AADE-05-NTCE-41

Would you like software with that?

Don Shafer, CTO Athens Group, Inc.

This paper was prepared for presentation at the AADE 2005 National Technical Conference and Exhibition, held at the Wyndam Greenspoint in Houston, Texas, April 5-7, 2005. This conference was sponsored by the Houston Chapter of the American Association of Drilling Engineers. The information presented in this paper does not reflect any position, claim or endorsement made or implied by the American Association of Drilling Engineers, their officers or members. Questions concerning the content of this paper should be directed to the individuals listed as author/s of this work.

Abstract

In mid-2002, an extensive analysis was done on a vessel with badly flawed software systems. While in isolation the individual systems worked as advertised, when the entire onboard computer network was brought online, response time slowed dramatically. Affecting all onboard systems, this catastrophic time lag situation resulted from installing numerous pieces of highly sophisticated, automated equipment on board a rig without giving thought to how to integrate them through multiple network structures. The real lack of formal integration and testing of all the various systems is a crucial flaw for offshore drilling rigs and production platforms. Incredibly sophisticated systems that involve phenomenal life threatening safety issues are treated as mere collections of individual hardware modules with a little software glue mixed in. The increased complexity of control systems is steadily making this problem worse.

Introduction

"Would you like software with that?" is a real question that must be asked early in the project life cycle – front-end engineering design (FEED) is the place to start. Hardware is being sold and purchased without enough thought as to how it will be controlled and the overall architecture of the drilling system. This presentation looks at case studies of software problems present in all complex drilling and production rigs. This problem analysis was done by software engineers who have been involved in the verification, validation and software integrity management of drilling rigs in the Atlantic, Pacific, Caspian, Gulf of Mexico, and Africa. Actions to be taken for software risk mitigation will be presented.

“Ready” means “fit for purpose”. The determination of equipment being ready and fit for purpose is usually done by a classifying organization or an engineering firm. This fitness for purpose can be done during factory acceptance testing, pre-commissioning, commissioning and formal surveying. The process for equipment is well documented and well known. The process for software is poorly understood and executed in an anecdotal fashion.

Why do we care about software? – Cases of software failure. Four cases are discussed focused on drilling and vessel control software. The author and other software engineers within Athens Group have experience on many of the rigs where these incidents occurred.

Where do we stand with oil and gas (O&G) exploration & production (E&P) software? This is a basic question with respect to today’s fifth generation rigs, upgrades of high end technology to older rigs and platforms and the collection and rationalization of performance data across fleets of rigs.

Goals of software engineering are no different than the goals of other engineering disciplines. The focus is on the process for arriving at a correct set of software that meets the customer needs in a safe and cost effective fashion.

“Is there hope?” is the question that is asked by operators, drilling contractors and equipment manufacturers with respect to software. The answer is yes and the off shore O&G industry can take lessons from the work done to automate the semiconductor fabrication factories over the past 30 years.

Software engineering payback is quantified with a look at the opportunity cost in having a rig or production platform down for indeterminate amounts of time.

“Ready” means “fit for purpose”

Figure 1 shows the rig development life cycle from FEED to operation. Ensuring the software is ready requires the services listed in each life cycle phase to be in place. Two of the most used hardware analysis tools – though seldom used for software – are Failure Mode Effect Analysis (FMEA) and Failure Mode Effect and Criticality Analysis (FMECA) tools. These tools can be used throughout the life cycle and have an enormous potential for reducing risk. They are critical to getting the software “fit for purpose”.

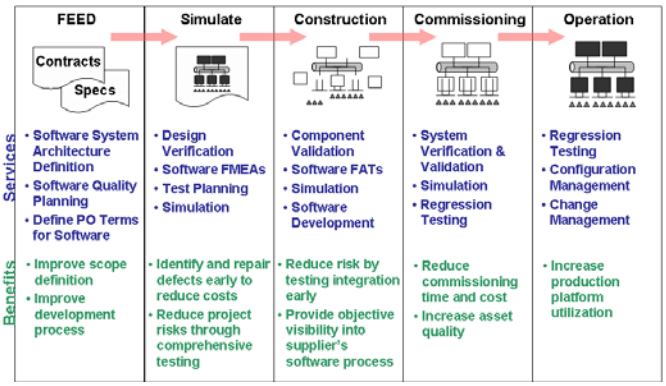

Figure 1: Rig Software Project Life Cycle

FMEA was originated in 1949 by the US Department of Defense. It was formalized for industry by the Society of Automotive Engineers as a reliability tool. The FMEA process is complementary to the design process for positive definition of a design's function to satisfy the customer. It was jointly developed by Chrysler, Ford and General Motors under the sponsorship of the United States Council for Automotive Research (USCAR). Formally documented by a standards body in 1994, Society of Automotive Engineers (SAE) J1739 introduces the topic of potential FMEA and gives general guidance in the application of the technique.

FMEA was originally designed to identify and provide a general description for individual elements/operations within a system that render it vulnerable; i.e. single point failures. FMECA added "Criticality" to the identification for specifying severity and determining probability assessments. Due to the virtual nature of software and the catastrophic results of failure in the offshore exploration and production arena, FMECA is the recommended approach. An FMECA can be described as a systemized group of activities intended to:

1) Define how this element can fail,
2) Identify what will happen to the system and its environment in each of the ways available to it (failure effects),
3) Recognize and evaluate the potential failure of a product/process and its effects,
4) Quantify the criticality of the failure event,
5) Identify mitigation factors could eliminate or reduce the chance of the potential failure occurring, and
6) Document the process.

The benefits of software focused FMECA are:

1) Discover potential single-point failures.
2) Assesses risk for potential, single-element failures for each identified target, within each mission phase.
3) Understand the threat of complex software system failure
4) Optimize reliability, hence mission accomplishment.
5) Guide design evaluation and improvement.
6) Guide design of system to "fail safe" or crash softly.
7) Guide design of system to operate satisfactorily using software of "low" reliability.
8) Guide component/manufacturer selection.

Why do we care about software? – Cases of software failure.

Case one is a result of no consistent FAT on the integration of pipe handling equipment. Case two concerns the testing of software fixes while in a production environment off shore. Case three looks at the failure of analyzing the performance characteristics of the network topology underlying drilling control and management software and equipment. Finally, case four looks at dynamic positioning failures.

Case 1: This is ready software?

Often the hardware factory acceptance test (FAT) is not designed to properly test the software. For example, one project we worked on had a test for a series of automatic control sequences – this was a piece of software that would build stands without driller intervention. The test was simply to get one length of pipe from the rack and do one make-up. There was no testing for most of the designed sequences, and no error-handling testing at all. During the test, the racker was bringing the pipe back from the fingerboards and was supposed to bring it over to a safe area. While it was doing this, one of the clients on the FAT reached down and pressed the emergency stop, just to make sure that worked, which it did. But when they started up again, the racker had forgotten it was supposed to be going to the safe area and went straight to the well centre, which would have killed anyone standing in the way. The client made the vendor test every sequence after that, which is what they should have done in the first place.

Case 2: This is process?

Hardware vendors' software maintenance personnel with laptops helicopter to rigs but cannot test changes except in real production. There is no test system for the software available on the rig nor adequate time to take all drilling equipment off line to make one fix.

A drawworks vendor sent one of their software maintenance technicians off shore to fix a problem. He hooked up his laptop and made the change, and then told the driller to try it. The drawworks code change caused bird's nesting. It took them hours to sort out both the cables and the code. The coder "felt" there was an if-test the "wrong way round". So he changed the code and the drill super said "How do I know it's going to work this time?" … They gritted their teeth - tried it - and luckily for the tech - it worked.

Case 3: Material Handling Under Software Control

The graph in figure 4 is of a set of timing observations taken on a pipe handler integrated into an automated drilling control and management system (DCMS). Two scenarios were tested: 1) the pipe racker ran autonomously with an operator controlling its operation from a control panel and joy stick connected directly to the PLC network within the handler, and 2) run from the drillers' cabin under the integrated software control system.

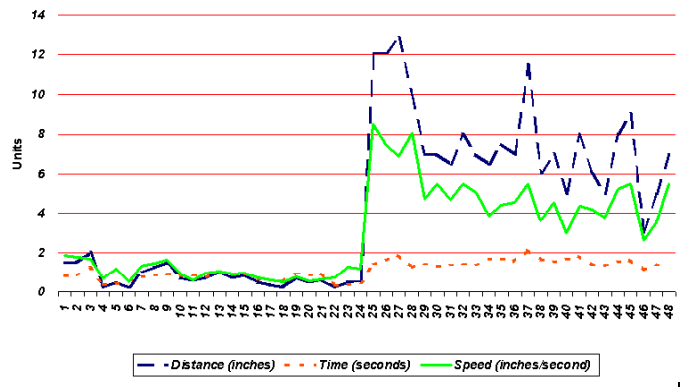

Figure 4: Autonomous versus DCMS Control

The observations were made over several days with time and distance measurements taken for equipment stop after being given the correct commands with various pipe loadings on the rack. Observations 1 through 24 were taken while the racker was under autonomous control. The remainder were taken while the racker was under control of the DCMS. The network latency contributed to the majority of the extended reaction time. The bottom line here is that under autonomous control the greatest distance that the equipment traveled was 2 inches. Under DCMS control the greatest distance was 12.5 inches.

Case 4: Drive Off

Drive-off is caused by a fault in the thrusters, or their control system, coupled with a fault in the position reference system. As for the drive-off incidents, the failure frequencies are derived from the DPVOA database, which consists of mainly DP Class 2 vessels. For the drive-off incident, these figures are representative also for the DP Class 3 vessels. The major difference between DP Class 2 and 3 is that only the latter is designed to withstand fire and flooding in any one compartment. These two incident types do not affect the drive-off frequency significantly due to the fact that drive-off is caused by logical failures in the thrusters, their control system and/or the position reference system. Hence, the DP Class 2 figures are combined with the DP Class 3 vessel.

"The frequency of fault in the thrusters and/or their control system can be obtained from report reference data. The control system is here meant to include all relevant software and hardware. The statistics show that the software failure is about four times as frequent as the hardware failure and slightly more frequent than the pure thruster failure.[1]"

The International Marine Contractors Association (IMCA) is the international trade association representing offshore diving, marine and underwater engineering companies. It was formed in April 1995 from the amalgamation of the International Association of Offshore Diving Contractors (AODC) and the Dynamic Positioning Vessel Owners Association (DPVOA). IMCA's Marine Division covers all aspects of vessel operations and marine equipment. Focusing on dynamic positioning; other key areas of interest are position reference equipment, reliability of systems and thruster-assisted systems.

As part of the above extensive work program, the division runs a DP operators' logbook scheme and each year publishes a report on dynamically positioned (DP) failure incidents. The division has, of late, expanded its work to include a broader range of specialist vessel operations, in particular focusing on offshore crane and other lifting equipment issues.

Since 1990, IMCA reports have demonstrated the risk inherent in DP system software. Figure 2 shows the published information on DP incidents with the percentage caused by software. With an eleven year average of 20%; 1 incident in 5 is caused by software, it would be assumed that any reasonably competent engineer would view the potential of a software problem as a high risk.

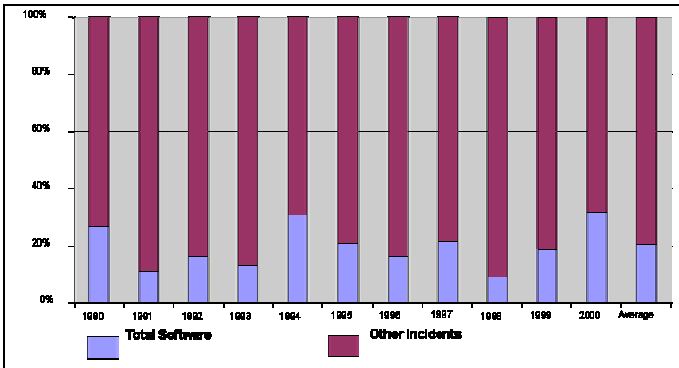

Figure 2: Software Incidents as Percent of Total

Figure 3 shows the percentage of "Loss of Position Class 1" DP problems that were caused by software, using data for the latest 5 years. This is the most serious class of DP incident that could result in "serious loss of position with serious actual or potential consequences". On average 33%; 1 in 3; of the most serious DP incidents were attributable to software problems.

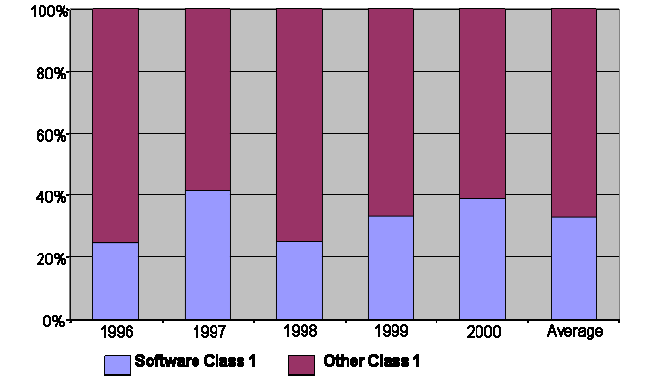

Figure 3: Class 1 - "Serious loss of position with serious actual or potential consequences."

Where do we stand with O&G E&P software?

These are the attributes of todays off shore drilling control system software:

1) Many vendors, multiple networks, no systems/software engineering
2) Incomplete, but incorrect, requirement sets
3) No enforced standards that all purchased components, including embedded Programmable Logic Computers, can communicate through existing, defined interfaces
4) Network topologies and protocols are incompatible
5) Single points of failure in the software and network subsystems are NOT identified
6) Early estimation of the network data throughputs and maximum loading scenarios is either lacking or inadequate
7) No early identification of overly complex control system designs that lead to impossible to test operational domains
8) No early estimates of the software development timeline and impacts on commissioning
9) No system models or standard equipment definitions against which to build a library of FAT, commissioning and regression test scenarios
10) No system or equipment models against which to rationally conduct software and network FMEAs
11) No baseline for comprehensive make versus buy decisions and the next system's contract
12) No life cycle model for system upgrade or work-over

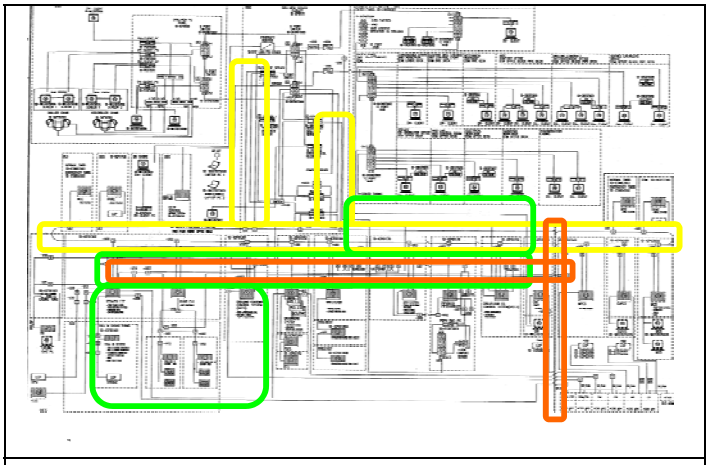

Figure 5: Typical Drilling Control System Topology

Figure 5 shows one of the reasons for the software problems. This is a typical topology for a drill floor control system. There are three Ethernet, two industrial Ethernet and two factory bus networks in this topology. There was no design done at the FEED to rationalize the networks' performance or balance out processing loads. It looks overly complex – and it is! It merely grew as each vendor added in their equipment.

Goals of Software Engineering

These are the goals of software engineering for off shore software:

1) Complete and correct set of requirements have

been collected
2) Requirements of all purchased components that include embedded Programmable Logic Computers (PLCs) can communicate with each other through existing, defined interfaces
3) Network topologies and protocols are compatible
4) Early identification of single points of failure in the software and network subsystems
5) Early estimation of the network data throughputs and maximum loading scenarios
6) Early identification of overly complex control system designs that lead to impossible to test operational domains
7) Early estimates of the software development timeline and impacts on commissioning
8) Development of a model system against which to build a library of FAT, commissioning and regression test scenarios
9) Providing a model against which to rationally conduct software and network FMEAs
10) Provide a baseline for comprehensive make versus buy decisions and the next system's contract

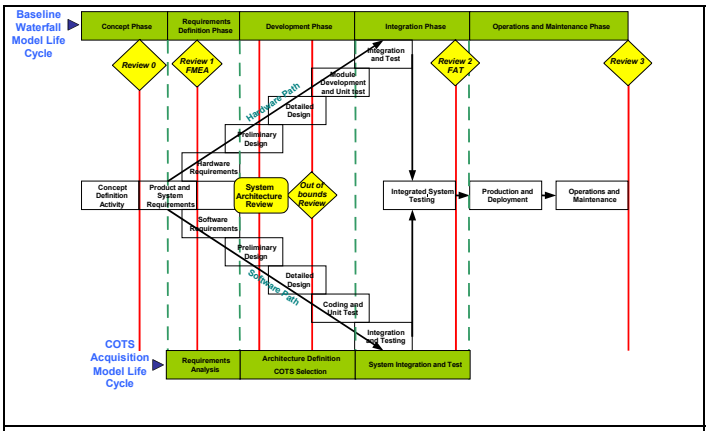

Figure 6: Software Engineering Life Cycle[2]

Figure 6 provides a software and hardware co-development life cycle that can be used as a framework to reach these goals. The software engineering life cycle maps easily on to the rig software life cycle of figure 1.

Is there hope?

Schedule delays, cost overruns, operational problems, and potential safety issues have plagued the installation, commissioning and start-up of large, integrated drilling control systems. A root cause of these problems is that the software interface between the computer controlled equipment (like Top Drives, Pipe Handling Equipment, Drawworks, and Iron Roughnecks) and the integrated control systems (like Cyberbase, Varco's Integrated Control and Information System (V-ICIS), and Drilling Management and Control System (DMCS)) is not standardized. That puts the onus on the integrated control system to resolve any differences in the location and meaning of the various software interface points. Since there can be anywhere from 2000-8000 such points, the potential for errors are great, and the time to discover and resolve the errors can be long. Worse, these errors are often only detected when the equipment is physically integrated during installation.

In contrast, using an example familiar to our industry, standards for grading drill pipe and classifying types of connections assure that drill pipe, regardless of manufacturer, can be effectively and safely integrated at the rig site and be fit for the purpose intended. There is no equivalent standard for drilling equipment. This situation is most acute when multiple vendors are used, but even exists when one vendor is used, since internally one vendor may have many product lines, and multiple products within those lines. The development of standards equivalent to those developed for drill pipe would address many of these issues.

Other industries, notably the automotive and semiconductor manufacturing industries, wrestled with exactly the same situation 10-15 years ago. In response, the semiconductor industry has developed a set of standards known as SECS (SEMiconductor Industry Equipment Communications Standard)/GEM (Generic Equipment Model) to allow fabrication equipment from multiple vendors to integrate seamlessly on the factory floor. These standards were derived from similar standards developed for the automotive industry. SECS/GEM standards implement functionality familiar to computer users as "Plug and Play".

An excerpt describing the functionality of GEM-compliant equipment demonstrates the potential applicability to drilling equipment:

"All GEM compliant manufacturing equipment share a consistent interface and certain consistent behavior. All may communicate with a GEM capable host using either Transmission Control Protocol/Internet Protocol (TCP/IP) (using the high speed messaging system (HSMS) standard, semiconductor industry (SEMI) E37) or RS-232 based protocol (using the SECS-I standard, SEMI E4). Often both protocols are supported. Each piece of equipment may be monitored and controlled using a common set of line management tools defined by GEM. When equipment has a GEM interface, it takes just minutes (or even seconds) for factory GEM host software to establish communication and begin monitoring the machine's activity. This means that equipment manufacturers may spend more time and

money improving the machine's quality by providing a common interface to all factories. It means that factories may spend more time and money improving production and processes, rather than setting up communication to the machines.[3]"

The experience in developing and implementing SECS/GEM standards and software tools for the semiconductor industry translated directly to analysis of major rig construction projects. These analyses involve integrated drilling systems, verifying the proper function of the integrated software systems, and identifying software and software development process problems. SECS/GEM standards can be effectively applied to computer controlled drilling equipment and their networked control systems. Like Plug and Play commonly seen on Windows personal computers, this will allow standards-compliant equipment to integrate easily on the rig floor, regardless of vendor.

The benefits to the operator and drilling contractor are a measurable reduction in risk during all phases of major rig construction projects - particularly in installation, commissioning and start-up. Potential safety issues arising from improper integration (such as a misreading of important alarms) would be minimized. Equipment vendors benefit from a clear standard to which they can manufacture their equipment. This minimizes miscommunication between buyer and seller regarding equipment functionality during the contract or implementation phases, which can lead to warranty issues later on. A test such as FAT and FMEA becomes more meaningful and consistent when the equipment can be tested to an established standard. Personnel required for installation and commissioning would be reduced, benefiting everyone.

The potential monetary impact per rig project could clearly be in the millions, the savings in time in months, and the avoidance of potential safety issues incalculable.

	Offshore Drilling Rig	Semiconductor Fab
Software Fails – People Die	Yes	Yes
Drills Wells	Yes - $10 * 10^3$ meters	Yes - $2 * 10^{-7}$ meters
Production Environment	Flying mud, 12 hour shifts, steel-toed boots	Clean Room, Positive Pressure, bunny suits
Cost of Software Failure	$100,000s/day	$10,000,000s/day
Automation Software	Plug and Prey	Plug and Play
Software Quality	No effort to measure	3.4 errors/million
Equipment Standards	No, not POSC , WITS nor WITSML	SEMI Standards begun in early 1970s
Software Process Model	NONE	SEI/CMMI
Years to Improve Quality	UNKNOWN	15

Figure 7: Semiconductor versus Off Shore Software

Figure 7 shows the differences and similarities between the offshore drilling rig and a semiconductor fab. Using this as a model is not as far fetched as first believed.

Software Engineering Payback

Based on off shore activities today, we can easily assume the following:

Average Daily Rate for 5th Generation Drilling Rigs: $154,022[4] per day, ~$77,000 per 12 hour shift

Production platforms pump 20,000 bbl per day with the sale price of Nymex Crude $41.25[5] per barrel

In startup, a rig could easily be down 10 shifts per week due to incomplete and incorrect software. This translates to a cost of $770,000 per week. This is a conservative number as it has been the author's experience on rigs with a day rate approximating $300,000 that they have missed their commissioning date by 30 to 120 days due to software problems.

Given the above production platform assumption, being down for 21 days would cause a loss of $17,325,000. Failure to address software engineering causes this magnitude of loss due to equipment and software failures.

Cost of doing a front end software and network Verification and Validation :< $100,000

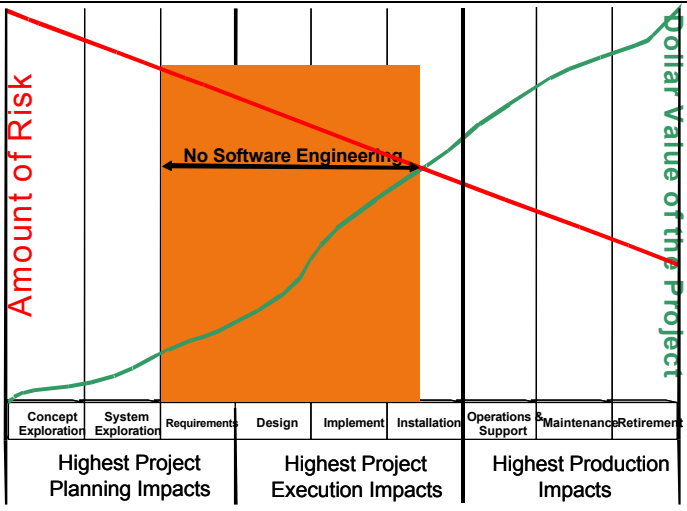

Figure 8: Software Engineering Risk Reward

Figure 8 shows the area where risk mitigation occurs – by plan or default – within a project life cycle. With little or no software engineering the risks become apparent

during requirements and the majority are resolved sometime into installation. The green line is the cumulative project cost and the red line is the amount of loss potential due to risk. Planning for risk early in a project will cause that red line to decrease at a more rapid rate thus crossing the green line sooner. The more rapid the risk opportunity for loss can be decreased the less the project will burn in overall resources.

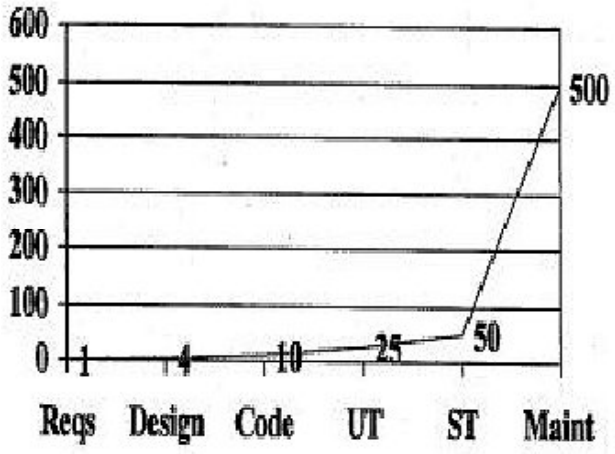

Figure 9: Cost to Fix a Line of Code[6]

Figure 9 shows the cost to fix a line of code in production at a manufacturing facility. This is an extremely well run semiconductor fab with continuous process improvement and the latest in process improvement techniques. The cost is still $500 to fix a bug that escapes into production. Finding the bug in requirements; e.g. defining the correct requirements up front and not waiting until production; costs one five hundredth the cost.

Conclusions

"Would you like software with that?" is a real question that must be asked early in the project life cycle - FEED is the place to start. Hardware is being sold and purchased without enough thought as to how it will be controlled and the overall architecture of the drilling system. Without out looking at the safety issues of mis-controlled equipment, the economic consequences are of such a magnitude as to warrant changes. What does it cost off shore? An estimate of $5,000 per line of code changed is much closer to reality than the manufacturing industry high now used of $500.

“Ready” means “fit for purpose”. Today there is no consistent, demonstrable set of equipment that uses software that is ready for its task. The software is not engineered for integration with all the equipment running on an automated drill floor. The underlying networks are not analyzed for performance before installation and FMEAs are not conducted on the aggregation of equipment.

There is hope for reducing the cost of start up, maintenance and refit. This is based on the application of known software engineering processes and procedures, standardization of equipment interfaces and adoption of software as a critical path item early in the rig development life cycle.

Acknowledgments

The author would like to thank the following colleagues for ideas and experiences contributed during the past many years of off shore software engineering analysis:

Keith Womer – KW Technologies, a major contributor to the working off shore DCMS software.

Tim Blatch – hardware classifier and surveyor par excellence.

Mike Christie, Pat Flanagan, Tom McAfee and Dave McGriffy – software engineers who put up with helicopter egress training to get to spend weeks on rigs in Singapore, Brazil, GoM and Angola.

Nomenclature

AODC	=	*International Association of Offshore Diving Contractors*
DCMS	=	*Drilling Control Management System*
DP	=	*Dynamic Positioning*
DPVOA	=	*Dynamic Positioning Vessel Owners Association*
FAT	=	*Factory Acceptance Test*
FEED	=	*Front End Engineering Design*
FMEA	=	*Failure Mode Effect Analysis*
FMECA	=	*Failure Mode Effect Criticality Analysis*
GEM	=	*Generic Equipment Model*
GoM	=	*Gulf of Mexico*
HSMS	=	*High Speed Messaging System*
IMCA	=	*International Marine Contractors Association*
SAE	=	*Society of Automotive Engineers*
SECS	=	*SEMiconductor Industry Equipment Communications Standard*
SEMI	=	*Semiconductor Industry*
TCP/IP	=	*Transmission Control Protocol/Internet Protocol*
USCAR	=	*United States Council for Automotive Research*
V&V	=	*Verification and Validation*
V-ICIS	=	*Varco’s Integrated Control and Information System*

References

[1] KONGSBERG OFFSHORE A.S., DEMO 2000 - LIGHT WELL INTERVENTION HSE EVALUATION. REPORT NO. 00-4002, REVISION NO. 00
[2] Futrell, Robert, Donald F. Shafer and Linda Shafer; Quality Software Project Management, Prentice Hall:Upper Saddle River, NJ, 2002
[3]Cimetrix, An Introduction to the GEM Standard, http://www.cimetrix.com/gemintro.cfm)
[4] Updated at 1330hrs, 16 July 2004, www.rigzone.com
[5] Updated at 1330hrs, 16 July 2004, www.bloomberg.com
[6] IBM Manufacturing data, East Fishkill Fab, 2002