QEC IPMS the Technical Challenge

James McKelvie MEng(Hons), Paul Lakey CofC Ch.Eng MBCS[1]
L3 MAPPS Limited

Synopsis

The task of scaling up a well-established Integrated Platform Management System (IPMS) to meet the requirements of the largest warship ever constructed for the UK Navy has provided a diverse set of challenges. Designing a control system for a constantly evolving ship design was never going to be a straightforward task, especially with the complexity of the Queen Elizabeth Class (QEC) aircraft carrier and the design intent of reducing crew size by automation. Adopting a development cycle suitably capable of providing resilience to the dynamic nature of the ships development would always be a significant risk. Some of the challenges were evident from the outset, such as the system requiring a multitude of external interfaces supporting a variety of communication protocols. The most demanding of these being the Electrical Power Control and Management System (EPCAMS), which requires the transfer of almost 10,000 signals. Whilst other challenges manifested themselves in unexpected ways such as the engagement with ships staff in the early stages of design, providing an excellent means to create solutions best suited to meet the expectations and ergonomic requirements of the end user. Also providing conflicting opinions of individuals resulting in a minefield of requirements to process in order to arrive at the best engineering solution. This paper explores the most significant challenges and the processes put in place to mitigate against the risk that they provide. Analysis of these processes and procedures allowed us to establish a well-defined set of lessons learnt that can be used to optimise and improve the development cycle for future projects. Many of these processes targeted the de-risking of the solution via testing, simulation and the engagement of the operator. Early de-risking workshops helped identify bugs early on in the design, the use of test rigs and simulators provided continued assurance and the development of automated test tools for testing both hardware and software proved invaluable. The use of a shore-based training facility provided early exposure to the operators, whilst also providing continued confidence in the system's capabilities. All of these contributed to the consistent deployment of software to the ship despite an increasingly demanding schedule.

1. Introduction

With software currently operating on more than 140 naval platforms worldwide and 30 years of experience, we were well suited to the task of developing and deploying an Integrated Platform Management System (IPMS) to meet the requirements of the Queen Elizabeth Class (QEC) aircraft carrier.

The IPMS solution for QEC is the largest UK Naval IPMS solution ever produced, consisting of 42,000 I/O signals processed by 250 PLCs, which communicate to 76 operator workstations [1]. The IPMS application itself consists of:

a) CORE application, which is used to process data received from the PLCs, hosts the application logic used to provide automated sequences, supports the calculation of 5000 derived signals and 6500 calculated devices and ensures synchronisation of the application across the ship.
b) DCS (Distributed Control System), which is the configuration of the PLCs, their connection to Plant and their ability to transmit and receive information from the CORE application.
c) HCI (Human Control Interface) application, which provides the operator with system pages, provides an interface to multiple external interfaces and generates read and write commands to the CORE application.
d) POBT (Platform On-board Trainer) application, which provides the operator with the ability to switch IPMS into a training mode allowing the operators to improve their understanding of the systems, whilst simultaneously maintaining the control mode.
e) DSAC (Damage Surveillance and control) application, which provides the operator with the capability to manage damage situations via advanced monitoring and control as well as providing enhanced communication conduits.

[1]Authors' Biographies

James McKelvie is a Software Engineer working for L-3 MAPPS Limited who graduated with a MEng from the University of Bristol. He has worked in marine engineering since graduation, initially working as a Systems Engineer progressing into Software Development. Primarily working on the QEC project he has also taken time out to work with the Naval Design Partnership developing conceptual designs.

Paul Lakey is a Principal Application Engineer working for L-3 as the Technical Manager and System Technical Authority supporting the Queen Elizabeth Class Aircraft Carrier's installed IPMS system. Paul started as an Application Engineer for L3 writing and progressing the Queen Elizabeth Class Functional Design. Prior to L3 Paul was in the RFA for 23 years as an Engineering Officer and has obtain a number of machinery OEM accreditations and further class recognitions.

http://doi.org/10.24868/issn.2631-8741.2018.026

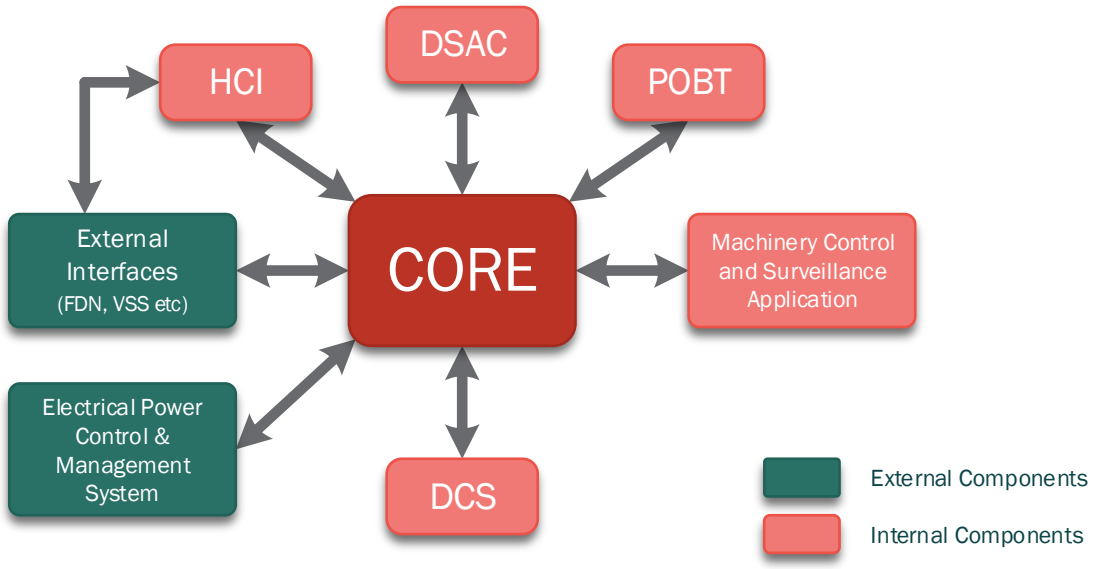

Figure 1: IPMS Breakdown

Many of the components of the system were well established and at a high level of maturity, but the architecture of the ship meant that there were a substantial amount of new external interfaces to deal with, these ranged from the Fire Detection Network (FDN) to the Video Surveillance System (VSS). Whilst many of these are simplistic entities, the vast amount of these interfaces meant that there would be an unprecedented amount of data being transferred across the system. Good engineering practice stipulates that a successfully developed system should be capable of being scaled up to a larger capacity, but until you have actually attempted this growth, despite extensive modelling, it is often very difficult to understand if the system can cope with the expansion [2]. This was demonstrated on multiple occasions throughout the development lifecycle, forcing a necessity to enhance pre-existing functionality as well as to develop new functionality.

The system also has a number of performance requirements such as an end-to-end speed, which means that a signal toggled anywhere on the ship must have been received by the DCS system, transmitted to the CORE application and passed to the HCI application for an operator to see on every workstation on the ship within a maximum period of time. The architecture of the system also ensures that there is a large amount of redundancy so that any single point of failure can be absorbed by the system. This entails an additional performance requirement to ensure that should an operator workstation fail that any alternative workstation can pick up any tasks being run by the failed workstation within a certain duration of time.

2. Development and Delivery

The development of the solution utilised the Department of Defense Architecture Framework (DoDAF) in order to generate an architecture that was both easily representable as well as provide suitable transparency to all stakeholders.

For the software development, a V-Model development Process was selected to suitably establish the relationships between each phase of the development cycle [3]. In simplistic terms, the process requires analysing the requirements from the customer, transposing these into a functional specification, which is then subsequently used to produce a detailed design. This detailed design is then coded and the output tested against each of the prior steps as shown in Figure 2. Throughout the process, there are a number of review gates to ensure that the design is of suitable maturity to progress; these include a Preliminary Design Review (PDR), Critical Design Review (CDR) and a Test Readiness Review (TRR). The process also includes progressive testing and acceptance in order to build confidence within the design cycle.

http://doi.org/10.24868/issn.2631-8741.2018.026

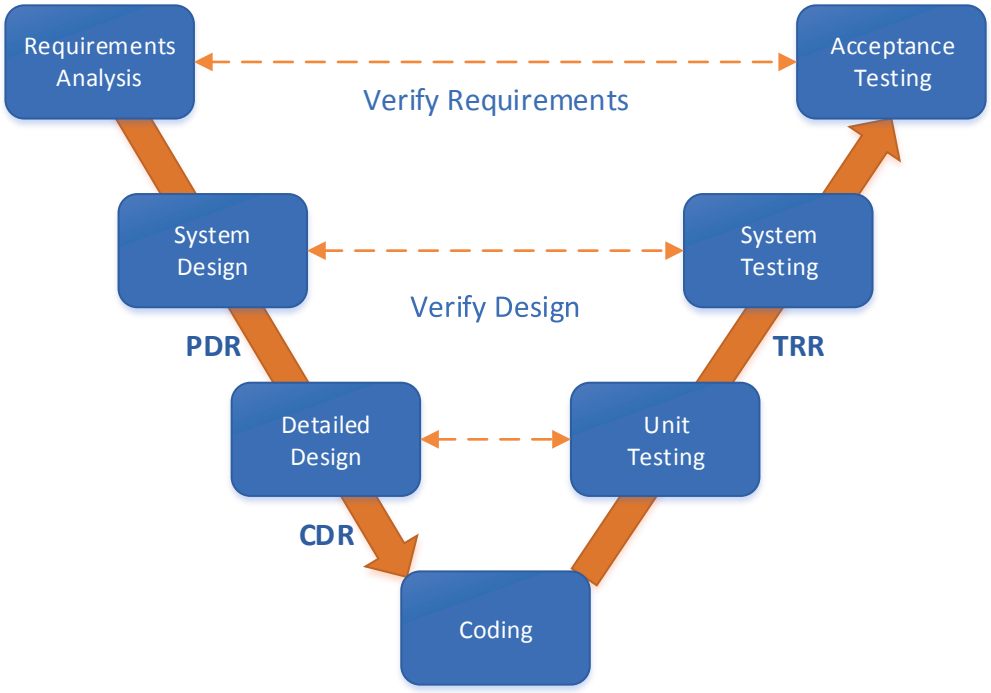

Figure 2: V-Model Development Process

The use of the V-Model allows for concurrent development of subsystems against a backdrop of immature data, allowing modular builds of the software in order to progress individual systems. This generates a reduction in the quantity of rework required as well as aiding in maintaining programme schedule. The capability of segregating systems led to the development of a delivery release model aimed at delivering systems to the ship in order of their maturity on-board. A significant amount of work was carried out to ensure that releases of the software were available to the ship in alignment with the maturity of the systems on-board. This allowed for the utilisation of IPMS to aid in the commissioning process.

3. The Technical Challenges

This section explores some of the technical challenges experienced during the development of the IPMS system and addresses the steps taken to overcome them. Taking these encounters and extracting the lessons learnt from them provides a real opportunity to improve the way in which we develop future systems.

3.1. Configuration and Change Management

The number of configurable items required for the construction of the software was always going to be immense, add the complexity of the release model and the obligation to be able to build varying components of the system at will and you could find yourself in a position of disarray. The need for a suitable change management system was therefore imperative.

The selected solution provided both a change management tool and a configuration management tool. The change management tool appropriately enforces that the correct stakeholders review each change and that in order to submit any updates to the software that an approved change is referenced. The assessment of each change was conducted by a Change Review Board, which ensured that the correct subject matter experts as well as suitable project team members capable of comprehending the impact to the schedule were present at the review. The tool provides full traceability allowing the source of every change to be interrogated. The configuration management tool provides a means of storing each software artefact with all of its history and versions as well as the ability to create and restore baselines. The use of a baseline allows for a snapshot of all the configurable items constituting a release of software to the ship.

The system in principal is very well suited for the task, however it is not easy to anticipate that the system would be exposed to multiple users of the system checking in and out files, whilst other users restore entire baselines or backups were being performed. The performance of the system, which was hosted on a local server situated at one site in the country, could be problematic. When the speed at which engineers can upload their files to the system begins to eat away at an already tight schedule, which is not designed to accommodate this task, you know

http://doi.org/10.24868/issn.2631-8741.2018.026

that action is required. As with most performance related issues, optimisation of the system was integral, working with the manufacture of the change management system to resolve the issues was well received by frustrated engineers.

The configuration of the tool was updated when it was discovered that users could make a change to a file which had already been updated, then check in this file over the top of the previous change, creating a parallel version of the file and rendering the previous update lost. There were also a few unfortunate instances were a user would create a new instance of an item when making a change rather than updating the existing item which essentially invalidates the purpose of the tool as there would be no history to the new item. The solution to this is relatively straightforward but very effective, ensure that the structure of the data is logical and easy to follow, develop thorough processes with appropriate guidance documents and ensure that users are suitably trained.

A project of this scale is always going to incur significant change, what is surprising is the amount of time required to generate impact assessments for these changes and the complexity of creating them to an accurate level. A matrix to provide suitable estimates was developed that helps identify the components of the change and allows a quick impact assessment to be produced. Using a feedback mechanism to evaluate the accuracy of these impact assessments allowed further refinements to be made to the matrix resulting in a quick and precise way of estimating change.

3.2. Operator Engagement

The chances of creating a solution that the operator will be fully satisfied with are nigh on impossible without a feedback mechanism for the operator to provide an opinion. From the outset of the project, the alliancing behaviours demonstrated by the Air Craft Carrier Alliance (ACA) provided a fantastic forum for engaging with both the customer and the end user to achieve this.

Whilst the human control interface (HCI) of the software is not the most complex of the components within an IPMS system, it is the first and most easily criticised. With this in mind the development of a Style Guide was undertaken in order to provide a standard with which to develop the pages as well as providing user interface consistency. Utilising the end user as a reviewer for the guide handed an element of control back to the operator both to provide assurance and to manage expectations.

The Style Guide was subsequently utilised to develop the pages in conjunction with the functionality for the various systems. The decision was made to further engage with the operator and customer by carrying out page reviews on the completed pages (Figure 3). Whilst the principle of the reviews is solid, the ease of which it is to be distracted by the aesthetics of a page clouds the intended focus of reviewing the page's functionality.

Page	No.	1.1 Does the page correspond satisfactorily with that endorsed at HCI static page review?	1.2 If changes by CR have occurred since HCI static review, are the changes agreed?	1.3 Is information clearly presented to the user, including in normal view and zoom?	1.4 Are the page title, icons, labels and colours satisfactory?	1.5 Are the navigation links logical, functional and easy to use?	1.6 Is it clear when an item has been selected by the user?	1.7 Can the user tell the state of the system and see the alternative actions available?	1.8 Is there feedback for every operator action?	2.1 Will the representation of the system allow the user to perform required duties?	Specific Page Comments
AVTUR (9510)	L522	Yes	N/A	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Page satisfactory
Diesel Generator Starting Air System (5300)	L548	No	N/A	See Comment	Yes	Yes	Yes	Yes	Yes	Yes	Arrows not centred as raised at static review; SW cooling nav button naming is incorrect; title bar click does not cancel zoom; pressure sensing lines not connected to data fields and connection points to be checked/staggered; text/object centring needs correction throughout page; centre upper boxes need centing under title and overall layout asymmetric

Figure 3: Sample Page Review Output

The greatest surprise in relation to operator engagement was the benefit of live platform feedback. Whilst feedback in the design stages is useful, it cannot compare to the quality of the feedback you can get from an operational vessel. The areas that most benefited being the development of operating procedures and the rationalisation of alarms.

Whilst guidance from ex-naval personnel and reviews of the functionality at design stage by ships staff create a very good starting point for specifying the functionality required, it does not compare to the feedback given by the watchkeepers that use the system on a daily basis. Their insight into the nuances of the way in which the ships systems operate has given rise to a multitude of enhancements to the system.

During the first set of sea trials, it was evident that the watchkeepers were being inundated with alarms and warnings due to the ship at sea, unsurprisingly behaving in a significantly different way than alongside. A process

http://doi.org/10.24868/issn.2631-8741.2018.026

of alarms and warnings rationalisation was conducted which resulted in 13,843 changes to the severity and priority of alarms and warnings, 1,521 updates to alarm delays and the inclusion of 2,172 new alarm inhibits. The reduction in alarms and warnings can be seen in Figure 4.

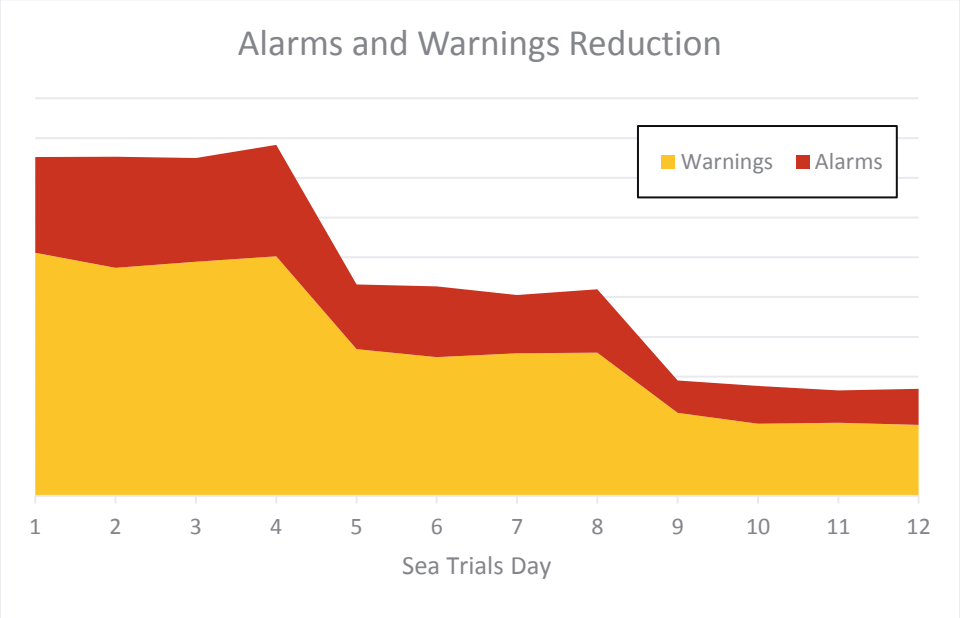

Figure 4: V-Model Development Process

3.3. Integration

IPMS interfaces with a range of external systems communicating via various different protocols. In order to successfully understand the interfaces and develop the functionality to translate each interface, a Software Interface Requirements Specification (SIRS) was provided. Each system was developed against its corresponding SIRS, however there was a significant risk of misinterpretation of the interface. To combat this risk, a method of early integration was endorsed. This involved thorough testing of the interface with the support of the OEM. This either took place within the Shore Integration Facility (SIF) a dedicated test site hosting ship equivalent hardware, or at an OEM's sites. The profits of this testing is hard to quantify, however it is fair to say that almost every interface that underwent early integration was consequently updated to iron out flaws.

The large amount of calibration required to successfully integrate some of the more complex systems that form part of the mission systems work package, such as the Combat Management System (CMS), Integration Navigation Bridge System (INBS) and the Air Group Management System (AGMA) required the use of a test facility. The Mission Systems Integration Facility (MSIF) hosted at the Portsdown technology park provided this capability. The resultant testing identified many flaws with these interfaces whilst also providing a suitable environment to diagnose the issues. The deployment of the software to the MSIF was not constrained to the same stringent configuration management that the released software to the ship was, causing a string of knock on issues.

The Electrical Power Control and Management System (EPCAMS) interface presented the greatest challenge with the goal being for IPMS to replicate the control provided by the EPCAMS system. The complexity of duplicating a control system developed by an external company relies on a thorough understanding of their system. It became evident that some of this comprehension was missing which was subsequently addressed by carrying out functional review workshops conducted with both the customer and the external company. The interface requires an exchange of almost 23,000 components between the two systems. The mapping of these components is relatively complex with the processes required to do so evolving throughout the development of the system. An earlier understanding of the complexity of this data exchange could have led to the use of a database to define and maintain the data required to implement the interface.

3.4. Automated Testing

The use of a V-Model development framework inevitably leads to a substantial amount of testing. Considering the fact that testing sits to the right of a schedule and is often pushed out leaving you with the unacceptable option

http://doi.org/10.24868/issn.2631-8741.2018.026

of missing the deadline or the fall back route of compressing the test window, you are left with a strong necessity to optimise the test process. Automated testing is the obvious mechanism with which to achieve this, with the benefits being the consistent repeatability of tests, the ability to run far more lengthily scripts and the earlier identification of issues via more rigorous unit testing [4].

The first tool developed was the apt named Automated Test Tool whose purpose was to provide the engineers with a way of running a script that would replicate any action that they would manually carry out. The tool utilises API calls directly into the CORE application to provide this functionality. The actions range from setting the value or status of a signal within the CORE applications database or imitating a command that might be sent from the operator via the HCI. The tool was an immense hit causing a sudden expansion in the depth at which testing was being conducted and highlighting a need for a more suitable means of managing these tests. The subsequent evolution was the development of schedules, which provide an ordered list by which to run these tests.

The distributed control system (DCS) on QEC makes use of 250 PLCs around the ship, each of these transmit data to a CORE application either directly or via a parent PLC. Testing is required to make sure that when a signal changes at the PLC level that the corresponding change within the CORE application is correct. Whilst the test is a straightforward one, toggle the signal at the PLC and check the response, the amount of re-testing required becomes vast when you are required to retest the entire PLC for a single change. To counteract this vast amount of testing a DCS Test Tool was developed. The tool functions by switching the PLC into a test mode, which prevents the I/O modules from updating the signal variables that are transmitted up to the CORE application. An OPC server is configured to contain each of these variables and is then connected to the PLC. The DCS Test Tool itself interfaces with the OPC server providing the functionality to set the variables as desired, the corresponding action is then monitored at the CORE application. The time saved using this tool allowed for multiple iterations to the PLC's configurations without causing a substantial impact on the schedule. Going forward there is a significant improvement that can be made for this tool, by closing the loop and allowing the tool to not only to set the variables, but also read back the resulting states of the signals within the CORE application using the API calls already employed by the Automated Test Tool (see Figure 5).

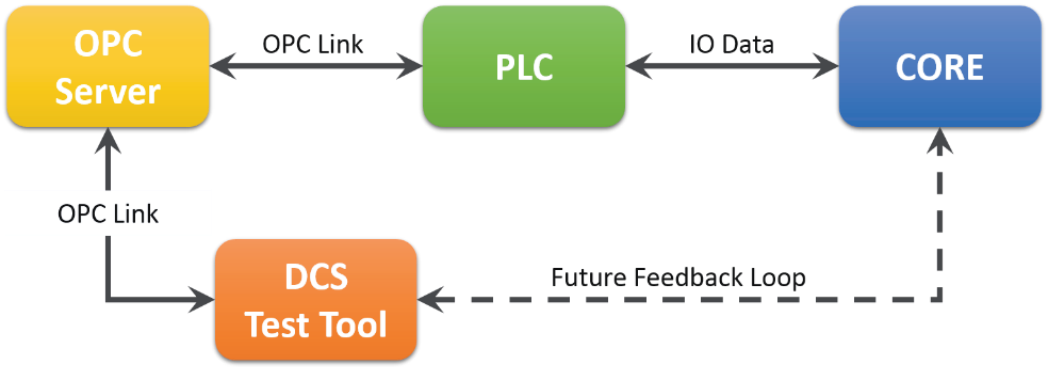

Figure 5: DCS Test Tool

The EPCAMS interface has already been mentioned as an area of significant complexity due to the number of variables required to be transferred. It is necessary to test that the impact of changing these variables in the exchange causes the right reaction either side of the interface. Fortunately, successful planning meant that within the test environment of the SIF a version of the EPCAMS application was provided, which allowed the exchange to be interrogated as required. Unfortunately, with the sheer number of components in the exchange and a number of planned releases of the EPCAMS software, it was not going to be possible to do this manually. The EPCAMS Test Tool was therefore developed which takes an empty PLC and loads it with the entire exchange worth of data. The test tool then sequentially sets each input variable of the exchange to ensure that the CORE application responds in the correct manner. Whilst the tool only tests the inputs to IPMS, these variables contribute to 90% of the data in the exchange. The test itself was left to run on its own and would take approximately 48 hours to run, when you consider how much time this would have cost for manual testing the saving is immense. The ease of running the tool also meant that the test could form part of the standard set of regression tests allowing the interface to be tested for every subsequent release of the IPMS software.

http://doi.org/10.24868/issn.2631-8741.2018.026

3.5. Performance Enhancements

With the release model focused on the delivery of systems at varying stages, it is not until the final releases that you have a completed system. In addition to this, a test environment such as the SIF does not truly represent the live ship. Using a test strategy primarily focused on testing the functionality of the systems leaves a gap in monitoring the performance of the system. Issues stemming from the performance of the application became evident on-board the vessel during trials. Running thorough diagnostics on-board allowed the issues to be broken down in a very comprehensive manor permitting the root causes of the performance issues to be identified. One of the more severe issues encountered stemmed from the application carrying out alarm inhibit calculations, with the number of inhibits being calculated being far greater than the system had ever supported, the CPU demand for the workstation was vast. Another significant issue also causing an unpredicted increase in CPU usage was the method by which the application was monitoring alarm delays, again there are more alarm delays in the QEC IPMS solution than in any previous project. The resolution of these issues was to reduce the CPU usage on the workstations by completing a task of optimising the code calculating these parameters. The means to test these enhancements was quite novel and utilised playing back live data recorded from the ship to ensure that the CPU usage had been reduced to a suitable level.

4. Conclusion

Whilst the size of the QEC IPMS system has been mentioned many times during this paper, it is worth stating how useful it has been for debugging some issues that would or could only ever have been discovered on a platform of this size. Some of the subsequent enhancements have been incorporated into the foundation of the IPMS solution and will be deployed to both current and future projects.

To utilise a change and configuration management system successfully in the future it is vital that the requirements from both a performance and functional stance are well understood and that the users of these systems are fully trained in how to use them.

Engagement with the end user will continue to be very important and careful thought should be taken to assess when they should be consulted. A project schedule should also incorporate time to carry out live platform operator feedback rather than have it exist as a reactive measure. The benefits from this feedback route are significant and the capability of the product greatly increased.

For the development of future interfaces, it is important that accurate requirements be obtained to increase the accuracy of the first attempt. However, it is essential that early integration activities are included within the schedule to ensure the success of the interface. For complex interfaces, it is worth taking the time to fully comprehend the manner in which it works in order to save time by implementing an adequate solution from the outset rather than causing a heap of re-work. It is essential that the configuration used for interface testing be accurately recorded so that the working solution can be successfully deployed. This includes recording the configuration both sides of the interface.

The automated test tools can and will be developed to suit future project needs. There are however, limitations to the capability of automated testing and it is important not to try and over engineer a solution. There should be an understanding that some tests should be conducted manually due to the cost of developing an automated means of testing far outweighing the cost of the manual testing.

Whilst the advanced performance testing conducted at the MSIF was productive, the inclusion of basic performance tests being conducted on every release of the software would have been very useful. This would ensure that the development of the product does not have an adverse impact on performance and provide a history of the systems performance that can be analysed for future optimisation.

5. The Future of IPMS

We have discussed the benefits of including automated testing, but there is another step in the automated testing environment and that is to automatically build, deploy and test the software. Imagine a standalone system capable of building the software to a specified baseline, deploying and executing the IPMS solution and then extracting and running a comprehensive set of tests. This method of continuous integration has already been proven within many organisations that produce software and provides the functionality to leave a system to test itself. The necessity to select suitable regression tests for a new release of software would be a thing of the past as the system would be capable of running all tests.

http://doi.org/10.24868/issn.2631-8741.2018.026

Within the current world of both naval and commercial ships, there is an ever-increasing trend in the amount of data available from the ships equipment. Largely driven by the intelligence and efficiency of modern components as well as the drive to maintain maximum performance, "Big Data" is a reality for modern IPMS systems. The ability to act as an efficient conduit for this data as well as processing it in an optimal manner are paramount to any new effective IPMS systems.

The decision to include safety as a central component in designing the new Type 26 frigate class shows a real desire for safety to be given a greater focus. The development of safety-accredited IPMS relies on a thorough, transparent and rigorous approach to developing software. These attributes should already be at the heart of any software development process and the necessity to accredit should only lead to reinforcing and enhancing processes already in place.

6. References

1. J Davis, S Jewell. Controlling the Royal Navy's Queen Elizabeth Class Aircraft Carriers
2. C U Smith, L G Williams. Best Practices for Software Performance Engineering
3. R S Yadav. Improvement in the V-Model
4. D Kumar, K K Mishra. The Impacts of Test Automation on Software's Cost, Quality and Time to Market

 http://doi.org/10.24868/issn.2631-8741.2018.026