

Conducting Verification & Validation following DO178B guidelines

In the avionics industry, software was used to extend and modify the capabilities of mechanical and analog systems. This was done in a manner much simpler than redesigning or modifying the hardware components. It was seen as an aid to inexpensive modification and functional extension of an originally inflexible all-hardware design. During system certification, it became clear that the classical statistical methods of safety assessment for flight critical software based systems were not possible. This is because unlike hardware where you have MTBF (Mean Time Between Failures), software either works or doesn't! Hence alternate knowledge and method(s) were necessary to establish equivalent integrity to deal with design errors.

Imagine you have alighted the aircraft to go across the Atlantic from London to Newyork. You come to know that the software which your team has developed & verified is the one which is used in the Aircraft which you are about fly! *Would you be desperate to get out of the Airplane ? Or would you be relaxed in the knowledge that nothing would happen to the plan because of the software you developed?*

Need for Standard

Because of the life criticality involved in the avionics software, it became necessary to establish a uniform, consistent definition of criteria for substantiating evidence for the absence of critical design errors, answering the following questions:

- 1.How was it known that the testing was comprehensive and complete?*
- 2.How was it known that the system requirements were comprehensive and complete?*
- 3.How was it known that the software requirements were comprehensive and complete and interpreted the system requirement accurately?*
- 4.How do we provide proof that a design or implementation error, which may be present, cannot produce a safety critical situation?*

Verification and Validation became new terms. Verification provided the proof that a system was built to the requirements and validation established that the requirements were complete and correct.

Developing a software for Aircraft differs from

- developing software for other embedded systems like entertainment electronics which become obsolete fast as their customer is bothered about new fashionable features rather than the safety of the equipment. Here the schedule is entirely driven by market needs which often vary substantially in a year
- developing regular computer software like a word processor where business/life criticality are negligible but compatibility with older versions, adhering to defacto standards and again meeting the market expectations is more important. Here again the quality is compromisable.
- Banking software where there is a financial risk but unlike Aerospace software, risks concerning interfacing to external hardware is minimal. Also Banking software may not be responsible (i.e., directly) for killing it's user.

In aircraft risk for life (pilot, crew, passengers) as well as great financial risk is there. Also the software you write does not change for 15 to 20 years and hence the stringent standard ...

Hence it requires that the avionics software be produced such that it minimizes or removes the risk of a malfunction or failure. This gave birth to DO-178.

Key information on DO178B

The verification of such a software as per DO-178B standards depends on the level of criticality of the software. Software level is based upon the contribution of software to potential failure conditions as determined by the system safety assessment process. The software level implies that the level of effort required to show compliance with FAA (Federal Aviation Administration) certification requirements varies with the failure condition category. The software level definitions are:

- a. Level A: Software whose anomalous behavior, as shown by the system safety assessment process, would cause or contribute to a failure of system function resulting in a catastrophic failure condition for the aircraft. For example software whose failure conditions would prevent continued safe flight and landing.
- b. Level B: Software whose anomalous behavior, as shown by the system safety assessment process, would cause or contribute to a failure of system function resulting in a hazardous/severe-major failure condition for the aircraft. For example software whose failure conditions would reduce the capability of the aircraft or the ability of the crew to cope with adverse operating conditions to the extent that there would be physical distress or higher workload such that the flight crew could not be relied on to perform their tasks accurately or completely.
- c. Level C: Software whose anomalous behavior, as shown by the system safety assessment process, would cause or contribute to a failure of system function resulting in a major failure condition for the aircraft. For example software whose failure conditions would reduce the capability of the aircraft or the ability of the crew to cope with adverse operating conditions to the extent that there would be a significant increase in crew workload.
- d. Level D: Software whose anomalous behavior, as shown by the system safety assessment process, would cause or contribute to a failure of system function resulting in a minor failure condition for the aircraft. For example software whose failure conditions would cause slight increase in crew workload, such as, routine flight plan changes, or some inconvenience to occupants.
- e. Level E: Software whose anomalous behavior, as shown by the system safety assessment process, would cause or contribute to a failure of system function with no effect on aircraft operational capability or pilot workload.

Verification effort table based on the level of software

	Objective	Ref.(DO-178B)	Applicabilty by SW level				Output
	Description		A	B	C	D	Description
1	Test procedures are correct.	6.3.6b	●	○	○		Software Verification Cases and Procedures
2	Test results are correct and discrepancies explained.	6.3.6c	●	○	○		Software Verification Results
3	Test coverage of high-level requirements is achieved.	6.4.4.1	●	○	○	○	Software Verification Results
4	Test coverage of low-level requirements is achieved.	6.4.4.1	●	○	○		Software Verification Results
5	Test coverage of software structure (modified condition/decision) is achieved.	6.4.4.2	●				Software Verification Results
6	Test coverage of software structure (decision coverage) is achieved.	6.4.4.2a 6.4.4.2b	●	●			Software Verification Results
7	Test coverage of software structure (statement coverage) is achieved.	6.4.4.2a 6.4.4.2b	●	●	○		Software Verification Results
8	Test coverage of software structure (data coupling and control coupling) is achieved.	6.4.4.2c	●	●	○		Software Verification Results

The following verification documents supporting compliance to DO-178B need to be produced:

1. Software Verification Plan
2. Software Verification Cases and Procedures (Unit and Integration test plan and procedures)
3. Qualification test procedures (Formal Evaluation Plan and Procedures)
4. Problem Reports
5. Software Verification Results

The planning of verification activities (like writing software verification plan, various test plans) is mandatory for a project which follows DO-178B standard. Also as illustrated in the verification effort table, based on the level of the software, structural coverage is very stringent when compared to a 'regular' project. Submission of problem reports to the certification authority is a must.

In general, managing Verification activities poses lots of problems.

- Any delay in the development activities directly relates to slippage of verification activities. The project manager is therefore typically forced to shorten the verification time which puts lots of pressure on the verification team.
- If the defects found by the verification team are not documented systematically, they may remain unsolved and finally remain unverified by the evaluator again.
- Poor documentation of the defects may lead to inefficient monitoring of project progress. This leads to poor tracking of defects.

The planning and co-ordination of verification activities amongst the team can be made easy and handled efficiently by using tools like SmartWorks (Smartworks-PE is a free download <http://www.smartworks.us>). I used this tool inorder to achieve my objectives.

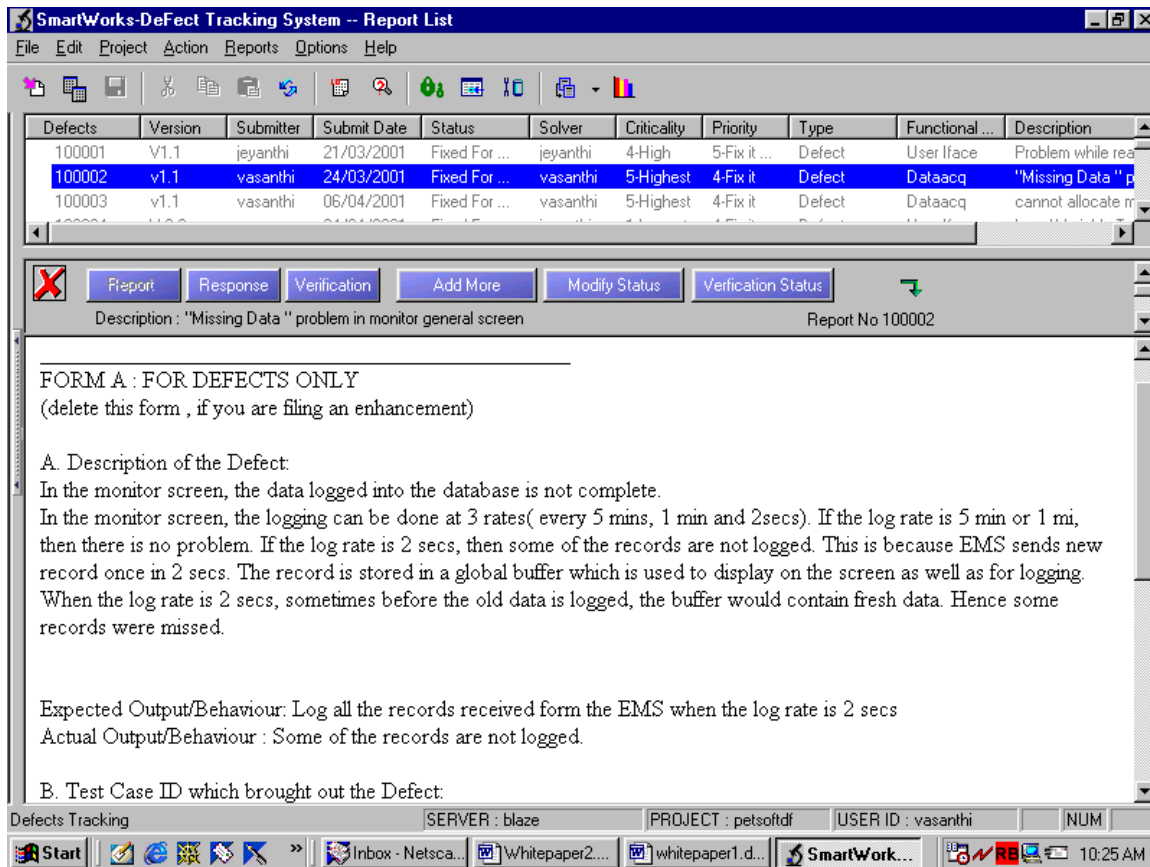
SmartWorks –PMT (Project Planner) can be used to plan the entire verification process activities. The various activities like writing test plans (Unit test plan, integration test plan, Formal evaluation plan), performing code review, conducting the various tests as per their plans, generating reports....can be created as tasks in PMT with task owner and the planned start and end date being assigned. PMT automatically sends a mail notifying the owners of their assignments. The document generated as a result of any task can be attached against the task. This allows the maintenance of the documents electronically. For example the test cases and procedure documents can be attached against its task and hence viewed by all the team members.

The status of the task (new task, task completed) can be maintained in PMT. The task owner is reminded with a mail if the task is not completed when the planned end date approaches. This helps one to monitor the project progress efficiently. Also whenever a change in plan occurs (may be due to slippage, attrition or cost escalation) Project planner recognizes that a risk had happened and suggests contingency plan from it's database. This helps the team to know how many risks happened during the course of a project and how effective were the contingency plans in handling the risks. It allows you to import risk database from similar projects done in the past by a company. This allows the project manager to be well prepared for the risks which are likley to occur and quickly take effective corrective action

Task Id	Task Name	Predec...	Pln Start D...	Pln End Date	Pln Dur...	Owner	Report Stat...	Cost
180	Unit Test Plan		01 Apr 2002	31 May 2002	46		30% Comp...	0.000
181	Preparation of Unit Test Plan		01 Apr 2002	10 May 2002	30	shobha	75% Comp...	0.000
182	Review of Unit Test Plan	181	13 May 2002	30 May 2002	15	vasanthi	Started	0.000
183	Release of Unit Test Plan	182	31 May 2002	31 May 2002	1	shashi	New Task	0.000
184	Delivery of Executable code and Unit...		25 Jun 2002	25 Jun 2002	1		New Task	0.000
185	Unit Testing		26 Jun 2002	06 Sep 2002	57		New Task	0.000
186	Carry out Unit Test - I Cycle	179	26 Jun 2002	08 Aug 2002	35		New Task	0.000
187	Report Defects in SMARTWORKS	186	09 Aug 2002	12 Aug 2002	2		New Task	0.000
188	Fix Defects	187	13 Aug 2002	26 Aug 2002	10		New Task	0.000
189	Carry out Unit Test - II Cycle	188	27 Aug 2002	06 Sep 2002	10		New Task	0.000
190	Preparation of Unit Test Report	189	09 Sep 2002	12 Sep 2002	3		New Task	0.000
191	Review of Unit Test Report	190	13 Sep 2002	13 Sep 2002	1		New Task	0.000
192	Integration Test Plan		02 Apr 2002	03 Jun 2002	46		New Task	0.000
193	Preparation of Integration Test Plan		02 Apr 2002	13 May 2002	30		New Task	0.000
194	Review of Integration Test Plan	193	14 May 2002	31 May 2002	15		New Task	0.000
195	Release of Integration Test Plan	194	03 Jun 2002	03 Jun 2002	1		New Task	0.000
196	Software Review and Analysis Report		26 Jun 2002	16 Aug 2002	40		New Task	0.000
197	Preparation of Software Reviews a...		26 Jun 2002	01 Aug 2002	30		New Task	0.000
198	Review of Software Review and A...	197	02 Aug 2002	16 Aug 2002	10		New Task	0.000
199	Delivery of Exec. code, Unit Test Re...		17 Sep 2002	17 Sep 2002	1		New Task	0.000
200	Assembly & Testing of 15 Units of GN...		01 Jul 2002	17 Sep 2002	58		New Task	0.000
201	Integration Test		18 Sep 2002	22 Nov 2002	45		New Task	0.000
202	Carry out Integration Test - I Cycle	199	18 Sep 2002	30 Oct 2002	30		New Task	0.000
203	Report Int. Test Defects in SMAR...	202	31 Oct 2002	06 Nov 2002	3		New Task	0.000
204	Fix defects found durina Integratio...	203	07 Nov 2002	07 Nov 2002	1		New Task	0.000

SmartWorks – DFT (Smart Tracker) can be used for defect tracking. Smart Tracker allows one to customize many of its features suiting their project needs. For example, the templates for problem, response and verification reports can be customized based on specific project needs.

The evaluator submits defects while conducting testing. This forms the problem reports



The PL/ PM of the project assigns the solver and the verifier of the defect. DFT notifies the solver /verifier of the assignment with a mail. After solving the defect, the solver fills in the response report in DFT. The verifier then verifies that the defect has been fixed (QA Passes) and fills in the verification report. In case the verifier finds that the problem is still not fixed, he/she makes the status of the defect as 'QA Failed' which signals the developer that the problem is not yet fixed. This cycle continues till the verifier finally verifies that the problem is fixed.

At any point of time PM/PL can therefore monitor the project progress and replan a task if need be. Project Planner allows one to replan the task and associate the risk factor with it due to replanning.

To conclude I would say, SmartWorks helps a long way in planning and tracking the verification activities. However I would appreciate, if it was integrated with a testing tool, which submits a report (test case id and the result) automatically in the Smart Tracker.

Vasanthi heads the verification & validation team at Accord Software & System which is currently engaged in executing a project which adheres to DO178B standards. The project is concerned with evaluating a navigation equipment for flight worthiness and eventual FAA certification. In case you have any questions you may reach her at vasanthi@accord-soft.com