DYNAMIC POSITIONING CONFERENCE
October 13-14, 2015

SOFTWARE DESIGN & CONTROL

Software management for a critical, real-time product

By Ivar-André F. Ihle

Rolls-Royce Marine AS

Abstract

Software is an essential component of the Dynamic Positioning product. The software management process plays an important part during all parts of the product lifecycle process. When the software program is critical for the vessel safety and placed on a vessel that can operate anywhere in the world, the software management process must account for more than the usual computer application. Some classification society notations (DNV-GL's Integrated Software Dependent Systems and American Bureau of Shipping's Integrated Software Quality Management) address integrated software systems. The Marine Technical Society Dynamic Positioning Committee addresses software management in the TECHOP paper on Software testing.

This paper aims to encourage a discussion in the community on this topic by showing how practical software management challenges are managed for the Rolls-Royce Icon Dynamic Positioning System.

The software management process encompass software requirement analysis, system design, software coding standards and methods, software development, software environment, software testing, system deployment, bug handling and –fixing, security and privacy, and quality assurance. In the paper we aim to share our practices and experiences with software management in general. We will discuss the special challenges that arise with the installation and maintenance of a safety critical control system onboard a floating vessel. The different steps of software testing, bug handling and fixing in different stages of the product lifecycle will receive particular attention.

Cases and experiences from the Icon Dynamic Positioning system will be used to exemplify and highlight what we believe to be key elements of software management. We hope that others find it interesting and share their experiences in the future to further encourage the discussion of business' best-practice.

Introduction

The software is becoming an increasingly important component of the modern vessel. Algorithms can now be used to control vessel movements and optimize performance. Sensor inputs can be monitored and long term trends can be found that reveal more information both to the operator onboard but also to the equipment manufacturer, the shore office and external third parties.

Vessel control systems have been around since the first autopilot, but the control algorithms were then implemented with specialized hardware components. Although it is easier to find logical faults by visual inspection, the possible extensions are limited. Thanks to the development of flexible hardware and software development the computer consists of two almost independent systems. Computer systems today can be extended and modified to suit almost every need. However, the complexity of a software system may increase exponentially with new features and additions.

A Dynamic Positioning product consists of software and hardware components. The software management process plays an important part during all parts of the product lifecycle process. When the product is critical for the vessel safety and placed on a vessel that can operate anywhere in the world, the software management process must account for more than the usual computer application.

In this paper we aim to share our practices and experiences with software management of the Dynamic Positioning System. We will discuss about tools for software management, development, bug reporting, and configuration management. We will also show how a rather old (from 1984) document preparation system and document markup language can be used to automatically create a complete and customized product documentation and user manual.

We will discuss the special challenges that arise with the installation and maintenance of a safety critical control system onboard a floating vessel. The different steps of software testing, bug handling and fixing in different stages of the product lifecycle will receive particular attention.

Cases and experiences from the Dynamic Positioning system will be used to exemplify and highlight what we believe to be key elements of software management. We hope that others find it interesting and the industry is encouraged to share their experiences to promote best-practice.

Brief introduction to software management

This section briefly introduces the overall context in which the continuous software management takes place. Following the industrial standard IEEE 12207-2008 Second Edition Systems and Software Engineering – Software lifecycle processes [4] the software development consists of four phases:

1. Concept,
2. Requirements & Design,
3. Construction, and
4. Verification, validation & transition.

Wikipedia defines the core activities of the software development process as

- Requirements,
- Design,
- Construction,
- Testing,
- Debugging,
- Deployment, and
- Maintenance.

There exist a number of standards, guides, and recommended practices related so software management. Some of the more relevant documents for the maritime industry are:

The DNV-GL Offshore Standard Integrated Software Dependent Systems (ISDS) [2] puts requirements on the work execution but no specific product requirements. Different roles are identified and their responsibility for project delivery is described: owner, system integrator, supplier and independent verifier.

The American Bureau of Shipping's Guide for integrated software quality management (ISQM) [1] presents the procedures and criteria used by ABS to review all parts of the software development life cycle for individual and integrated computer based control systems.

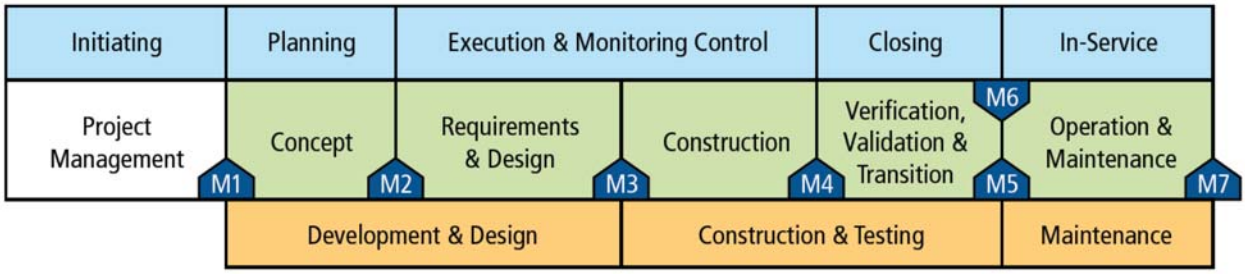

Figure 1 ABS ISQM process

The most relevant document for dynamic positioning systems is the Marine Technology Society's TECHOP on Software testing [10]. It highlights the software issues for DP redundancy and fault tolerance not typically covered by FMEAs.

There are many different software development methodologies. One of the earliest described is the waterfall method where the development moves through the process steps in a linear fashion. To reflect the relationships between phases in the development life cycle the V model has a link from the validation to the concept phase. This shows the dynamics when testing shows that the initial design does not cover the intended scope. The agile method is considered a pragmatic approach to show the process and the process interdependencies from the software developers' point of view.

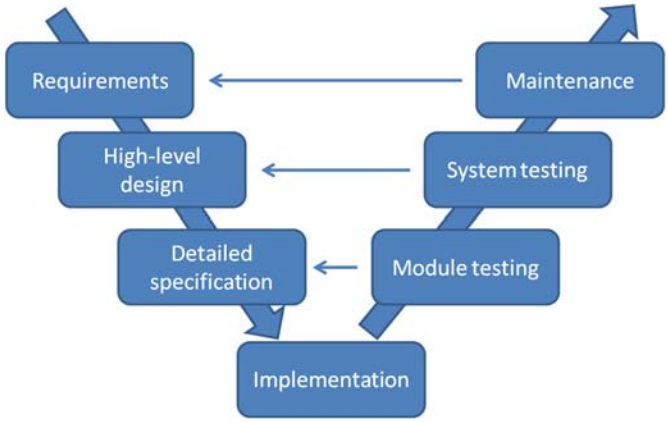

Figure 2 The V-method

A development project in Rolls-Royce Marine will be executed according to the Rolls-Royce Good Practice Framework process (GPF) [11]. The GPF is a framework for good project management, product

development and procurement using experience from the aerospace sector and best practice from industry.

The six stages of the process cover all the activities associated with a project's life-cycle, i.e. from generation of an initial requirement for a product or service, through to its disposal. The process:

- Has gates not only at the end of each stage but intermediate gates are also required to provide adequate control.
- Incorporates practical considerations that must be taken into account in each stage, phase, review and gate.
- Provides a disciplined framework which establishes at the gates that all necessary activities have been completed at the right time. If all activities have not been completed then management must recognize the potential risks in proceeding through the gates.

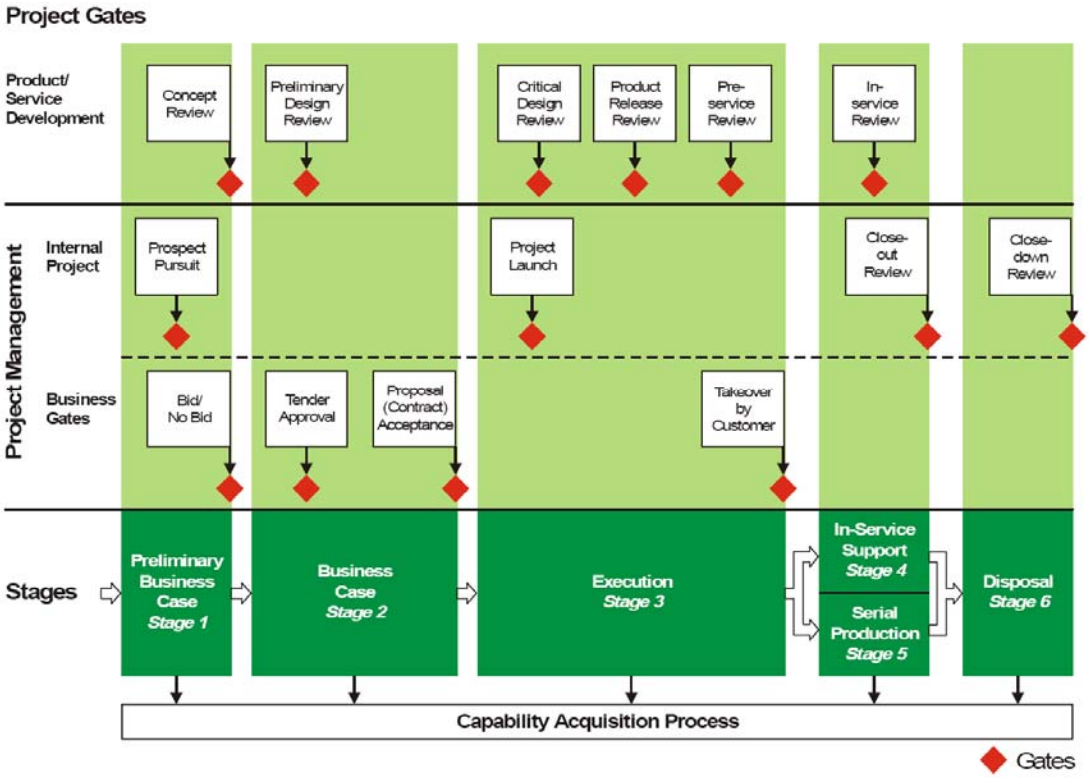

Figure 3 Rolls-Royce Good Practice Framework – Stages and gates for project execution

This paper will in particular consider the continuous software development process that occurs in an existing product – basically be part of project Stage 3 – Execution.

There is already an established framework for the DP product and the development team faces a problem known to many software development projects: new functionality is continually added while previous versions must be supported. In the sections below, we will outline the software management process, share some of our practices, and describe the tools we use. Parts of this paper are adopted from [15]—[18].

Important Lessons

The most important lessons are outlined below. The details can be found in the following chapters.

- Focus on the final product throughout the software development process.
- Incorporate robustness, error handling, fault detection and tolerance at an early stage in the design process. Operator and workplace considerations must be included.
- The code base should be as general as possible.
- Maintain transparency and communication during the development phase.
- Organize software structure for reuse and limit dependencies.
- Write readable and testable code.
- Test extensively at the earliest stages
- Verify system and function input.
- Keep track of execution time at all times.
- Track issues from requirement stage to final code

For more lessons on real-time systems, see [19].

The DP software

The DP system is an industrial control system which operates in real time. A real time system differs from a software application running on a smart phone or desktop because operations depend not only on logical correctness but that all events must be processed within timing constraints.

Each software delivery consists of all necessary executable files and configuration files to run the relevant positioning product on the relevant hardware. The set of configuration files has delivery-specific attributes as described below. The DP software consists of three main parts:

- a code base for the controller software,
- the graphical user interface (GUI) software, and
- a configurator.

The controller software is in charge of the real-time computation for maintaining the vessel position. It must interface other equipment onboard, process signals from all sensors and position reference systems to determine vessel position, and control available thrusters and propellers to maintain the desired position.

The graphical user interface (GUI) on the operator station is a separate piece of software. It is responsible for operator interactions, displaying vessel position and orientation and present results from the controller software. The GUI must be integrated into the overall operator workspace to ensure the best possible workspace. All graphical components follow the Rolls-Royce Marine Common Look and Feel Style Guide [14] which has defined guidelines for the graphical user interface, principles for interaction and usability, and a library of common controls, icons and symbols.

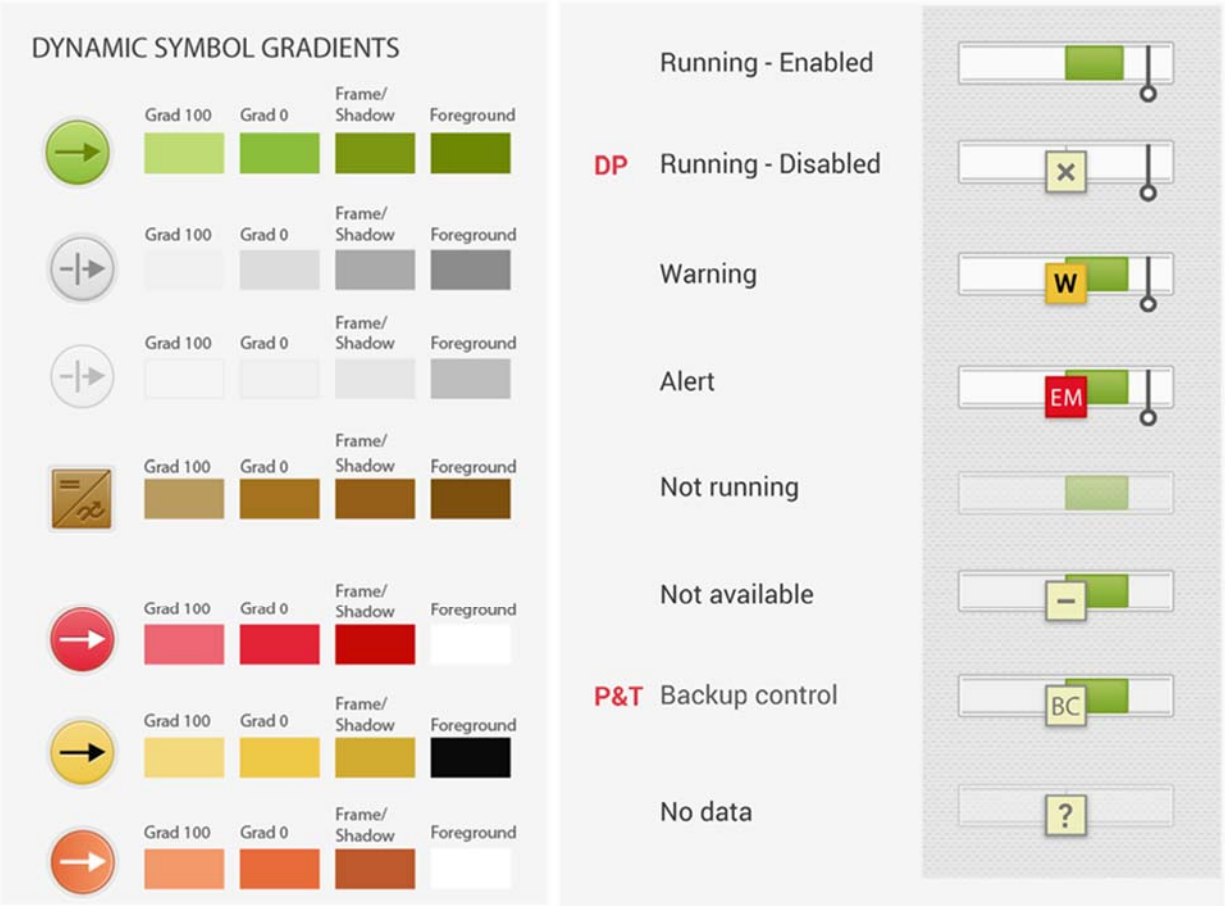

Figure 4 Common look and feel color gradients and thruster/rudder states.

The configurator produces a set of configuration files to be used by the controller, GUI software and for producing the user manual. It contains all information relevant for the delivery. The configurator requires input regarding:

- Control system configuration. Level of redundancy (e.g. DP class 1, 2, or 3), class society classification, number and type of operator stations, network configuration, and number and type of IO-units that are configured according to devices of the vessel
- Sensors and position reference systems configuration
- Thruster and power systems configuration, e.g., thrust curves
- Vessel data: length and breadth and hydrodynamic values.
- System functions and features according to scope of supply.

For all software deliveries which are not direct repeats a configuration test shall be performed. This test shall ensure that the configured data is correctly entered into the configuration system and that the particular configuration is supported by the selected software version.

The code base is generic with no hardcoded values and will be identical for all deliveries (with the same version number). Parameters and specific values are separated from the code and can be edited before runtime.

Figure 5: The DP system consist of the DP code base and the delivery-specific configuration files.

Establish requirements

The requirements for the existing DP system are determined by class rules, business requirements and customer inputs. When developing new functionality the problem is analyzed internally to identify the actual need. Field experience and DP operator interviews identify the key challenges for the problem. A set of requirements are then formulated to solve the need while addressing the main challenges. The requirements are aligned with class and business and documented.

A product may fulfill the technical requirements but lack the usability that makes it useful. We may also revisit the requirements at a later stage if the results are not fulfilling the actual intent. A target audience of DP operators with different background can be used to review concept, test solutions before launch, end ensure that user and developer have similar understanding.

Figure 6 Both coffee pots are made according to specifications. However, but one lacks the usability.

Our software development tools

The development team has tools that help us improve our work and the software management process. These tools include programs for writing code, tracking requirements, version management and testing.

Independent considerations are made to determine code language and integrated development environment (IDE) for the controller and the graphical user interface. The need and availability of existing libraries and tools is examined for each part of the system. It is important to understand and acknowledge the possibilities, strength and vulnerabilities of different code languages. Some tools to consider in a development environment, both from a technical and a developer point-of-view are cross-compilation functionality, debugging options, code analysis, support for code refactoring and auto-completion tools/extensions.

Controller code is written in C++ and software development may occur on Windows or Linux. To run on a controller the code must be cross-compiled to fit the particular machine. Being able to debug directly on the controller platform may resolve technical issues at an early stage. The graphical user interface for the DP system is written in Java which is cross-platform compatible.

The development team uses a software issue tracking program extensively. Issues can be requirements, new features, improvements, bugs, tasks. To track development progress requirements are mapped into the program as a single issue or as a set. Issues belong to a software component and have fields for time spent, comment fields, relations with other issues. By incorporating links to the software repository the complete development history is maintained at all times. A software release can then be composed of a set of issues. Furthermore, issues can be prioritized and be of different types to efficiently organize software development.

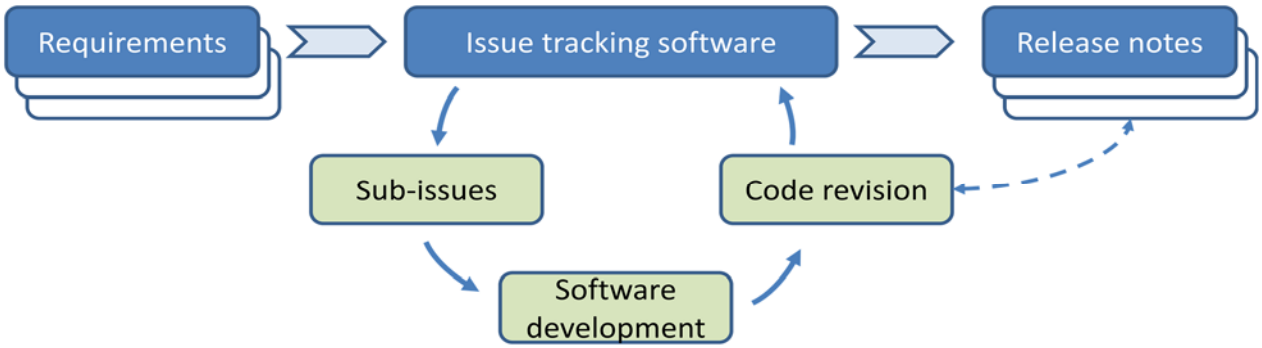

Figure 7 Relationship between documented requirements, issue tracking, code versions and release notes.

To coordinate team efforts a program for software project management is employed. It contains functionality for creating and monitoring dedicated team efforts where a set of issues are to be fixed within a fixed time frame. Team progress is available in different reports and visualization methods. Team boards makes it easy to immediately see the current project status by displaying which issues are on the to-do list, which are in progress and which issues are completed.

A software revision control program is used to keep track of software versions and maintain current and historical versions of source code and documentation. More information on revision control is found below.

A build server is our tool for continuous integration and regularly builds the project on the trunk and reports of any compilation errors. This is useful in a multi-user project as an additional layer of insurance to make sure submitted changes doesn't cause build errors on the trunk.

Coding

Coding is a core part of the software development and it is important to have proper guidelines. The code base is written and maintained by different persons and each one may have their own favorite code writing techniques and formats. To improve code understanding, debugging, addition of new functionality, and maintain readability our team has a standard set of coding guidelines. The C++ code shall comply with the MISRA C++ code standard. MISRA stands for Motor Industry Software Reliability Association and facilitates code safety, portability and reliability of embedded systems [9].

Function, type, variable and file names are descriptive. Variables are nouns while functions are verbs or questions. To accommodate longer names CamelCase (or MixedCase) separates words. We may also separate between states (may be updated at every time step) and parameters by using an extended prefix: mx or mp. Similar functions are named such that the functionality is given in the last word of the name. To avoid filling a class with too many methods, each class has a single responsibility. New functionality may be grouped with similar functions or placed in a new location.

Numerical robustness is pursued at all times. We assume that all floating point numbers are C++ doubles. A set of constant variables are defined to avoid ill-conditioned values and exceptions. For example, for floating point division the divisor size is checked to avoid floating point exceptions, division by zero and overflow. For operations that include integers and floating point numbers the code must carefully cast the correct type.

To ensure code robustness all system and method input should be validated, (correct format and value range), before used further. Before a pointer (reference in C++ code) is used in the code a zero pointer check is used to confirm its validity. Input validation is also important from a software security point of view by preventing vulnerabilities and protecting against potential security holes.

When using loops extra care must be taken to ensure that final conditions will be fulfilled to avoid infinite loops.

Variables and methods that are no longer used should be removed and not just commented out. When using a software repository it can always be restored later.

The units for physical quantities follow the SI-system and are the same throughout the code. To accommodate the DP operator preferences, the operator can select units presented and the conversion takes place in the user interface.

Software development methods

Our team sits in two geographic locations. Distributed development works well when you have the necessary tools for sharing project information. Face-to-face meetings are important in the project start-up-phase where several people can discuss, draw and interact in different ways. Regular project meetings are held via web-conference, 1-on-1 discussions are available via phone or internet chat (added benefit: rapidly show code examples and illustrations which can be retrieved later). Tracking of software issues and project development should be available for all participants.

We have experiences from test-driven development of new functionality. Test driven development is a software development process with a very short development cycle. Automated tests are written in parallel with the code. A valuable lesson has been the introduction of unit tests where all methods have a predefined test that can be automatically executed after new code changes have been compiled. This helps ensure that the software behaves as expected when code is modified or extended by instantly alerting the

programmer about errors. It is also a useful tool for organize and write better code as it pushes the programmer to write functions that can be tested as stand-alone functions or in a hierarchy.

Programming in pairs can be highly efficient for starting development of new functionality. Many decisions have to be made and discussions around code structure, algorithms etc are taken on the fly instead of leaving the desk. This also works well for distributed teams where participants share a screen and talk to each other. The time spent coding increases but so do the code quality and, most importantly, the number of errors decrease.

Systems are developed in parallel with a software simulator. The simulator includes relevant models such as environmental, vessel model, thrusters etc. We have therefore managed to test most new software components in an isolated environment before further system integration. A simulator enables operational testing and improved verification of DP system functionality.

User manual and product documentation

The DP documentation is generated in a similar manner as the DP software: a general documentation basis and the set of configuration files produce a completely delivery specific user manual and detailed product documentation. This is technically feasible with LaTeX and tags and code fields within the overall tex-file. This ensures that only relevant parts of the documentation are included in the delivery. The manuals will then be completely vessel specific and can be produced in a very short time.

LaTeX is a typesetting tool from the early 1980s with features popular for scientific and technical documentation: it handles complex math expressions well and can automate most typesetting functions: e.g., numbering and cross-referencing of tables, figures, and chapters. It is interesting that a higher degree of user knowledge (text is not formatted as in Word) may automate the process and simplifies the overall complexity of user manual generation.

Testing

The testing phase shall establish the software quality, performance and reliability. Test criteria are

- Repeatability – tests can be reproduced by others to produce the same result and provoke the same error to find root cause.
- Reliability – tests shall produce stable and consistent results
- Accuracy – only a single item should be tested

From a user point of view the execution speed and ease of use is relevant.

The DP software system is tested on different levels to address separate test criteria. Before delivery a system goes through:

Factory acceptance tests (FAT): All software shall be tested on the hardware as built for a particular installation. This test is done according to a class society approved test program. This test ensures that all software and hardware in the positioning product works as specified.

Sea trial Acceptance Test (SAT): Before the commissioning of a positioning product is finished, a SAT test has to be performed. This test is done according to a class society approved test program witnessed by the class society, the customer and the end user.

Failure Mode Effect Analysis (FMEA)

For dynamic positioning system of equipment class 2 and 3 [5], a FMEA test while in DP operation shall be conducted. This test will among other things test the DP systems handling of failures on all systems critical for the positioning of the vessel.

Extended System Verification (ESV)
For customers wanting a higher level of testing of a positioning product several vendors provide ESV testing [3]. The ESV testing will typically be done in two parts. One part will be done at factory before the FAT. And one part will be done during the sea trial after the SAT.

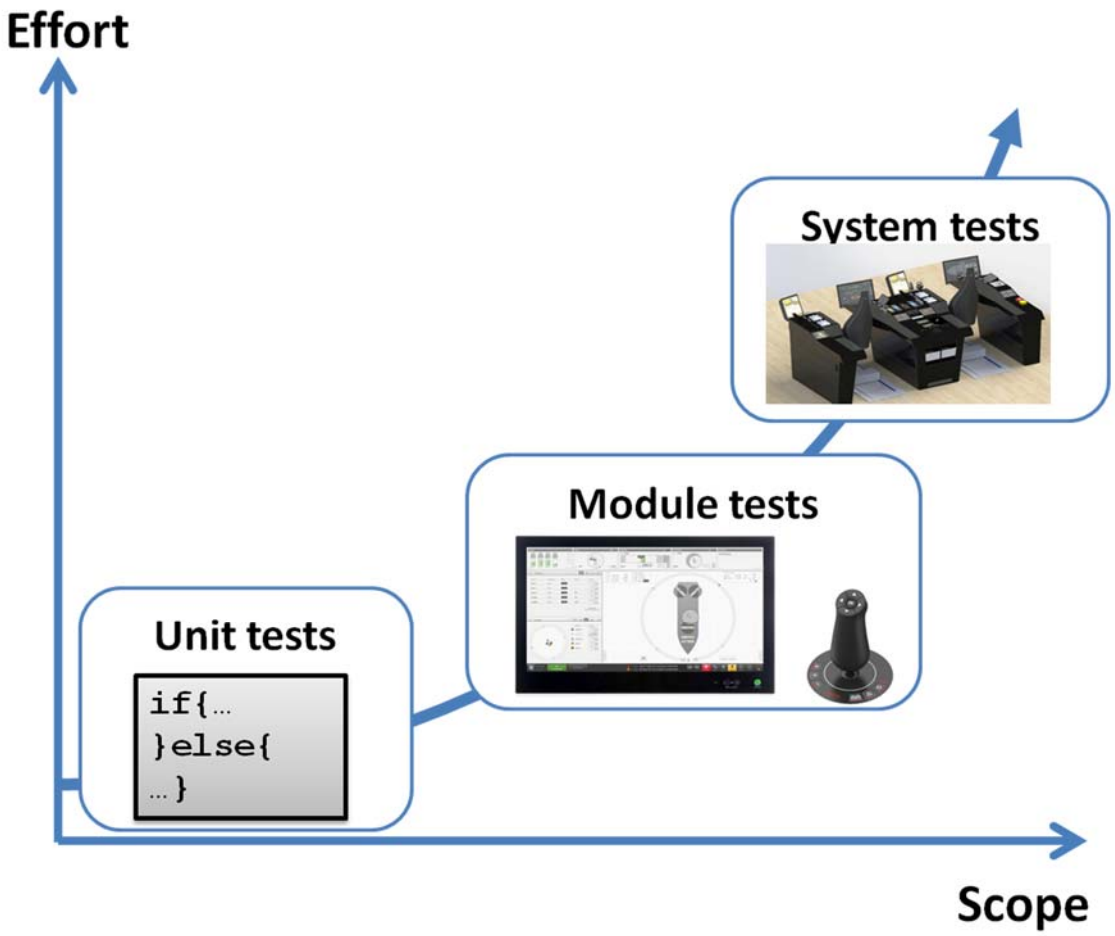

Figure 8: The amount of effort to fix the error increases at the different test levels and the scope is extended

.
The following in this section will consider tests performed during the software development cycle before FAT:

Automated unit tests runs at compilation time. They are written for before the code and can alert the programmer about any errors that arise in the edited method or methods that call the new method. The main benefits of unit tests are that they run independent of user inputs, provide the same scope of tests every time and cover a large part of the code base. The developer can be sure that code work as expected after new changes. The tests are easy to repeat and performed without additional effort. The latter point is important because inconvenient procedures make testing less attractive. A drawback of automated unit tests is the lack of variation in test input and scope. The tests should always test for special cases.

Additionally, the build server periodically builds the latest trunk version and reports to the development team if errors occur.

Module tests target a specific module in the software and can be tested manually or automatically. Functionality is tested locally or on an emulated software version of the controller - so-called Software-In-The-Loop testing. This stage we obtain quantitative measures of model and algorithm dynamics and execution time. We can also address different interactions and use a larger variation of test input. However, it can be time consuming to repeat tests, and some test conditions may be hard to repeat. Module tests will also cover integration between user interfaces and the control system for a specific function.

The next and final step is system-testing with a fully assembled software system with input devices, IO controllers and operator stations. With the complete setup we can verify advanced DP features such as cloning and synchronization of controllers, verify that all subcomponents in the DP system are up-to-date and functions as specified, test interfaces between DP and other onboard systems. Additionally, lab testing ensures that input handling is tested in accurate scenarios.

Additionally, defect tests shall be done to verify that modifications done to resolve a defect works as specified. Test scenarios and detailed test reports are managed and produced using a software testing tool. Test reports for smaller or larger parts of the system can be generated with details on software version and type of tests performed. Reports can be stored and compared to monitor software evolution. The test report quality depends on the user input and the accuracy of written test cases and description of action results.

Software version control

The DP source code is a dynamic product. The development team designs, develops, updates, and deploys software – often simultaneously – and it is essential to organize and control revisions. With a version control system the source code can be efficiently managed. The source code relies in a repository and each programmer can download a working copy of the latest version to a local drive, make changes and submit back to the repository. The submission is marked with a timestamp, user name and other variables to aid the overall software management. It enables the possibilities to revert a file to a previous revision, access different revisions, track changes for a file or a component, branch out a version, tag a version, and merge different versions. A branch is a separate line of development. A tag is a mark of a particular revision to simplify recreation of a certain build. The code base is referred to as the trunk. Including the software issue identification during submission renders the different issues traceable during the development phase.

During the release of new DP software versions the current software base is tagged and a set of release notes is published. The release notes identify and describe the changes in the control application and the graphical user interface software. All associated software issues are mentioned in the release notes.

A key part of our software management strategy is to never modify the software base without incrementing the software version, that is, no software modifications will be done on specific installations

The revision control software is also used to maintain documentation and configuration files in separate repositories.

If a bug is found and exists only on the trunk, the bug is simply resolved and submitted. This is often the case during development of new features. If the bug exists in several branches the bug must be fixed on

each branch. This may be done by merging the branches. Afterwards, new releases are tagged and a new set of release notes are published.

Software deployment, support and maintenance

The DP software is installed onboard before sea trial. The parameters for the DP system are found during sea-trial and saved. Updates are performed onboard by Rolls-Royce service crew. Minor updates are checked with a verification test, while a major software update is verified with a new SAT. The classification society is notified about all changes. Parameters are not part of the software updates and remain the same when the software is updated.

After an error is reported by a vessel crew, DP support begins a search for the root cause. The search depends on the crew's description of the incident and data collected by the onboard action logger. If the incident is identified as a software bug an issue in created in the software issue tracking system.

Logs can be collected by crew onboard and e-mailed to DP support. The logs are replayed and analyzed on a local computer. To determine root cause the team tries to replicate the situation either locally or in the and determine root cause. It is also possible to remotely log on to extract log files, change parameters and remotely monitor operations.

The data logger lies outside the control network for safe data extraction. To improve security of online access the onboard crew must physically establish an internet connection.

Software safety management

Reported issues of potential safety concerns are handled according to the Rolls-Royce Positioning Safety Management Plan and based on company requirements for management of safety, see references [12] [13]. A reported safety concern is internally investigated before determining if it shall be accepted as an unsafe condition. Safety concerns are managed through a Safety Alert Report (SAR) that is communicated internally to other projects and sectors that may be affected.

The risk assessment for the positioning control systems is based on severity categories and probability (likelihood) levels to define the corresponding risk matrix. These categories and levels are defined in IMO (references [6] and [7]) and internally [12]. At any time in this process if there is factual evidence that an event may occur that would result in a product hazard, a Safety Hazard Report shall be raised.

Unsafe Conditions will be recorded in the Safety Hazard Report, which is used to communicate the Unsafe Condition to relevant Rolls-Royce recipients and external authorities as necessary. The Safety Hazard Report will only be closed when an appropriate level of safety is demonstrated. All affected deliveries must have implementation plans to achieve the appropriate level of safety.

Vessel owners are informed of confirmed positioning software issues on delivered systems through a service letter. This contains detailed information about the possible control system failure, how it may occur and can contain sufficient information to prevent the failure from arising again based on operator awareness.

Non-critical bugs are updated during routine service missions. Critical bugs with a high risk may warrant a separate visit onboard to update the software.

Conclusion

Managing the development of a software control system that is installed on vessels around the world is no trivial task. An organization is required to maintain software and support DP operators and vessel owners. To establish software safety established standards must be implemented and followed up. The ongoing software development should be transparent to all participants and the software should be organized such that it is easy to retrieve a former version and recreate situations. An effort must be made to have the highest software quality possible the code should be readable, flexible and testable.

Acknowledgements

The author is grateful for the support of the Rolls-Royce Marine Automation & Control department and in particular the DP development team in writing this paper.

References

[1] ABS Guide for Integrated software quality management (ISQM), September 2012, American Bureau of Shipping

[2] DNV-OS-D203 Integrated Software Dependent Systems, December 2012, DNV GL, https://exchange.dnv.com/publishing/codes/docs/2012-12/Os-D203.pdf

[3] DNV-GL Enhanced System Verification, Rules for classification of ships, Part 6 Chapter 22, July 2013, DNV GL, https://exchange.dnv.com/publishing/rulesship/2013-07/ts622.pdf.

[4] IEEE 12207-2008 - ISO/IEC/IEEE Standard for Systems and Software Engineering — Software Life Cycle Processes. 2008. doi:10.1109/IEEESTD.2008.4475826. ISBN 978-0-7381-5663-7.

[5] IMO, MSC/Circ 645, Guidelines for vessels with dynamic positioning systems, International Maritime Organization, 6 June 1994.

[6] International code of safety for high speed craft, Annex 4 Procedures for failure mode and effects analysis (FMEA), International Maritime Organization, Resolutions from the sixty-third session of the Maritime Safety Committee, May 1994.

[7] International code of safety for high-speed craft, Annex 3 Use of probability concept, International Maritime Organization, Resolutions from the sixty-third session of the Maritime Safety Committee, May 1994.

[8] LaTeX http://www.latex-project.org

[9] MISRA, http://www.misra.org.uk

[10] MTS DP Committee, TECHOP Software testing, September 2014 http://dynamic-positioning.com/files_mailing/TECHOP Software Testing.pdf

[11] Rolls-Royce, Good Practice Framework

[12] Rolls-Royce, Identify and Sentence Safety Concerns, GP PS 3.1 Issue 3

[13] Rolls-Royce, Resolve Unsafe Conditions, GP PS 4.1 Issue 5

[14] Rolls-Royce Marine, Common Look and Feel, GUI Style Guide v 2.0.

[15] Rolls-Royce Marine, Positioning Safety Management Plan – Positioning Products, POS-OSM001-01SP

[16] Rolls-Royce Marine, Software Coding Guidelines – Positioning Products, POS-OSM001-04QS

[17] Rolls-Royce Marine, Software Development Plan, POS-OSM001-01QS

[18] Rolls-Royce Marine, Software Quality Plan, POS-0CM001-05QS

[19] Stewart, D. B. (2006), Twenty-Five most common mistakes with real-time software engineering, Embedded Systems Conference 2006.

.