XA9846500

METHODOLOGY OF FORMAL SOFTWARE EVALUATION

JAN TUSZYNSKI
Sydkraft Konsult AB
Malmö, Sweden

Abstract

Sydkraft AB, the major Swedish utility, owner of ca 6000 MW_{el} installed in nuclear (NPP Barsebäck and NPP Oskarshamn), fossil fuel and hydro Power Plants is facing modernisation of the control systems of the plants. Standards applicable require structured, formal methods for implementation of the control functions in the modern, real time software systems. This presentation introduces implementation methodology as discussed presently at the Sydkraft organisation. The approach suggested is based upon the process of co-operation of three parties taking part in the implementation; owner of the plant, vendor and Quality Assurance (QA) organisation. QA will be based on tools for formal software validation and on systematic gathering by the owner of validated and proved-by-operation control modules for the concern-wide utilisation.

1. Software project organisation

1.1. General

QA of the real-time systems has two main approaches available:

- vendor evaluation
- product evaluation

QA through vendor evaluation is presently the most favoured approach. Assumption that "good vendor will guarantee right quality.... " is comfortable for both plant owners and vendors. Vendor's assurances of ".. full turn-key, functional responsibility based on our standard products ..." sounds nicely for the owner as it will simplify his organisation for both purchase and implementation. The problems will show up first when guarantees are concerned. There are no vendors willing to cover dominating costs of the control system modernisation; cost of plant stand-still.

The method recommended will accordingly include direct product evaluation by application of formal tools for verification and validation (V&V). Application of formal tools puts anyhow special demands on owner's and vendor's organisation. That is generally valid for any tools having meaning in a context of work pattern only.

1.2. Standards and recommendations

Power industry and especially nuclear one is bound to follow standards and recommendations. Methodology described in this paper is based on,

- IEC880: Software for computers important to safety for NPP.
  Including recent updates as Supplement to IEC880 and draft IEC880-1.
- The Swedish recommendations corresponding IEC-880 are,
  KSU-TBE106: Programmable control systems. Software

Other standards give recommendations for general industrial systems, rules of classification, etc.:

- IEC1508: (Draft) Functional safety of electrical/electronic/programmable electronic safety related systems
- IEC SC65A WG9: Software for computers in the application of industrial safety-related system"
- IEC1226: NPP- Instrumentation and control systems important for safety. Classification
- IEEE 308 (384, 379) - Standard Criteria for Class 1E Power Systems for Nuclear Power Generating Stations
- DIN V 19250: Grundlegende Sicherheitsbetrachtungen für MSR-Schutzeinrichtungen

Draft IEC880-1 chapter 2 recommends formal methods for software procurement meant here as "methodical framework"[1] consisting of methods for formal specifications, formal verification of specifications, formal software development. The same norm requires *proof obligations* defined in mathematical terms. *Formal proof* is then defined as mathematical proof of fulfilment of proof obligations. Formal proof is practically possible only if special tools are available.

1.3. "Third party"; external QA organisation

Procurement of the control systems is normally done through tendering when the owner selects the vendor[2]. Both parties have the same implementation aim but different interests. All differences are supposed to be regulated in the contract defining strictly division of responsibilities. The contract includes demands on quality assurance and thus request for a third party to be involved; QA organisation. QA organisation can be internal or external. Internal QA, inside owner's or vendor's organisation, concerns mainly verifications, i.e. check-up that tasks performed comply with demands of earlier stages of the project. External QA goes in between the owner and the vendor and concerns mainly validation, i.e. check-up of the compliance between demand (specification) and the final result.

This study concerns mainly an external QA organisation which shall be independent of both the vendor and in some degree of the owner[3]. External QA organisation will be the main user of the formal tools for validation of real-time systems.

---

[1] "formal" means then; conceptually right, according to fixed rules.

[2] Selecting of the vendor is usually complicated process concerning naturally economy but in a great degree evaluation of the vendor's capability to deliver real-time system of required quality. Vendor's evaluation can be done according to ISO9000 or still better by application of various software procurement models as e.g. (CMM) Capability Maturity Model

[3] independence is defined by the contract or by valid standards and means often participation of authorities

2. The process of software procurement

Procurement of SW shall be done in well defined stages[4], mainly as the following,

- (FA): Functional Analysis.
  Crucial activity defining, according to controlled process requirements, functions of the real-time system.
- (FS): Function Specification.
  Functions as identified by FA are here described formally and unambiguously.
- (DP): Design and Programming.
- (Int): Integration
  of Soft and Hardware into the common, final product
- (Ver): Verification,
  actually not a stage but a QA process where particular project stages will be checked up and accepted
- (Val): Validation,
  tests of conformance between FS and functions delivered

2.1. Types of software

Two groups of SW products are concerned,

1. Operating system; principally integrated part the vendor's system.
2. Application functions; executing the actual, process dependent functions required of the control system.

QA methodology discussed here concerns mainly application functions. Operating system will be treated generally in the same way as hardware; i.e. as a program environment for applications provided by the vendor. It means that all formal test of applications must be done in vendor's target environment of the real-time system. Exclusion of the operating system from formal testing does not exclude that system from the particular interest of the owner. Operating system is presently main CCF[5] factor of redundant links of control systems.

2.2. Modularity

QA shall be based on modularity. Division in modules (Design and Programming) shall be formalised according to the following,

1. *Structuring and modularisation* of the software according to Function Specification, most often on the object control level, get fixed "standardised" form of "type-circuits" (figure 1). The control functions on group and block level will be normally based on "type-tools" (e.g. for sequence control)[6]
2. Modules shall be *classified* according to required safety and availability (figure 2)
3. Class and function shall be described formally in the *specification of module requirements.*
4. Modules shall be procured according to well established *project routines* concluded by *module validation* (certification)
5. Modules when ready and certified shall be placed in *software libraries.*

---

[4] main condition for any verifications

[5] Common Cause Failure; existence of a common function in separated links of the system which in case of failure can stop functioning of all links.

[6] Division in object, group and block levels corresponds to the traditional control system structure as applied presently for Swedish Power Plants.

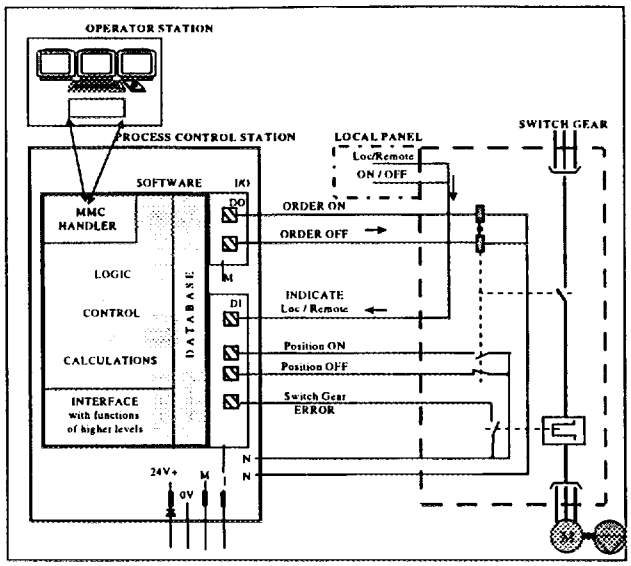

Figure 1: Components of the complete type-circuit

2.3. Quality through re-application. Software libraries.

Basis for the proposed QA system is re-application of the software modules. Quality can be achieved through systematic product improvement through experience gathering and continuous reduction of systematic errors in the specifications and software. This policy is clearly defensive by assuming errors in all new software.

The main conclusion of that assumption is that new software shall not be allowed in

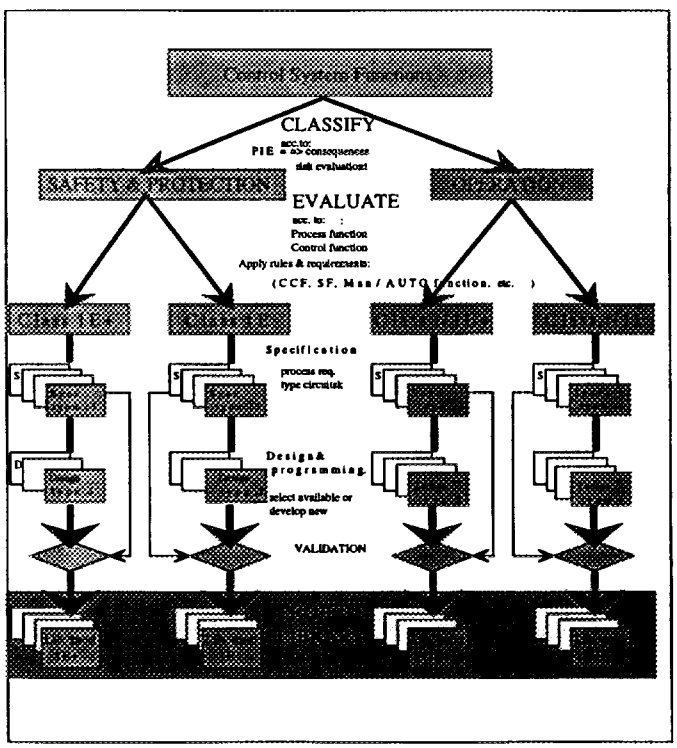

Figure 2: Modularity through division in classes

the highest class of the control modules; i.e. safety systems. The problem can be solved by initial application of modules to the lower classes only. Gradual advancement to higher classes will be allowed parallel to operating experience.

QA through re-application requires software library (figure 3) as a base for all software procurement. The following will be required for the library:

- fixed routines for module documentation, validation, storage, retrieve, update and revision
- library structure, e.g. according to module applications and classes
- means for creating library subsets suiting various power plants, organisations, etc.
- special attention shall be given to validation of library modules. The modules shall be certified for storage in the library through validation based on formal methods.
- the rules for module handling shall be adopted to class of modules.
- responsibility for validation, maintenance, up date, etc. of each library subset shall be well defined.

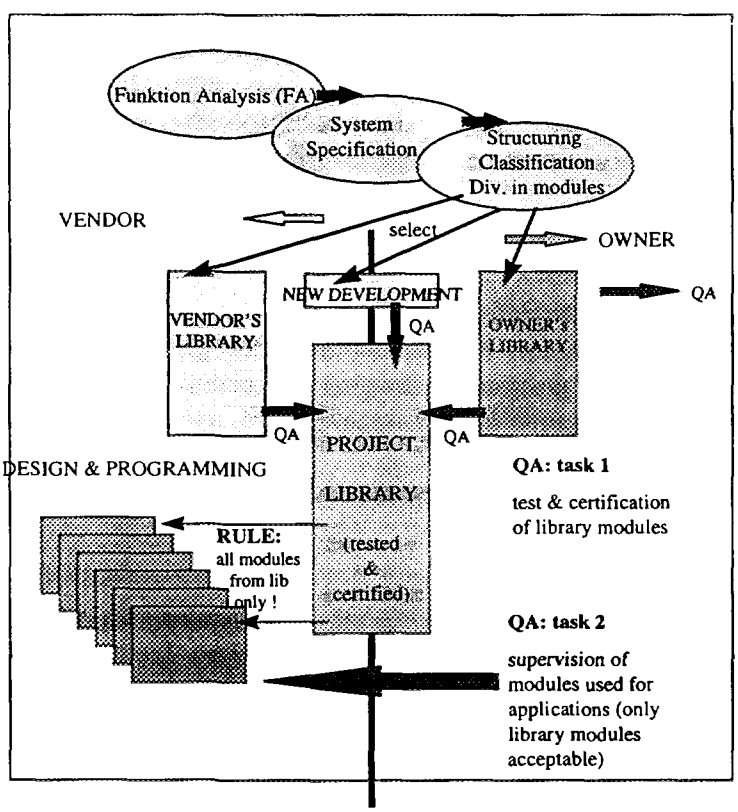

Figure 3: Software implementation through project library

Management of library shall be facilitated by the special software maintenance system. The system shall allow configuration of the project specific library. The project library shall gather all the structures, tools and modules identified during structuring and modularisation of the software. The identified elements can be subsequently taken from the owner's or / and vendor's libraries. In case suitable elements are not available the new elements must be developed. Inclusion of all modules into project library shall follow owner's rules for library handling. The project library will become the basic element of the project bound QA.

One of the main advantages of a formally organised libraries will be possibility to trace down development process of any module purchased or developed for the actual application.

3. Verification of Specification

An external QA consists mainly of validation of the final software product. As validation compares the product with product specification it must be assumed that specifications are correct. Verification of specifications becomes accordingly crucial for software correctness.

Verification methods start usually from recognition of potential errors in specifications, e.g.

- principal function errors caused by misunderstanding of the process controlled, errors in that process etc.
- erroneous object assignment
- errors in syntax and semantics
- errors of consistency

De-bugging of errors of assignment, syntax, semantics and consistency can be handled by formal specification languages and corresponding verification tools.

The problem of principal function errors is normally approached through the functional analysis (FA). FA, carried by the owner (process supplier), is based upon fixed rules and criteria applicable to the process controlled, and on a deep knowledge of that process.

There is a number of well established FA methods, majority based on some form of failure / event tree. The method used widely in nuclear industry is PSA[7] dealing with event tree initiated by the probable events of accidents.

FA will be documented in functional circuit diagrams, in several levels of detail, starting with overviews and "zooming" down to the formal language function specifications of modules and structures employed [8].

4. Final product validation

4.1. General

Validation means final product acceptance, or formal proof as defined by IEC880, licensing the product for inclusion in the software library. Validation will be performed by the QA organisation during design and programming period with the final module acceptance during FAT[9].

Validation can be performed through:

- direct check-up according to test programme
- check-up by comparison

Both methods require special tools. Validation will be applied to both specific modules of the library and to the groups of modules interconnected to perform specified application function.

Test objects for all validations named here are vendor's real-time target systems.

---

[7] Probabilistic Safety Analysis

[8] Various methods of FA are well described in the literature: e.g. ref [8]

[9] Factory Acceptance Tests

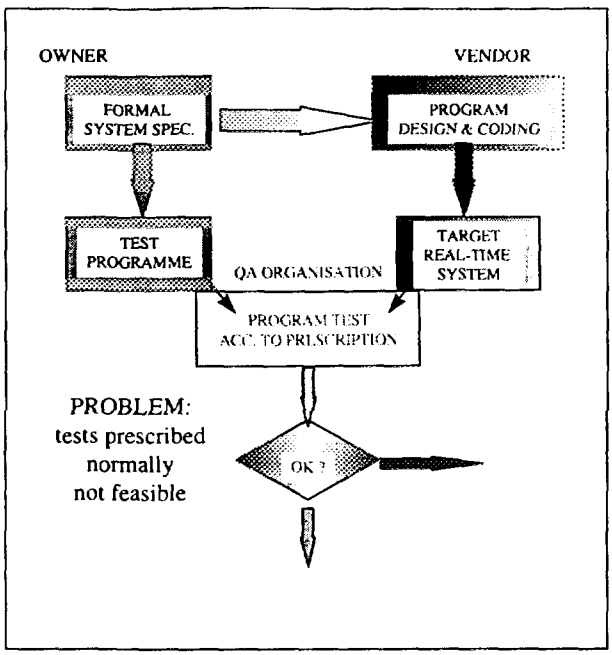

Figure 4: Validation through test programme

4.2. Validation through test programme

Validation through test programme (figure 4) assumes that such a programme is feasible and can be run in a limited time period. That assumption can be realised for simple modules only. Validation of this kind will require,

- complete test programmes defining test input data values and sequences and expected test results. Test programmes must be automatically generated from functional specifications.
- test environment including test data generator and test object response analyser.

All criteria for the product acceptance must be included in a response analyser

4.3. Validation by comparison

Validation by comparison is based on procurement of an alternative to target program produced by vendor (figure 5). Validation of this kind will require,

- "pattern program"[10] generated automatically from the formal functional specification.
- complete test environment including test data generator and comparator of responses.

Validation will be here realised by parallel inclusion of identical test data into a pattern and a target. All response deviations must be reported by the comparator and subsequently analysed. Main advantages of this method are simplified selection of test data (through e.g. statistical approach) and direct representation of the actual specification through its pattern

[10] VTT Finland (P. Haapanen el al) uses here "test oracle" [7]

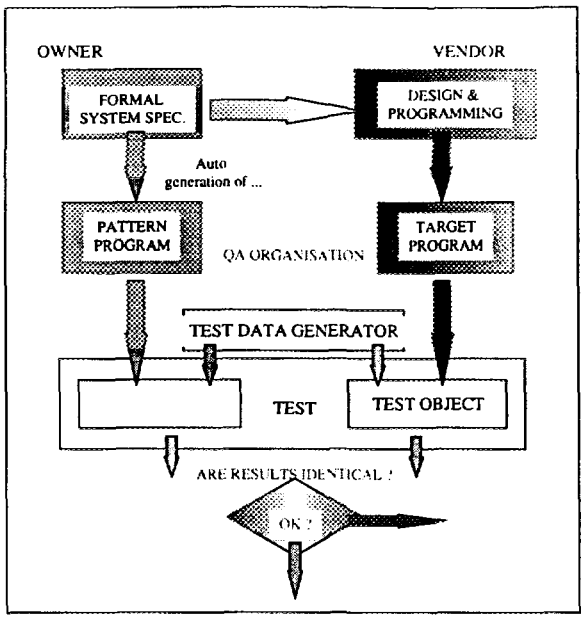

Figure 5: Validation through comparison

REFERENCES

[1]: "An introduction to Formal Methods", A. Diller, Z. Wiley and Sons, Inc., NY 1990

[2]: "An Integrated Formal Approach for Developing High Quality Software for Safety-Critical Systems", Meng Ouyang, Michael W. Golay, MIT, Michael S. Novak, ABB Combustion; Report No. MIT-ANP-TR-035

[3]: "Formal and Abstract Software Module Specifications - A Survey", Yabo Wang, Tech. Report 91-307, ISSN 0836-0227, Queen's University, Ontario, Canada, 1991.

[4]: "Validation and Reliability Testing of Safety-Critical Software for Wolsong NPP Units 2, 3 and 4", J.S. Baxter, el al Atomic Energy of Canada, Ltd, and H.B. Kim, el al, Korean Atomic Energy Research Institute.

[5]: "The Use of an Integrated Test Environment in the Design and Verification of Digital Closed Loop Automatic Control Systems for the Sizewell B PWR", G.P. Paulson, el al, Westinghouse; K. Drury, el al, Nuclear Electric Ltd; Paper at the 1996 ANSI Meeting, Proceedings NPIC & HMIT '96.

[6]: "Qualification of an Advanced Digital Safety System", Werner Bastl, Dieter Wach; Institute for Safety Technology (ISTec) GmbH; Paper at the 1996 ANSI Meeting, Proceedings NPIC & HMIT '96.

[7]: "Validation of Programmable Automation Systems for Safety Critical Applications", Pentti Haapanen, el al, Technical Research Centre of Finland (VTT); International Workshop on Licensing Issues ...., March 1996, GRS/ISTech, Munich

[8]: "Dependability of Critical Computer Systems,Guidlines" The European Workshop on Industrial Computer Systems Technical Committee 7 (EWICS TC7). del 1, 2 och 3, F.J.Redmill (1988).

[9]: "Digital Instrumentation and Control Systems in Nuclear Power Plants. Safety and Reliability Issues" National Research Council. Procured on request of USNRC. National Academy Press, 1997.