Challenges of Control System Acceptance

Christopher Goetz

Kingston Systems LLC

KINGSTON SYSTEMS LLC

Challenges of Control System Acceptance

Why do we continue to experience quality problems with control systems software on rigs? Why do we accept 20% downtime on our new equipment? And why can't software and testing quality be treated with the same respect in our industry as in many other industries; the aerospace industry for example, where 100% uptime is an absolute requirement!

We have all heard of serious software and control system issues on new equipment. Many in the industry have compared the experience of accepting new equipment to that of buying a new luxury car that breaks down on its first trip home from the dealer, only to have the dealer insist the buyer pay to have one of their mechanics to be on call 24/7. Analogies such as these force a grimace of understanding and shared experiences. But real world examples such as the two here, must force us to acknowledge and address the issue.

1. A Top Drive accelerated out of control after a technician came on board to 'tweak' its' parameters.
2. Remote or 'dummy' terminals had ability to control safety critical equipment. It was discovered that Active Compensating Valves could be triggered from the Sack Store room.

This paper will examine some real world case studies and explore answers to the questions outlined above. It will holistically discuss how software is treated in the acceptance process of our rig equipment. Finally it will provide some practical steps that all operators and contractors can take, today, to improve the situation.

Case Studies

The thinking that software fails is a bit of a misnomer. Software does not fail, age or wear out. It is simply there, subject to the inputs that it receives for processing. It is not until a specific sequence of events occur that the output is undesirable and our concept of 'software failing' comes to being. Our primary opportunity to correct this is through proper testing at all stages of design and operation. The following case studies illustrate what happens when testing is insufficient.

Incorrect Commissioning Procedure

In the first example we consider the impact and consequence of an incorrect commissioning procedure.

> Case 1: A Kingston System Inspector was asked to attend and witness the commissioning of the top side Fire and Gas detectors. The generic,

> unedited, vendor commissioning procedure was provided and the commissioning started. Testing commenced, looking for an individual sensor gas signal and system response at 20% and 60% gas concentration. However, a quick review of the Functional Specification combined with the Systems HMI (human machine interface) indicated to the inspector that the system was supposed to provide a unique response when two co-located detectors measured 20% gas. All other parties missed this error. When brought to their attention the shipyard sheepishly admitted this testing would be more valid. The Class inspector thus completely rejected all testing to date and we started over.

We were lucky; the testing was successful and the software appropriately provided the correct output with the various input situations. However if this flaw had been missed then this portion of the code would never had been tested and the vessel could very well have sailed with the F&G system unable to detect small quantities of gas over large areas. This kind of oversight could prove devastatingly fatal.

But how did this happen? In our rush to complete the project the parties involved all failed to connect the dots between the design requirements and the testing program. This error probably initially happened several months before in the factory, probably before the FAT (Factory Acceptance Test) when the non-co-located engineering teams did not communicate the change in design requirements. Perhaps it was a case of a non-standard product delivery. Perhaps this client had actually requested the design change. One team implemented it; however the next failed to redesign the testing procedure to account for the change. The result was the provision of a standard test procedure. Thus the error propagated through FAT, construction and into acceptance.

A solution might be to expect the client to monitor the vendors' every step through the design, construction and testing phase. This is not realistic as it requires the clients to impose project management controls on the vendors. This is neither their core competency nor a desired state by any party. The alternative is leaving the vendor to their devices; that led to this unsatisfactory situation in the first place.

A better solution would be to work with the vendor, who by proxy are software companies and have software development and management techniques in place, to improve how your specific requirements are being reviewed and communicated between the various departments. Specifically a client can reasonable ask to expect to inspect and provide direct input upon the development of documentation and testing procedures specific to their requirements at any and all stages of the process. Vendors should welcome this as an opportunity to confirm that requirements are still valid and that they are working towards the correct goal.

Access to Restricted Operation

The previous case touched on lack of communication with the testing department, but the seeds of poorly designed software might be deeper than pure testing.

> Case 2: Part of Kingston Systems service package is to attempt to test the controls systems in as many noninvasive ways as possible. During one of these tests the inspector noticed that a significant level of command and control was given to the office terminals in even their lowest command state. Said another way, the office mimic machines had the ability to command and control drill floor equipment like motors, pumps, and compensating and tensioning valves. The implications of a miss-guided key press could be devastating. Shutting down the Crown Compensator from the office during operation was a conceivable case.

As in the first example the standard testing package was not satisfactory. In this case the application was indeed programmed incorrectly and the software was ready to happily accept a valid input and produce potentially disastrous results. How could a standard product that has been installed on several rigs have missed this?

The likely explanation is the lack of a clear and detailed functional specification. A functional specification is the engineers blue print against which they cut steel. The same is true of software development. Without a clear and detailed functional specification the programmer can only guess what a valve is supposed to do under various operating situations. In this specific case it is likely that the functional specification did not provide a matrix of allowed and disallowed functionality by user, mode and access point.

The solution to prevent this problem leans again on the original vendor to mature their software development procedures and practices. However, at this point we see that it might be prudent for the client, the one who really understands how the equipment will be used, and the one at risk if it fails, to increase their level of oversight on project execution towards achieving the realization of the functional requirements to their needs. The client must understand the capability and limitations of the equipment they are buying. And if it fails to meet their requirements then they must work with the vendor to ensure that the vendor understands those requirements and adjusts not only the testing process but the core design as well. This is common practice in our industry for most engineering work, yet we have failed to incorporate software into this practice.

Regression

> Case 3: Well after client acceptance of the entire drilling package, a support engineer on site was tweaking the Top Drive torque calculations. When asked if he was testing, archiving, backing up and commenting his code, he replied that he was. Later the TD started to rotate and accelerate to extreme speeds causing damage to the equipment.

> Damage aside, he was asked to restore the code to its previous state. He could not, as he had failed to follow recommended testing and archiving procedures. It took a week to return the top drive to its former state and the acceptance test was never repeated.

This last case study is all too familiar in our industry. A technician arrives on the helicopter, walks into the control room, sets up shop and plugs directly into the control system. We all assume that someone else on the rig knows what he is there to do. His activities fall outside of our established Permit To Work, maintenance and safety practices, so we effectively ignore him. Later he proclaims the problem is solved and is on the next helicopter with the crew left scratching their heads; what did he just fix and how? And of course, we almost immediately notice that something that had been working is now no longer working. We are forced to call the technician back as we do not have access to a backup copy of the code he changed.

These situations beg the questions. Why do we treat a weld job with more precaution that a software engineer modifying our vessel and drilling control equipment code? Why do we take more ownership of mechanical components than we do of software? Why do we not pressure our vendors to prove that the code is tested and vetted after changes?

The drilling industry is traditionally a mechanically minded one. It's been that way for a long time and it has slowly changed over time with the introduction of hydraulic and electric controls. Control software has been the most recent adaptation. However we lag other industries as pointed out earlier. As an industry we are changing and adapting slowly to this new technology and, as part of that adoption, to the idea of automated rigs. Change does take time but the operators and contractors that drive the successful adoption of the change may prove to be more efficient.

This regression case points to our general over reliance on vendors for maintenance and support. This in not necessarily a bad thing. They built and designed the equipment. They are the software programmers and know and own the control code. However the software is on our rigs and at the end of the day the owner pays the price when it doesn't maintain day rate. The solution is for contractors to start recognizing software as an engineering discipline just like Mechanical and Electrical Engineering.

Current State of Controls Acceptance

We come full circle with the recognition that if operating companies and contractors do not drive change in the industry then they will be the ones bearing the brunt of quality problems touched on in the paper. It is evident that more respect and attention needs to be paid to software development and testing throughout the life cycle of the equipment. The most cost effective time to impact quality is during early

design. However quality can be improved regardless of the phase of the equipment. Even late in the commissioning or operation cycles.

Currently, however, the fashion of purchasing large new equipment is to accept turnkey operations from the lowest bidder and accept the risk of the quality during a final commissioning stage. This practice is in vogue over the OFE fashion of the last decade which may have proved costly for some contractors recognizing that their competencies lay elsewhere. Is the tradeoff of upfront savings vs. back end spending and safety concerns worth the risk? That is a topic in itself and we leave that to the reader to debate.

Immediate Corrective Actions

Recognizing that software is a critical component of our rigs and adapting our processes and procedures to account for it does not have to be a difficult or expensive undertaking. And given that a new build has or is about to come on line with unknown quality issues, what can operators and contractors do today to try to minimize the operational impact of hidden software problems? Fortunately there are small, incremental and immediate steps that any contractor can take.

Test, Inspect and Repair

Step one is to test, inspect and repair the equipment you are about to receive or already own. This is especially important during the warranty periods offered from shipyards where, while not free, it is certainly more cost effective to find and fix issues. Obviously the earlier issues can be found the better. It is very feasible and cost effective to implement a rigorous screen and functional verification and validation program regardless of your operating state. The result of this verification often leads to the identification of potential land mines like the ones above, as well as more benign quality issues such as the incorrect spelling of words like ‘paralell.’ This simple spelling mistake for example was found in 2006 and has yet to be corrected on the last new build that Kingston Systems inspected.

Build Software as an Engineering Discipline

Another tactical path would be to work towards the recognition of software as a critical asset and core to your business. We talked briefly of having software as an engineering discipline along with the others currently in place. A few contractors have started to move this direction with insourced or outsourced software project and equipment quality activities. One of the first and relatively simple milestones in this direction is to establish a Software Management of Change (SMOC) process. This establishes software access expectations and boundaries with vendors and employees. And just like the “Permit to Work” concept, if a change is planned, the executors must notify appropriate parties and plan accordingly. A basic software management of change process has the following tools:

- Notification and approvals of team leaders

- Consideration for backups and recovery of software
- Testing plans to ensure full functionality and avoid regression
- Version progression as well as an index of all software assets

Relatively easy to implement, and usually recognized as a much needed change by rig crews, these processes and tools establish basic levels of SMOC.

Prepare for the next project

This papers' final recommendation is perhaps the hardest. Once you have inspected and corrected the equipment that you have, and you have put in place measures to prevent inadvertent changes, the next step is to start planning for the next equipment order or update.

A good place to start might be with a regularly scheduled periodic inspection (SPS). An equipment upgrade or modification is a great starting point for a contractor to cut their teeth on resolving some of the problems outlined in this paper. The primarily solution starts with, you guessed it, the function requirement document. This relatively simple document defines what the buyer expects from the equipment in terms of what it should do and how. The how part of the document is key.

Rather than just focus on mechanical and electrical aspects, the contractor needs to consider control aspects and not leave this important aspect to the imagination of the programmer. One engineering method commonly used is a state diagram which illustrates all the potential normal (green) and restricted (red) operations that a piece of equipment can perform. With these red and green patterns defined it is then quite easy for programmers to develop applications that correctly meet interlocks to prevent inappropriate equipment access. As well it becomes a much more manageable task for testers to define test cases against the red and green paths that fully stress test the inputs and outputs of the equipment.

Like any engineering objective, software quality can be achieved through the application of engineering and project management best practices. Our industry is gradually beginning to accept software as a recognizable entity. And the leaders of tomorrow may be the contractors and operators that adjust to the management of this entity.

About Kingston Systems

Kingston Systems is an independent engineering consulting organization specializing in applied upstream technology and dedicated to improving client operational performance. One main area of focus is equipment commissioning and troubleshooting.