the possibility for anything downstream to go off track.

Test new features and requirements against the current domain model, and ask yourself what the most elegant way to accommodate them is. Make your data model intrinsically easy to understand. It's not about creating the most abbreviated or esoteric schema you can.

Rarely is significant performance gained by heavy use of abbreviations or re-using data fields for unrelated purposes.

Having a data structure that is clean and easy to understand will filter down to all aspects of your system.



3. Manage & compartmentalise complexity

If unchecked, software being developed will naturally tend towards complexity, with technical debt eventually creeping in and eroding performance and progress.

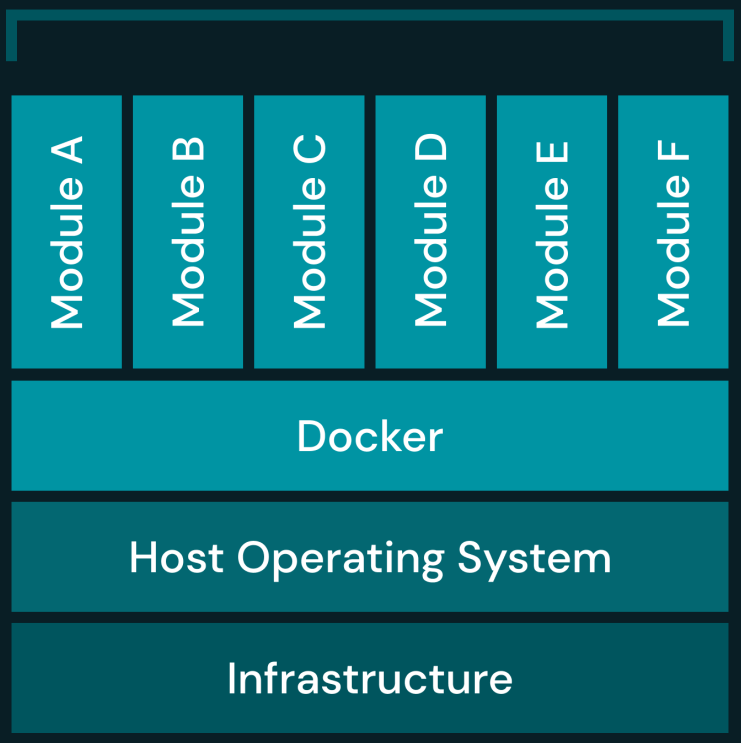
Generally speaking it is easier to add complexity later, than remove it from something existing. In other words, the design should only be as complicated as absolutely necessary, and no more.

Often when the requirements aren't fully understood, there will be the temptation to keep adding more to 'cover all bases'. The team should fight this temptation, and keep in mind more can be added later if really needed.

A way to combat necessary complexity is to think about logical boundaries in the overall system, and use namespaces to keep separation.

These namespaces should be used to organise everything from the UI, to APIs and the data model. The more consistency through the tech stack the better.

Clear sections to the system will make it easier to navigate and hide complexity from users that do not need those features.



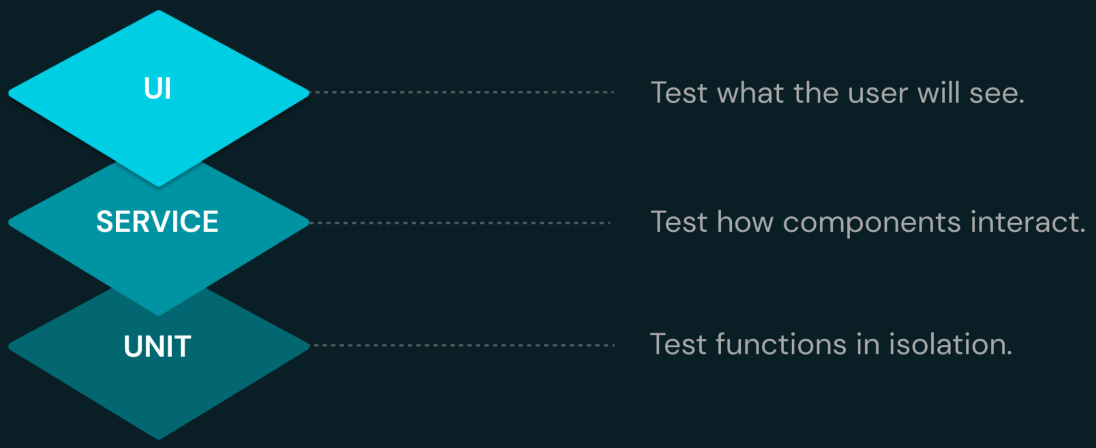
4. Test, test, test (The test pyramid)

The ability to move quickly and innovate with software is directly linked to the system's stability and team's confidence in that.

If you find yourself taking 1 step forward, and 2 steps back each time a change is made, any ability to innovate will be stifled.

Having sufficient automated tests will mean that critical bugs and issues are found and eliminated long before they become an issue in production.

Mike Cohn's original test pyramid consists of three layers that your test suite should consist of (bottom to top):



As well as these it is a good idea to also add stress and performance tests.

By ensuring your build and deploy systems automatically run these tests, the confidence you will have in your software will .

Which leads to our next point...

5. Automate everything

Every step in your software development lifecycle should be automated via a script or other process.

All steps manually performed, whether that is inside a console or copying files over to a server, are an opportunity for mistakes to happen or processes to be forgotten when they are needed most (i.e. in emergencies).

Conversely, every time a task is automated, that time is freed up for the team to work on something else. Over the course of a large project, this will add up to significant productivity.

Automation can also act as a useful form of documentation. When a process is automated, and regularly being used to perform that task, it can be relied upon as information as to the complete and accurate way to perform that task. Written documentation that is disconnected from the process itself can easily become out-of-date or incomplete over time.

Introducing output.industries

We work with our manufacturing partners to solve critical digital challenges and help see opportunities others don't.

Plan. Operate. Optimise.

Our cloud native Manufacturing Execution System (MES) leverages industry leading open source technologies with robust and secure API connectivity to enable tight integrations with existing platforms and systems.

Our platforms provide the perfect building blocks to begin unlocking the power of data and create a unified data structure which can be built upon and customised to meet bespoke manufacturing processes.

Made to measure, built to last.

What next?

Get in touch with us to discuss your MES challenges and understand how we can start to unlock the power of data to drive operational efficiencies in your manufacturing operations.

hello@output.industries