Dynamic
Positioning
Committee
Marine Technology Society

TECHNICAL AND OPERATIONAL GUIDANCE (TECHOP)

TECHOP (D-08 - Rev1 - Jan21)

SOFTWARE TESTING

JANUARY 2021

DISCLAIMER

The information presented in this publication of the Dynamic Positioning Committee of the Marine Technology Society (‘DP Committee’) is made available for general information purposes without charge. The DP Committee does not warrant the accuracy, completeness, or usefulness of this information. Any reliance you place on this publication is strictly at your own risk. We disclaim all liability and responsibility arising from any reliance placed on this publication by you or anyone who may be informed of its contents.

CONTENTS

FIGURES

TABLES

1 INTRODUCTION

1.1 PREAMBLE

1.1.1 The guidance documents on DP (Design and Operations and People) were published by the MTS DP Technical Committee in 2011, 2010 and 2012, respectively. Subsequent engagement has occurred with:

- Classification Societies (DNV, ABS)
- United States Coast Guard (USCG)
- Marine Safety Forum (MSF)
- Oil Companies International Marine Forum (OCIMF)

1.1.2 Feedback has also been received through the comments section provided in the MTS DP Technical Committee Web Site.

1.1.3 It became apparent that a mechanism needed to be developed and implemented to address the following in a pragmatic manner.

- Feedback provided by the various stakeholders.
- Additional information and guidance that the MTS DP Technical Committee wished to provide and a means to facilitate revisions to the documents and communication of the same to the various stakeholders.

1.1.4 The use of Technical and Operations Guidance Notes (TECHOP) was deemed to be a suitable vehicle to address the above. These TECHOP Notes will be in the following categories:

General	TECHOP (G)
Design	TECHOP (D)
Operations	TECHOP (O)
People	TECHOP (P)

1.2 TECHOP NAMING CONVENTION

1.2.1 The naming convention, TECHOP (CATEGORY (G / D / O / P) – Seq. No. – Rev.No. – MonthYear) TITLE will be used to identify TECHOPs as shown in the examples below:

Examples:

- TECHOP (D-01 - Rev1 - Jan21) Addressing C^3EI^2 to Eliminate Single Point Failures
- TECHOP (G-02 - Rev1 - Jan21) Power Plant Common Cause Failures
- TECHOP (O-01 - Rev1 - Jan21) DP Operations Manual

Note: Each Category will have its own sequential number series.

1.3 MTS DP GUIDANCE REVISION METHODOLOGY

1.3.1 TECHOPs as described above will be published as relevant and appropriate. These TECHOP will be written in a manner that will facilitate them to be used as standalone documents.

1.3.2 Subsequent revisions of the MTS Guidance documents will review the published TECHOPs and incorporate as appropriate.

1.3.3 Communications with stakeholders will be established as appropriate to ensure that they are notified of intended revisions. Stakeholders will be provided with the opportunity to participate in the review process and invited to be part of the review team as appropriate.

2 SCOPE AND IMPACT OF THIS TECHOP

2.1 SCOPE

2.1.1 MTS TECHOP (D-02 - Rev1 - Jan21) FMEA TESTING covers all aspects of testing associated with proving a DP vessels redundancy concept and fault tolerance. This TECHOP on software testing is intended to highlight the particular issues associated with this branch of the verification process. FMEAs performed on DP vessels typically exclude software from the analysis.

2.1.2 This TECHOP considers the need for software testing for systems associated with computer based control systems that may affect dynamic positioning systems including:

- DP control.
- Power management.
- Vessel management.
- Safety systems.
- Interfaces with industrial equipment.
- Steering, propulsion and thrust.

2.1.3 The principles apply equally to any safety or mission critical systems.

2.1.4 The scope of this TECHOP is software testing but it is widely recognised that the effectiveness of testing as a risk reduction measure is limited if the target software product is not developed within an effective quality control and verification scheme. Such schemes are available from the major classification societies. The structured approach to such verification processes is included in the scope of this TECHOP.

2.1.5 The guidance considers current practice and the gap between current practices and expectations in relation to station keeping integrity, efficient and incident free execution of the industrial mission. The distinct contrast in the low visibility of software testing with the significant efforts applied to hardware failure simulation that are a well-established part of the class approval process is starkly apparent.

2.1.6 The availability of tools for software testing is reviewed and opportunities for improvement identified.

2.2 IMPACT ON PUBLISHED GUIDANCE

2.2.1 This TECHOP impacts Section 25 of the MTS DP Vessel Design Philosophy Guidelines.

3 CASE FOR ACTION

3.1 DP INCIDENTS

3.1.1 All modern dynamically positioned vessels employ software based control and monitoring systems in a wide variety of applications from the DP control system itself to vessel management, power management and engine control systems. Even governors and voltage regulators for generators are numerically controlled as are the relays that protect the main switchboards in diesel electric power plant. Extensive use is made of PLCs for control of DP and industrial mission equipment such as drilling and pipelay systems.

3.1.2 The IMCA Station Keeping Incident Database for 2010 (M 218) shows that 11% of all DP incidents were associated with computer problems with the most significant sub-group being related to computer software (See Figure 3-1.)

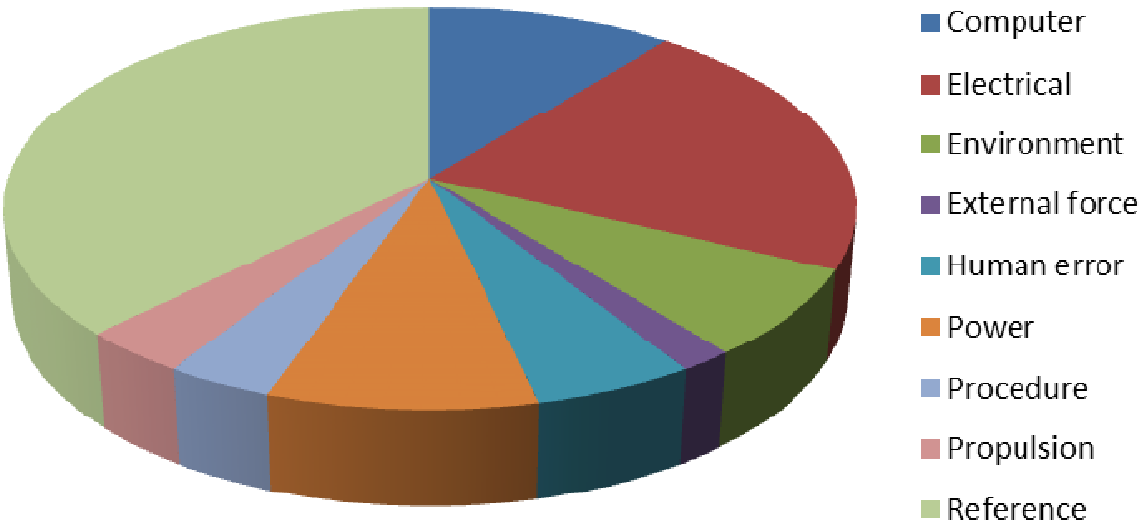


Figure 3-1 From M 218 Station Keeping Incident Database for 2010 – (Courtesy IMCA)

3.1.3 The largest number of incidents recorded was related to position reference issues. The next most significant causes were electrical, computer and power. Human error, environment, propulsion and procedures are the least significant causes. However, as software forms a part of so many systems it is likely to be a factor in categories other than computer systems and review of previous incident records by software professionals suggests the actual number of incidents where software was a factor is closer to 19%.

3.1.4 This database does not record those software issues that affected the execution of the industrial mission without affecting DP but there is no reason to believe the situation is any better in this category. Note: Although this TECHOP addresses DP critical equipment MTS guidance attempts to retain focus on the industrial mission as the reason for using DP as a station keeping method.

3.2 TYPICAL SOFTWARE DEVELOPMENT AND REVISION CONTROL ISSUES

3.2.1 The following list of issues is well known within the DP community. The most significant problems tend to be with development of bespoke software for novel applications.

3.2.2 Insufficient evaluation of automation scope:

- Vessel owner’s expectations for the scope of bespoke automation software are unrealistic.
- Shipyard does not recognise that owner’s expectations are unrealistic and puts owner’s scope out to tender without adequate emphasis.
- Experienced automation suppliers may recognise the true level of difficulty in the scope but reflecting this in their commercial offer makes them an unattractive option.
- A less capable automation supplier may win the contract and hope to fill their resource gap during project execution.
- Unforeseen problems develop during execution.
- The scope cannot be completed in the available time.
- The shipyard negotiates to deliver the vessel with a partial implementation of the scope.
- The vessel does not meet expectations but goes to work using what functionality it has supplemented by temporary work-arounds and other procedures.

3.2.3 Poor control of integration process:

- Project specification calls for the integration of systems from several suppliers.
- The importance of the integration roles is underestimated and poorly executed.
- Contracts establishing the role of vendors in the integration and verification phases are inadequately defined and not communicated to all stakeholders.
- The project fails at the integration stage.

3.2.4 Inadequate software revision control:

- A DP vessel requires additional functionality to address a new industrial mission or to rectify deficiencies in the original control systems.
- The automation supplier attends the vessel and installs the software with the new functionality.
- The new functionality is regression tested and found to work satisfactorily.
- The vessel’s return to service is delayed when it is discovered that the new functionality has been added to an old revision of the base software release and does not include all the updates that have been installed to correct earlier problems.

3.2.5 Poor control of settings and parameters:

- A DP control system is upgraded to a new version of software.
- The normal settings are replaced by default settings.
- The default settings are not suitable for a vessel of that type and displacement.
- The vessel suffers a DP incident when the default settings reject all the position references simultaneously.
- Inadequate fail safe mechanisms built into the DP application software so that when a failure or forced restart occurs, the system returns to a known safe state at restart or recovery.
- Lack of availability in the office (development team) of the actual hardware and system mock-up for the software engineer /software tested to verify and validate the software changes against.

3.2.6 In general, software control is vulnerable to changes performed on the vessel which do not make it back to the Software Configuration Management (SCM) repository in a timely fashion to update the code repository.

3.3 NEED FOR AN INTEGRATED APPROACH TO TESTING

3.3.1 This TECHOP on software testing recognises that testing is only one part of the verification process and that testing of software cannot be completely disassociated from testing the hardware that software controls and the entire system must be the testing target. There are opportunities to test software in isolation before hardware is available and in a manner that it not easily replicated once the target system is commissioned and operational. This section includes a discussion of the general principles of testing before focusing on the practice of software testing in particular.

3.4 ADVANTAGES OF SOFTWARE BASED CONTROL

3.4.1 The practice of implementing software control of systems is now almost universal in all but the simplest or most specialist applications. The advantages of software based control systems include:

- Flexibility to add new functionality and correct errors.
- The precision and repeatability associated with software control.
- The ease with which complex mathematical control algorithms can be implemented.
- Ease with which a backup can be created and stored.
- Opportunities to include diagnostic and self-test functions.
- Ease of parameterisation and tuning (self-tuning).
- Ease of replication.
- The ease with which remote access can be provided for troubleshooting and maintenance.
- The ease with which alarms, monitoring, remote manual control and data logging can be implemented as parts of the control systems.
- Adaptability to run on a variety of hardware platforms and operating systems – not limited to one hardware supplier.
- Ability to test to a certain extent without the physical systems it is intended to control being present.

3.4.2 Unfortunately, the greatest strengths of software based control systems also create significant vulnerabilities and it is not unsurprising that many problems are associated with the misuse of these attributes.

4 CURRENT PRACTICE

4.1 LIMITATIONS OF FMEA

4.1.1 MTS TECHOP (D-02 - Rev1 - Jan21) FMEA TESTING provides general details on the philosophy of testing DP systems by FMEA.

4.1.2 Software testing in the DP community is largely performed by equipment vendors on their own equipment during the development and commissioning phases. A limited amount of software functionality is tested at the DP FMEA proving trials. Testing of fully integrated systems is limited to demonstrating functionality and failure response during DP FMEA proving trials.

4.1.3 From the perspective of software, the DP FMEA will exercise certain discrete capabilities of the software hence validating only a small portion of the functionality. FMEA testing as it has been developed in the DP community focuses on design flaws and construction defects that could defeat the redundancy concept. Probability is still a factor in so far as failure effects in redundant systems may not be entirely predictable as hidden failures may go undetected. Protective functions may fail to operate effectively on demand. In contrast, a failure in software is inherently systematic, without a stochastic component. This is why the FMEA should not be relied on to validate the software.

Note a stochastic system is one whose state is non-deterministic (i.e. random) so that the subsequent state of the system is determined probabilistically.

4.2 SOFTWARE TESTING

4.2.1 When software testing is carried out it is generally performed on selected systems in discreet phases when the software has reached a suitable point in its development. This point may align with the Factory Acceptance Test (FAT) of the associated control system hardware. However, there will be a need to capture any changes that are made during the commissioning of the control system when it is installed onboard the DP vessel. Such changes need to be carefully managed and the system tested again. Software changes should be tested on shore at the manufacturer’s facilities to the best of their abilities prior to being deployed to a vessel in operation or under construction in a shipyard.

4.2.2 Software testing as part of Hardware in the Loop (HIL) is carried out on some DP vessels in service on an annual basis and can help to reduce the risk of DP incidents associated with parameter changes or software updates. In general, software should be tested any time there is a change to the control logic. Attributes of critical variables within the control logic change the performance specification of the system.

4.2.3 The most compelling reason to test software and software changes is lack of transparency. For a simple system (e.g. pump) the impact of a change is intuitively obvious and there is less incentive to execute an exhaustive test. For complex systems, where the impact of a small change is not obvious, the way to validate that change, so as to ensure it has the required impact and no adverse effects, is to perform a thorough design review and test to validate it. Verification and validation of software, in the DP industry is even more challenging as code reviews for software changes are not common place and typically resisted. As a result, with no visibility into the details of the software change, there is a need to place more focus on testing for the validation effort.

5 THE GAP BETWEEN PRACTICE AND EXPECTATIONS

5.1 EXPECTATIONS

5.1.1 The expectation of DP vessel owners and their clients is that a DP system designed to meet the requirements of notations equivalent to IMO DP Equipment Classes 2 & 3 is fully fault tolerant in respect of the defined failure criteria. It is expected that compliance with these requirements ensures an adequate level of safety to mitigate the risks associated with using a dynamically positioned vessel to carry out operations.

5.2 THE GAP

5.2.1 The gap includes:

- DP FMEA proving trials do not currently represent a comprehensive test of all aspects of the DP system upon which the redundancy concept depends, in particular software. Software and computer system faults are responsible for a significant percentage of the total number of reported DP incidents and an unknown number of issues with industrial equipment where there is even less regulation and scrutiny.
- A large amount of testing is carried out during FAT, commissioning and sea trials, some of which would satisfy the verification requirements of the redundancy concept. Unfortunately, in most projects, there is no central record of what has been done in a format that allows it to be used for that purpose. Personnel witnessing such activities may have a different focus from that required.
- Software development does not have the same level of scrutiny currently afforded to hardware and systems testing by the present DP FMEA and proving trials regime.
- Software testing beyond that performed by the developer to satisfy his own quality processes is currently viewed as an optional extra and uptake of testing and verification schemes is low.
- The range of testing tools available to carry out hardware and software verification at testing opportunities is adequate to meet current custom and practice but not to provide any significant improvement in the process.
- There are no standards that focus on software testing currently in place.

5.3 CLOSING THE GAP

5.3.1 The gap between current practice and expectation can be closed by adopting the following strategy:

- Develop a holistic approach to the subject of testing in general that recognises that this is an indispensable part of proving the redundancy concept of a DP vessel and the reliability of its industrial equipment where appropriate.
- Evaluate the benefits of software quality assurance schemes that are available from the major classification societies and other organisations. Implement its procedures and processes. These schemes are generally aimed at new build projects. The risks introduced by lack of adequate software management controls should be addressed by suitable risk reduction measures on vessels in service.
- These schemes are of particular value where the target application has bespoke elements; novel features or requires significant integration effort between multiple system suppliers.

- Evaluate software testing methods. Different methods may be suitable at different times in the project. Software testing carried out throughout the development stage is likely to pay dividends in terms of more efficient commissioning. Software testing which is not part of an overall software development strategy is less valuable. This is because there are practical limits to what can be tested in a very complex system even with sophisticated simulators. A two pronged approach which recognises that it is better not to produce software errors in the first place than to find them during testing is likely to produce the best results.
- Develop a testing strategy that starts with testing of the concept at the basic design stage and expands to include all available test opportunities at FAT and commissioning. Because software testing can be carried out without the hardware it controls, this process can run in parallel with other phases of the build program.
- Having completed what testing can be conducted in other phases of the build program identify those elements that must be tested or retested with the vessel on full auto DP in its defined power plant configuration. The justification for this should be based on the criticality of the system or function to the DP redundancy concept and is likely to encompass all protective function tests upon which redundancy depends. Exemption to test when on full auto DP may be considered on those functions that are entirely software based and that act on data of known integrity that has been verified as part of the software trials program provided they have been fully tested as part of the software testing process using the current software release and operational parameters.
- Ensure the availability of test methods and tools to supplement traditional DP FMEA proving trials methods to allow more realistic testing.
- Provide effective guidance on a software test process for a generic DP system, i.e. work towards a standardised approach.

5.3.2 Much of the above process is not yet part of the mainstream approach to DP vessel construction and verification and requires commitment and engagement of many stakeholders. However, almost all the elements required to put the process into practice are already available and some with considerable maturity in DP vessel design and other applications. These processes are discussed in the sections that follow.

6 SOFTWARE TEST METHODS

6.1 ACKNOWLEDGEMENT

6.1.1 The discussion and figures which follow are taken from material provided by classification societies and commercial organisations involved in software testing. Contributions have been taken from class rules, recommended practice and white papers in order to provide a summary of what is a detailed subject. Readers are directed to the section on further reading at the end of this TECHOP for references to the original works from which this material was taken.

6.2 SOFTWARE TESTING BACKGROUND

6.2.1 Software testing as part of overall quality assurance is well established in other industries such as automotive, aerospace and defence. Most major marine control system suppliers have quality control procedures in place but an increasing amount of bespoke software development is done to address specific control system requirements.

6.2.2 In large DP vessel projects, there can be a need to integrate many computer controlled systems from several different suppliers such as power and vessel management systems, fire and gas systems, emergency shutdown systems etc.

6.2.3 Control system software is an essential and integral part of all modern DP vessels. From small embedded control systems on panels and sensors to the complex DP control system. Software constitutes an inherent part of a multitude of safety and mission-critical automation systems.

6.2.4 The shipbuilding industry differs from automotive and aerospace in so far as it does not benefit from the economy of mass production to the same extent.

6.2.5 Although shipyards increasingly promote stock designs, the owner’s preference for a particular DP control and automation system provider, power system vendor and supplier of industrial equipment introduces sufficient variation to ensure it is unlikely that more than a handful of vessels share the same design. In practice, equipment obsolescence and hardware and software development during the build cycle means none of these will be truly identical.

6.2.6 Software development is as vulnerable to human error as any other process. In the worst cases, software functions are written and configured under tight time schedules, often in bad working conditions on board the vessel and late in the project. Integration of systems from different vendors is challenging and error prone, often with interfaces between multiple systems and integrated/distributed functionality. The probability of software defects increases with complexity.

6.2.7 Several of the typical vessel control systems introduce a dependency between the redundant groups. This also applies to many of the systems that have a dedicated controller for each redundant group. The dependency and potential for common cause failures exists because the control system introduces communication between equipment from different redundancy groups. It can no longer be ruled out by inspection only, that a failure in one redundant system cannot be transferred, via control system handling or failed handling, to the other redundant group. The fundamental problem of software is that any input signal may affect any output or group of output signals. Bugs in the software might introduce unintended cross dependencies between otherwise redundant groups.

6.2.8 International standards for software development and verification have been produced; classification societies such as ABS and DNV recognised the need to provide a process for systematically addressing elements of software development and integration that are required to ensure adequate traceability. These standards require the assignment of roles and responsibilities for the overall integration process including the role of systems integrator and independent verifier. On completion, a class certificate confirming the successful conclusion of the integration process is issued for the vessel.

6.2.9 The DP system FMEA of the various software components, on which the overall vessel FMEA relies, is usually undertaken by the control system vendors themselves without third-party testing and verification. In the FMEA proving trials, which constitutes an important part of the sea trials for a new-build, focus is placed on the hardware components and partly the Input Output (IO) layer of the computer systems. The software functionality of the computer systems is only superficially tested, mainly due to a lack of appropriate testing tools. The problem is further compounded for aging systems. In order to properly support a testing effort for the vessel's life-cycle, the software developers need to have the legacy tools and systems maintained so that the software changes can be validated.

7 REVIEW OF EXISTING SOFTWARE TEST METHODOLOGIES

7.1 DISCLAIMER

7.1.1 The MTS DP Committee and MTS Sub Committee on Guidance and Standards clearly and unambiguously state that they do not comment or hold a view about the independence of the assurance/verification process for software.

7.1.2 Requirements for independence, or acceptance of dependence are set by owners or their clients.

7.1.3 Content contained in the TECHOP in general and specifically in Section 7.6 and 8.3, encompasses views of proponents of both viewpoints; namely, independent 3rd party verification as well as dependent verification as described in this document. Comments have been included to foster inclusiveness and capture diverse viewpoints (to the extent feasible).

7.2 CURRENT PRACTICE

7.2.1 Established control system manufacturers typically have processes for software development, quality assurance and testing which are derived from recognised software industry standards. However, there are significant numbers of less well established engineering companies offering products which include a significant software element. Ship owners and operators may have little visibility of this process other than through scrutiny of quality system certification by auditors etc.

7.2.2 Quality assurance processes considering the vessel and all its computer controlled systems as a whole are available but uptake in industry is not widespread.

7.2.3 Software maintenance is the process of change initiated by faults, design improvements, changed requirements or changed hardware. Software maintenance quality assurance provides guidance for management of software and should be used in dynamic positioning systems, active mooring systems and position measurement equipment. The purpose is to ensure that the computer controlled systems are of known quality such that any fault rectification, modification, upgrade or other work is fully compatible with the system as originally installed or as previously modified or upgraded.

7.2.4 This type of quality assurance complies with the following rules:

- ISO 9001:1994 (Ref 1) provides the model for quality systems.
- ISO 9000-3:1997 (Ref 2) provides guidelines for the interpretation and implementation of ISO 9001 (Ref. 1) for software and is the basis of these guidelines.
- ISO/IEC 12207 (Ref 5) defines key life cycle processes, some of which are applicable to the close manufacturer–owner/operator relationships involved in the development and use of the subject products.
- ISO 1007:1995 Information Technology – Software life cycle processes.
- The TickIT Guide: A guide to software quality system construction and certification using ISO 9001:1994.
- ISO/IEC/IEEE 29119: Software Testing Standard. Covers software testing concepts, processes, documentation, techniques.

7.3 RULES AND GUIDELINES

7.3.1 Guidance on software testing and quality assurance can be found in:

- IMCA M 163, ‘Guidelines for the Quality Assurance and Quality Control of Software’, September 2001.
- IMCA M 15/12, ‘Programmable Logic Controllers’, October 2012.
- ABS-Guide for System Verification (SV).
- DNV-Enhanced System Verification (ESV), Rules for classification of Ships Part 6 Chapter 22.
- ABS-Guide for Integrated Software Quality Management (ISQM).
- DNV-Integrated Software Dependent Systems (ISDS), DNV-OS-D203.
- DNV-Standard for Certification (SfC), Hardware in the Loop Testing (HIL), No.2.24.
- SEI – Capability Maturity Model, (CMMI-DEV 1.3) – CMMI for Development.
- SWEBOK V3 – Software Engineering Body of Knowledge (Chapter 4 Software Testing).

7.4 HARDWARE - IN–THE-LOOP, SOFTWARE-IN-THE-LOOP & ENDURANCE TESTING

7.4.1 Hardware in the Loop Testing: This is a software verification process inherited from other industries. In marine and DP applications a mathematical model of the vessel and its propulsion system is interfaced with the real control system so that the effects of failures can be observed in the absence of the actual vessel.

7.4.2 The concept of HIL is testing and verification of the computer software using a vessel specific simulator, Figure 7-1, capable of simulating the dynamic response of the vessel, thruster and propulsion system, sensors, position reference systems, power generation, distribution, main consumers, and other relevant equipment.

7.4.3 HIL testing may be conducted in several phases of a new-building or retrofit, where the first phase is usually an extensive software test conducted at the factory or a lab facility. By using HIL simulator technology a virtual sea trial with thorough testing is conducted before the vessel is built. The objective is full functional and failure testing of the software before the commissioning and integration starts, building confidence that the software will be more finalized and ready for commissioning. Follow-up system and integration testing is normally conducted during commissioning and a final verification of the integrated functionality is conducted on board the vessel at the end of commissioning.

7.4.4 The simulator is connected via network or bus interfaces to the targeted control system such that all relevant feedback and command signals are simulated in response to command signals from the control system. In order to achieve the test objective, the simulator is capable of simulating a wide range of realistic scenarios defined by operational modes, operational tasks and single, common mode and multiple failure modes in order to verify correct functionality and performance during normal, abnormal and faulty conditions.

7.4.5 HIL testing is accomplished by isolating the control system and its operator stations from its surroundings and replacing all actual I/O with simulated I/O from a HIL simulator in real time. The control system cannot sense any difference between the real world and the virtual world in the HIL simulator. This facilitates systematic testing of control system design philosophy, functionality, performance, and failure handling capability, both in normal and off-design operating conditions. HIL testing only superficially tests hardware and redundancy, since the main test target is software. The main common objective of both FMEA and DP HIL testing is to verify the vessel's ability to maintain position after a single failure. Another important objective of HIL testing is to verify the control system software functionality is as specified by the supplier and possibly the end user of the DP system.

7.4.6 From an industrial mission point of view, the same tendency is driven by the fact that control system safety and performance testing by hardware-in-the-loop (HIL) simulations contribute to reduced time for tuning during commissioning and sea trials and, not least, reduced risk of incidents during operation caused by software bugs and erroneous control system configurations.

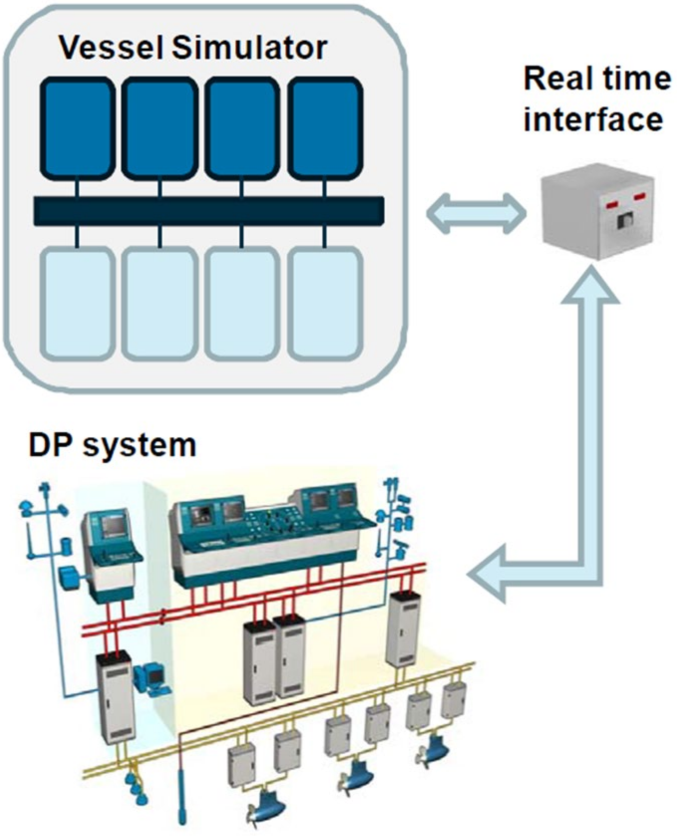


Figure 7-1 HIL Testing

7.4.7 Software-in-the-loop (SIL) does not verify the networked connections, input or output hardware modules or the processor. The configuration consists of an exact copy of the control system software running on a machine that emulates the hardware controller closely and another machine that simulates the plant. PCs are often used providing flexibility and relatively cheap simulator environments compared to the simulated system configuration. This SIL simulation can be achieved by various configurations using several PC machines. The simulated plant and control model can be loaded on one PC machine, which is connected to the other PCs and has the Man Machine Interface program.

7.4.8 Configuring a software-in-the-loop simulator requires programming the controller model as well as the plant model. Using a proper simulation tool for this purpose provides many advantages for time and modelling efforts. The desirable combination of simulation tools uses a continuous process simulation tool for the plant model and a computer aided control system design (CACSD) tool for the controller model.

7.4.9 The advantages of a SIL system include low equipment cost, flexibility, and portability due to the use of readily available hardware and software, without the need of a high-cost replica of MMI and controllers such as a distributed control system (DCS).

7.4.10 This TECHOP does not address SIL beyond this reference.

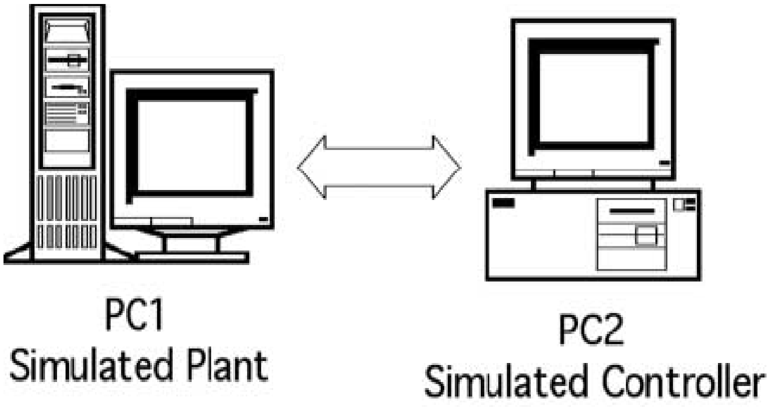


Figure 7-2 Typical Software in the Loop

7.4.11 Endurance testing: The term ‘endurance testing’ or ‘soak testing’ is commonly associated with hardware testing but is also performed on software.

7.4.12 Endurance testing may be used to provide an estimate of how long software can operate without crashing due to memory leaks, resource capacity issues, bandwidth limits etc. During such tests, memory utilization is monitored to detect potential memory leaks and performance degradation is monitored to ensure that the response times and throughput does not vary significantly with time. The goal is to discover how the system behaves under sustained use.

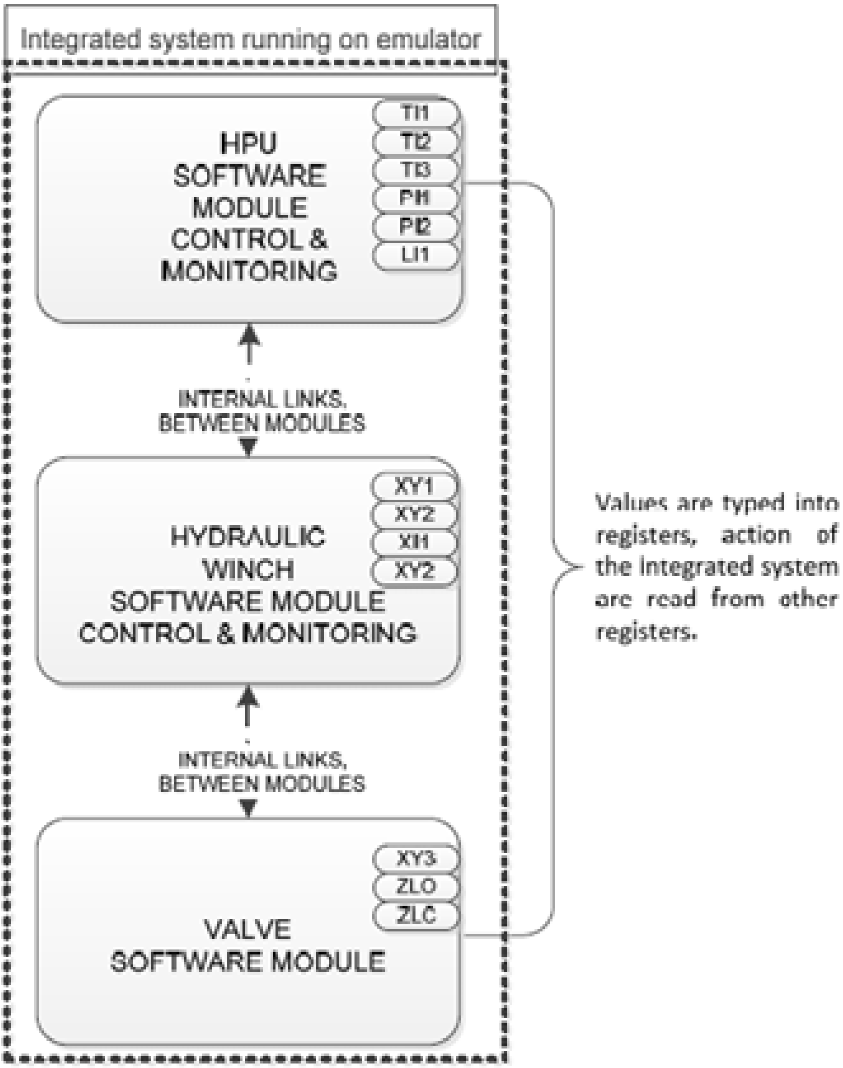
7.4.13 Although software faults are normally associated with design issues and software itself does not deteriorate with the passing of time it is possible to collect similar reliability metrics to those used in hardware testing.

7.4.14 In certain DP applications the controllers running the DP software are expected to perform reliably for many months and therefore knowledge of long term performance is essential.

7.5 CLOSED LOOP VERIFICATION

7.5.1 See Figure 7-3. Closed loop verification is acceptable with less complex control systems. In closed loop verification, detailed knowledge of the process and programming is necessary to verify the correct operation of the software. The data registers are interpreted to verify the integrated system's response is per the specifications.

7.5.2 The inputs and outputs of the computer-based integrated system are simulated with minimal interaction of the other integrated components. Closed loop verification may require changing register values of the program to evaluate the integrated system software response. A comprehensive understanding of the software code and its functions are required and this limits the application to simple systems.


Integrated system running on emulator
HPU SOFTWARE MODULE CONTROL & MONITORING
TI1
TI2
TI3
PI1
PI2
LI1
INTERNAL LINKS, BETWEEN MODULES
HYDRAULIC WINCH SOFTWARE MODULE CONTROL & MONITORING
XY1
XY2
XI1
XY2
INTERNAL LINKS, BETWEEN MODULES
VALVE SOFTWARE MODULE
XY3
ZLO
ZLC

Values are typed into registers, action of the integrated system are read from other registers.

Figure 7-3 Closed Loop Verification

7.6 INTEGRATION TESTING

7.6.1 For large upgrades or new installations there are many different system suppliers that are involved in delivering a portion of the overall control system. In practice, unless otherwise required, each provider will satisfy software test requirements by solely testing their deliverables. Software intensive systems have one or more interface points with other software intensive systems. In an ideal project, there will be well defined interface documentations that describe the protocol details for the relevant communication layers employed as well as the expected sequence of commands and responses. As part of software testing, it is important to verify and validate internal interfaces as well as the external ones. A Systems Integration Test (SIT) is a practical way of achieving such validation and can be scheduled to be carried out outside the critical path in a controlled and benign environment. The party responsible for integration needs to identify the critical interfaces and bring the software teams and relevant hardware into a single area where the systems may be connected and validated to ensure the system-system interactions are properly tested.

7.7 NOTATIONS FOR SYSTEM VERIFICATION BY TESTING

7.7.1 For full details, see DNV Rules for Ships, July 2013 Pt.6 Ch.22 Enhanced System Verification (ESV). ESV is a class notation assigned by DNV where specified on-board systems are analysed by the use of one or more verification methods as described in the DNV ship rules in order to provide objective evidence of acceptable functionality during normal, abnormal and degraded conditions and quality according to stated requirements. Vessels built and tested in compliance with the requirement of these rules may be assigned the class notation for enhanced system verification. Table 7-1 shows the types of system to which this notation can be applied and the scope of the testing within these systems. Note - not all parts of the system as described in the rules are subject to testing.

Table 7-1 ESV Target Systems

Qualifier	Target System	Scope of Verification
DP	Dynamic Positioning Control System	DP control system Typically, not including the reference systems and sensors.
TAM	Thruster Assisted Mooring System	Thruster assisted control system. Typically, not including the reference systems and sensors.
PMS	Power Management System	Remote control and monitoring of power generation. Remote control and monitoring of power distribution. Load dependent start/stop. Load sharing Blackout prevention and load reduction Blackout recovery.
SPT	Steering, Propulsion and Thruster System	Steering control and monitoring system Propulsion control and monitoring system. Thruster control and monitoring system.
ICS	Integrated Control and Monitoring System	Control and monitoring of vessel main function. Control and monitoring of valves and pumps. Main alarm system.
DRILL	Drilling Control System	Zone management/anti-collision system Rotation system Hoisting system Hydraulic power unit Equipment handling Vertical pipe handler Horizontal pipe handler Cranes and winches Heave compensation and tensioning system Marine riser tensioners Active compensation systems Passive compensation systems Drilling fluid circulation and cementing Mud circulation system Cementing system.
CRANE	Crane Control System	Control and monitoring Safety functions.
BOP	Blow Out Prevention System	Operator panels Choke and kill. Diverter Emergency disconnect system. Acoustic control system Hydraulic power unit Stack control.
SAFETY	Process Safety System	Emergency Shutdown (ESD) Fire and Gas (F&G).

7.7.2 Hardware in the Loop testing is one means of enhanced system verification. There are now two different HIL notations (see Table 7-2) one for independent systems verification where the entire test package is provided by an independent HIL supplier. The second is for dependent HIL where the test simulator is provided by the organisation that produces the control system being tested. There are advantages and disadvantages to both approaches and some of the arguments made by their proponents are given below.

7.7.3 Dependent HIL may offer cost savings in relation to the absence of the requirement for an in-house simulator. For those vessel owners that are significantly cost constrained this could make the difference between carrying out some form of testing or none at all. It seems likely that there would always be some benefit from software testing particularly in the case of those systems with a strong vessel specific element and safety critical nature.

Note: There may be drivers other than cost savings that dictate the choice of dependant HIL. The benefits of HIL regardless of whether it is dependent or independent is being recognised and realized by industry.

7.7.4 For independent HIL, there is a cost associated with creating a new simulator but this can be offset when it can be reused for subsequent projects with minor adjustments or configuration changes. For maritime systems like DP and PMS, the work associated with configuring the simulator is not excessive since it overlaps with the system analysis that has to be performed in order to produce an effective test program.

7.7.5 When considering the degree of independence required, IEEE 1012 Annex C states that rigorous technical independence, which includes independent test tools, should be applied for systems requiring a high integrity level. Therefore, the independence of the simulator developer and configurator should be driven by the criticality of the system.

7.7.6 The rigour required in the simulator should be driven by the criticality of the system, rather than the independence and knowledge of the simulator verifier or the independence of the simulator owner. The independence of the simulator owner is less important than the independence of the developer and configurator of the simulator.

7.7.7 The quality of any simulator is of course dependent on the knowledge of the developers whether they are associated with the simulator owner or an independent organisation. The same argument is true for a simulator verifier. If the verifier is not competent, the rigour of the simulator may not be satisfactory.

7.7.8 When a dependent simulator is used, the independent verifier must verify two black boxes: i.e. the control system and the simulator. Without detailed knowledge of the design, implementation, and configuration of the simulator, the independent verifier may have an additional burden related to identifying configuration errors or design weaknesses common to the control system and the simulator.

7.7.9 HIL simulators are to a large degree constantly validated by the control systems they test and vice versa. This is comparable to the relationship between production code and test code (unit tests) and double-entry book-keeping in a financial accounting system. Independent HIL simulators are further validated since they are tested against control systems from different vendors. If the control system and simulator are developed and configured by the same vendor, the opportunity for such validation may be lost. A vendor can save time by automatically configuring the simulator based on the control system configuration, but this may cause masking of erroneous configurations. The error might not be discovered since it is the same in both the controller and simulator.

7.7.10 In general terms, the basis of choice if a system will be tested to HIL-IS or HIL-DS standards should be objective and based on sound technical reasons. For example, the organization that holds the most relevant mathematical model (and the HIL test harness to activate it) could form the basis for choosing the organization that is to provide the HIL service. If an Independent HIL supplier has a suitable mathematical model then they would be the best technical solution. If a drilling system supplier has an internal 3D environment with collision detection that is well developed to test their software, then they should supply the HIL service in this case.

Table 7-2 HIL Notations

Qualifier	Description
HIL-IS	HIL test package provided by independent HIL supplier.
HIL-DS	HIL test program package and HIL test package report provided by independent HIL supplier. HIL test simulator package provided by the organization delivering the HIL target system.

7.7.11 Reference should be made to IEEE 1012 Annex C for further guidance on the degree of independence but the following may be useful when reaching a conclusion.

- Carefully consider the guidance in industry standards on software testing particularly in relation to safety critical activities.
- Carefully evaluate the value of any verification process not just the cost.
- Consider competence as well as independence.
- Given the safety critical nature of some industrial missions, some level of software verification is essential. The inability to reach alignment on dependence or independence should not preclude such verification.

8 SUGGESTED METHODOLOGY FOR IMPROVED SOFTWARE ASSURANCE

8.1 TESTING METHODOLOGY

8.1.1 Verification or testing of software can be divided into two sub categories - static testing and dynamic testing. Briefly described, static testing comprises reviews and inspections of documents and code, while dynamic testing is testing of running code - the program is executed while tested.

8.1.2 Dynamic testing can again be divided into two main groups - Black box and White box testing. As the name indicates, Black box testing means that the internals of the system or software are not known to the tester, while White box testing indicates that the source code is known to the tester. Often Black box testing is also known as functional testing, or specification based testing, and White box testing is known as structural testing. In white box testing the internal perspective of the system as well as programming skills are used to design test cases. The tester choses inputs to exercise paths through the code and determines the appropriate outputs. While white box testing can be applied at the unit, integration and system levels of the software testing process, it is usually done at the unit level. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements.

8.1.3 All methods are necessary within a good software development life cycle. Different types of errors are typically detected by the different test methods.

8.1.4 Software testing requires a clear structured approach which defines how testing should be planned, conducted, and documented.

8.2 TOOLS

8.2.1 Testing different types of software and test environment

8.2.1.1 The difference between control system software and a desktop application - like Microsoft Word®, is that while the only interaction MS Word has with the surroundings is the user of the program, a control system also interacts with its associated plant.

8.2.1.2 A realistic user / test environment for MS Word, is a PC with MS Word installed. In addition, you would most probably connect a printer device to test that a document can be printed on paper.

Note: 8.2.1.1 and 8.2.1.2 describe an example.

8.2.1.3 A realistic test environment for a control system will not only be a computer with the control system software application installed; it must also contain a representation of the plant which is controlled by the control system. This type of software testing is called simulator based testing. In the marine and offshore sector this kind of testing is known as Hardware-in-the-loop (HIL) testing.

8.2.2 Test Location

8.2.2.1 The real plant often imposes limitations in terms of the test scope, since testing with the plant in the loop may be impractical. Testing outside plant limitations may also lead to sub-optimal test conditions both for the plant itself, and the people performing and witnessing the testing.

8.2.2.2 In order to overcome limitations of using a real plant, most of the software functionality and failure handling capabilities can be tested against a plant simulator. This allows for enhancing the quality of testing by increasing the test scope. However, some parts of the test scope must be covered by on-board testing.

8.2.2.3 A plant simulator also enables testing without access to the vessel, which usually reduces the cost of such testing.

8.2.3 Software life cycle considerations

8.2.3.1 As indicated in Figure 8-1, the other main category of software testing is static testing. Typically, this includes reviewing requirement, design specifications and code. This is also an important test method as it may reveal different types of defects compared to dynamic testing.

8.2.3.2 Good quality software requires a systematic and mature development process. To be able to maintain the quality during the life cycle of the software (and the vessel), good maintenance processes are also a necessity. Within all life cycle models, verification is essential and should be applied during the entire software life cycle.

8.2.3.3 Verification will be performed by stakeholders with different viewpoints usually influenced by their roles. Some verification methods are better suited to specific life cycle phases than others revealing different kind of defects.

8.2.4 Standards for software development and verification of integrated systems

8.2.4.1 Software quality depends on good and well-defined development processes. Within the maritime industry there have emerged standards and guidelines that place requirements on the software development and verification processes.

8.2.4.2 These new process oriented standards and guidelines by DNV and ABS are complementary to the equipment specific class rules. The standards may be divided into two main groups - those focusing on the whole software life cycle, and those focusing on the verification of the software.

- SW verification standards: DNV-ESV/SfC, and ABS-SV.
- SW life cycle standards: DNV-ISDS, and ABS-ISQM.

8.2.4.3 The software verification standards, DNV-ESV/SfC, and ABS-SV, describes HIL-testing (simulator based dynamic testing - black box testing) of the control system software.

8.2.4.4 The other two software standards issued by DNV and ABS, DNV-ISDS, and ABS-ISQM may be categorized as software life cycle standards, see Figure 8-1. They will have a broader focal point than the two previous; as they put requirements towards all software development activities in all phases of the software development process, including the verification process.

8.2.4.5 Both DNV-ESV/SfC, and ABS-SV may be used within DNV-ISDS and ABS-ISQM as a framework for HIL testing.

8.2.4.6 DNV-ESV/SfC, and ABS-SV focuses on safety. DNV-ISDS and ABS-ISQM focus on other software quality aspects.

8.2.4.7 The system owners are an integral part of the System Development Life Cycle (SDLC) and Validation & Verification (V&V) portions of software engineering as the end user. The system owner must have a rigorous and documented software change management (MOC) process in place.

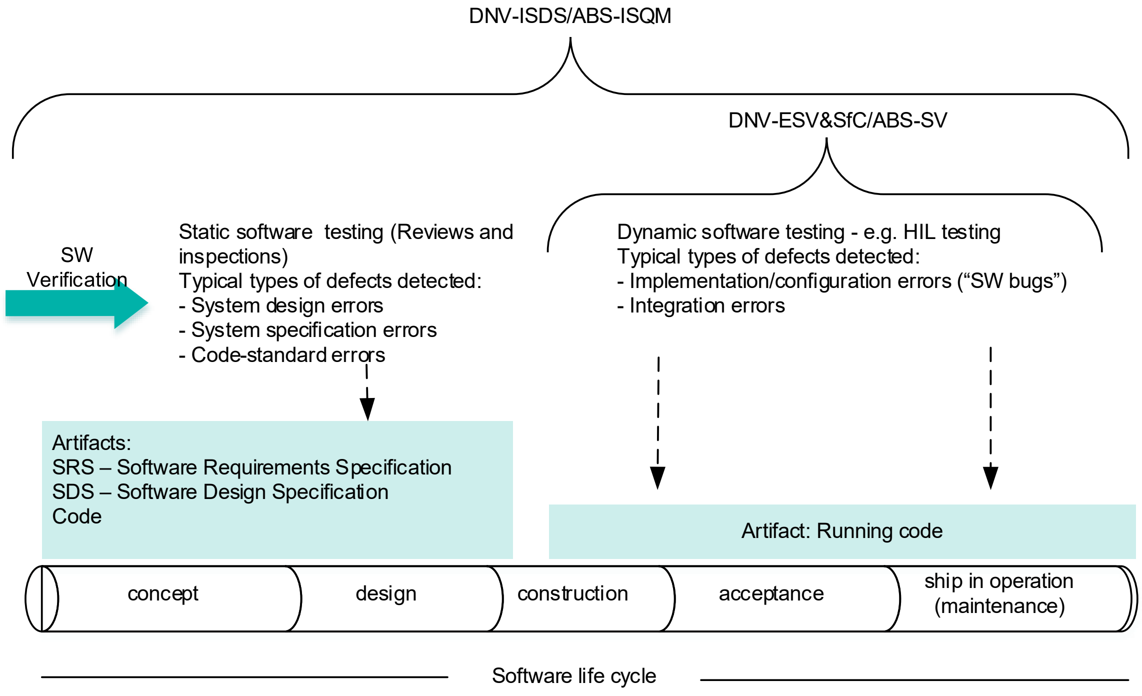

DNV-ISDS/ABS-ISQM
DNV-ESV&SfC/ABS-SV
SW Verification
Static software testing (Reviews and inspections)
Typical types of defects detected:
- System design errors
- System specification errors
- Code-standard errors
Dynamic software testing - e.g. HIL testing
Typical types of defects detected:
- Implementation/configuration errors ("SW bugs")
- Integration errors
Artifacts:
SRS – Software Requirements Specification
SDS – Software Design Specification
Code
Artifact: Running code
concept
design
construction
acceptance
ship in operation (maintenance)
Software life cycle

Figure 8-1 Software Life Cycle and Verification Standards

8.2.5 Independence in the verification and validation effort

8.2.5.1 Software testing should be as objective as possible. The usual perception is that the closer the test organisation is to the developer, the more difficult it is to be objective. Another usual perception is that the level of independence and therefore the objectivity increases with the 'distance' between the developer and the test organisation.

8.2.5.2 These perceptions are usually used to support the requirement for independence in software testing of safety critical systems and thus important to separate these two activities, not only regarding work processes, but also in mind-set.

8.2.5.3 Independence facilitates a fresh view of the system and therefore ensures objectivity. The degree of objectivity should increase with increased system criticality. These overall principles are described in numerous standards, both generic and in the DNV and ABS standards. The test organization must however demonstrate a competency in the field they are testing. Independence does not however require it to be from a separate commercial organization as identified in 0.

8.2.5.4 The IEEE1012 Standard for System and Software Verification and Validation is recognized as the de-facto verification standard by several US organizations such as NASA and for critical weapon systems for the United States armed forces.

8.2.5.5 IEEE1012 is very specific regarding different aspects of independence in the verification effort. The independence is described in two dimensions – type and form:

8.2.5.6 Independence type is defined by three parameters in the standard:

1. Technical independence.
2. Managerial independence.
3. Financial independence.

8.2.5.7 Five forms of independence are defined:

1. Classical.
2. Modified.
3. Integrated.
4. Internal.
5. Embedded.

8.2.5.8 Classical independence is when the verification organization is an external organization (different company), and embodies all three types of independence (technical, managerial, financial). This is the level of independence expected when testing safety critical systems.

8.2.5.9 If the verification effort is organized in the developers' group (although not directly involved in the development) it is called Internal. Non-safety critical systems may be verified according to this level of independence.

8.2.5.10 Table 8-1 summarizes the forms and levels of independence. Notice, only "Classical" independence ensures rigorous independence with respect to all three forms of independence.

Table 8-1 Form and Level of Independence According to IEEE1012

Form	Technical	Managerial	Financial
Classical	I	I	I
Modified	I	i	I
Integrated	i	I	I
Internal	i	i	i
Embedded	0	0	0
Note: I = rigorous independence, i = conditional independence, 0 = minimal independence			

8.3 ROLES WITHIN THE VERIFICATION EFFORT

8.3.1 Different standards call for slightly different structures within the verification process and, different roles are held by different organizations or stakeholders. The following discussion considers one particular arrangement but others are equally valid. Four of these roles are relevant for the purpose of providing independence in the verification effort:

- Validation team.
- End user.
- System integrator.
- Vendor (developer).

8.3.2 It is important to distinguish between these roles because they possess different responsibilities within the verification effort and require a different kind of knowledge and focus. Although significant focus is placed on the independence of the 'test plan' some flexibility is allowed in relation to the issue of the independence of 'test objects' such as simulators. IEEE 1012 states:

"For system tools, technical independence means that the Independent Validation & Verification effort uses or develops its own set of test and analysis tools separate from the developer's tools. Sharing of tools is allowable for computer support environments (e.g. compilers, assemblers, and utilities) or for system simulations where an independent version would be too costly."

8.3.3 This can be interpreted to mean that technical independence requires independent test tools. This requirement should be discussed with all stakeholders and agreement reached as stipulating independent test tools has a potential cost impact.

8.3.4 The **verification organization** is responsible for running the independent test process:

1. Planning.
2. System analysis.
3. Test design.
4. Test case and test procedure implementation.
5. Test execution.
6. Test result analysis and follow up.

8.3.5 Extensive knowledge about software test techniques and system analysis in addition to general system knowledge are required for this role. It is the level of independence of the validation team that determines the verification effort independence, not the independence of the test witness or the system integrator.

8.3.6 The **Test Witness** will monitor the fidelity of test execution to the specified test procedures and witness the recording of test results. It is important to note that the performance of the Test Witness task largely relies on the transparency of the test processes set in place by the Verification Organization. The transparency of the vendor's verification process may become degraded due to a conflict of interest, while the independent Verification Organization will not have such conflicts of interest.

8.3.7 It is also important to notice that witnessing a test execution will not qualify the witness to take the role of the validation team. This role requires more hands on effort in the testing process including actual usage of the test tools such as e.g. the HIL simulator.

8.3.8 The **system integrator** will be responsible for the integration of the total system and therefore may be regarded as part of the development process. This role is intended to be taken by the shipyard for new build and by the owner for vessels in operation.

8.4 VERIFICATION & VALIDATION STRATEGY

8.4.1 Depending on the product and the responsibility of the organization to which the system belongs, a test plan needs to be designed. A test plan may include one or more of the following:

- Design Verification or Compliance test - to be performed during the development or approval stages of the product, typically on a small sample of units.
- Manufacturing or Production test - to be performed during preparation or assembly of the product in an on-going manner for purposes of performance verification and quality control.
- Acceptance or Commissioning test - to be performed at the time of delivery or installation of the product.
- Service and Repair test - to be performed as required over the service life of the product.
- Regression test - to be performed on an existing operational product, to verify that existing functionality is not affected when other aspects of the environment are changed (e.g., upgrading the platform on which an existing application runs).

8.5 OPPORTUNITIES FOR IMPROVEMENT

8.5.1 While third-party testing, verification, and classification of structures and mechanical systems are well-established in the maritime and offshore industries, the increasing use of computer control systems has not yet been subject to the same degree of third-party testing and verification activities. The result is that a large portion of the automation systems on today’s vessels are commissioned with less software testing than could be carried out. This is a paradox considering that the automation systems often contain failure handling functions that are critical for safety but may be difficult or sub-optimal to test on board the real vessel. Therefore, these systems have the opportunity to be more thoroughly tested prior to being activated during an emergency situation.

8.5.2 For DP vessels, the internal software of the position reference systems e.g. GPS, HPR and sensors like Gyro and MRU are not tested when HIL testing is employed for DP control systems. The HIL testing only verifies and tests how well the DP controllers are able to handle HW and SW failures in the position reference systems. During these tests if the result is not as expected, it will be flagged as a finding from the test and vendor, owner and yard have to agree upon what to do about it. This is where a comprehensive SIT will benefit the stakeholders.

8.5.3 Consideration should be given to extending the software testing process to ensure the correct functionally of functions such as signal quality indicators within reference systems particularly where the DP control system makes use of such indicators to determine the reliability of a position reference and weight it against others. This can supplement the DP control systems ability to determine whether a reference system is trustworthy.

8.5.4 Rules and guidance for the development of safety systems exist but testing of software for safety systems such as Emergency Shut Down (ESD) systems and Fire and Gas detection systems does not appear to be part of the current focus on software testing. Consideration should be given to determining the extent to which including safety systems in the software verification process described in this TECHOP would improve DP safety and reliability. The testing of these systems needs to extend beyond validating the C&E matrix. Such systems could benefit from more rigorous tests (example HIL, additional V&V testing).

8.5.5 The conclusion is that software testing should be developed utilising a strict review and testing process, adopting existing standards but also employing verification (independent & dependent). The degree of verification required should be directly related to the criticality of the system under test and the consequences of failure and its effects on life and the environment.

8.5.6 The choice of independence or dependence of verification should be agreed by all stakeholders on the basis of objectivity and sound technical grounds.

9 FURTHER READING

1. ‘Control System Maker Involvement in HIL Testing’, Oyvind Smogeli, Marine Cybernetics, 2011-10-31.
2. ‘DP System HIL vs. DP System FMEA’, Odd Ivar Haugen, Marine Cybernetics, 2012-02-06.
3. ‘HIL Test Interfacing’, Oyvind Smogeli, Marine Cybernetics, 2011-10-31.
4. ‘HIL Test Process Class Rules’, Odd Ivar Haugen, Marine Cybernetics, 2012-12-12.
5. ‘Introduction to Third Party HIL Testing’, Oyvind Smogeli, Marine Cybernetics, 2010-10-26.
6. ‘Why Chose an Independent Verification Organisation’, Odd Ivar Haugen, Marine Cybernetics, 2012-12-13.
7. ‘Control Systems Acceptance: Why Prevention is Cheaper than Test’, Athens Group.
8. ‘Cross Checking the Checklist’, Bill O'Grady, Athens Group, September 2011.
9. ‘How to Build a Stronger Contract to Improve Vendor Accountability and Reduce Control Systems Software Costs and Risks’ - Athens Group, 2010.
10. ‘Fit for Purpose Control Systems Emulation - The Case for State Based Emulation and a Lifecycle Approach to Reducing risks and Costs on Offshore Assets’- Athens Group, 2011.
11. ‘Getting Software Risk Mitigation Right from the Start, Nestor Fesas and Don Shafer’, SPE 143797, 2011.

10 MISC

Stakeholders	Impacted	Remarks
MTS DP Committee	✓	To track and incorporate in next rev of MTS DP Operations Guidance Document Part 2 Appendix 1. Communicate to DNV, USCG, Upload in MTS website.
USCG	✓	MTS to communicate- FR notice impacted when Rev is available.
DNV	X	MTS to Communicate- DNV RP E 306 impacted.
Equipment Vendor Community	✓	MTS to engage with protection suppliers
Consultant Community	✓	MTS members to cascade/ promulgate.
Training Institutions	X	MTS members to cascade/ promulgate.
Vessel Owners/Operators	✓	Establish effective means to disseminate information to Vessel Management and Vessel Operational Teams.
Vessel Management/Operational Teams	✓	Establish effective means to disseminate information to Vessel Operational Teams.