2009 First International Conference on Advances in System Testing and Validation Lifecycle

Using the Testability analysis methodology for the Validation of AIRBUS Systems

Fassely Doumbia, Odile Laurent
Systems Design
AIRBUS France
Toulouse, France
{Fassely.Doumbia, Odile.Laurent}@airbus.com

Chantal Robach, Michel Delaunay
Systems Design & Test
LCIS – Grenoble Institute of Technology
Valence, France
Chantal.Robach@esisar.grenoble-inp.fr

***Abstract*—The experiments, carried on AIRBUS systems, show that testability analysis can ease system formal detailed specifications validation activities. Indeed, testability information can highlight testing efforts, guide functional tests definition, facilitate detailed specification coverage analysis against system requirements, and support tests coverage analysis against formal detailed specification.**

Keywords- requirement; testability analysis; information flows; testing strategy; test; coverage analysis

I. INTRODUCTION

Development of avionics systems must comply with the DO-178B standard [11]. The Validation and Verification (V&V) process is very demanding and contributes to high development costs. Therefore, innovative methods and tools that can alleviate and efficiently support V&V activities are of great interest for the aeronautics domain.

In this context, the testability analysis methodology proposed in this paper can offer useful methods to support the validation of systems formal detailed specification. Our testability approach deals with detailed specification relying on data-flow languages such as SCADE [4]. The testability analysis proposed is based on SATAN (System's Automatic Testability ANalysis) technology [2, 6]. This approach determines information flows and metrics that can be helpful information for system designers and system design validation engineers.

A significant number of research projects have dealt with software testability and a set of complexity metrics have been proposed. The cyclomatic number [7] measures the number of linearly independent paths through the control graph built by using a program source code. The Npath metric [8] computes the number of possible execution paths through a function. Freedman [5] introduced the domain testability of software components based on controllability and observability. Voas and Miller [10] proposed the DRR (Domain/Range Ratio) metrics which exhibit fault-hiding tendencies of software subcomponents. This technique can be used to predict a subcomponent's ability to cause program failure if it contains a fault. Do, Le Traon and Robach [1, 2, 6] proposed testability measurement applicable to data-flow designs. The study presented in this paper is based on the same approach.

This paper proceeds as follows: Section II introduces the SATAN technology and the associated testability analysis. Section III describes the AIRBUS flight control systems validation process. Section IV proposes an enhanced validation process integrating the testability analysis approach. Experiments results are depicted in Section V. Finally, conclusion and perspectives are given in section VI.

II. TESTABILITY ANALYSIS

"Fig. 1" gives a simple view of testability analysis, proposed by the SATAN technology. This approach has been initially applied to hardware systems [2]. Further studies [1, 6] demonstrated that this approach could be used for analyzing the testability of data-flow designs. This method provides metrics that allow the identification of critical parts (which can be hard to test). It also identifies information flows which are a set of operators involved in the computation of one output from relevant inputs. They can be used to guide tests definition in the system design validation cycle.

The SATAN approach is based on a testability model that represents the transfer of information through the system. This model is called *ITM* (Information Transfer Model). Information flows are identified from this ITM (Section A). Test strategies (Section B) can be applied to select relevant information flows to help the test generation process.

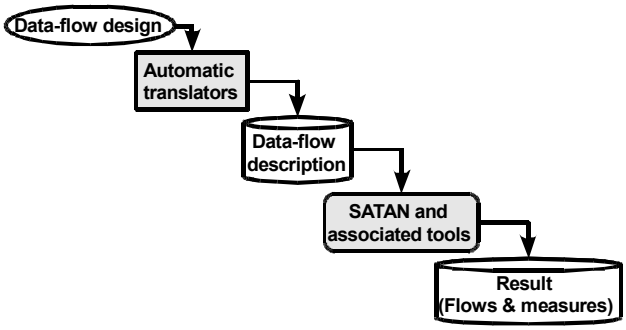

Figure 1. Testability analysis process.

A. Information Transfer Model

The testability model is a graph defined by a set of places, transitions and arrows "Fig.2". Places represent inputs, constants, functional modules, outputs and test injection points specified in the system. Transitions express information transfer modes between places. Arrows

978-0-7695-3774-0/09 $26.00 © 2009 IEEE
DOI 10.1109/VALID.2009.8

connecting places and transitions represent information media throughout the system.

Three different information transfer modes are used in an ITM "Fig. 2":

- Junction mode: the destination place needs information from all source places;
- Attribution mode: the destination place needs information from one of several source places;
- Selection mode: the same information is sent from the source place to some destination places.

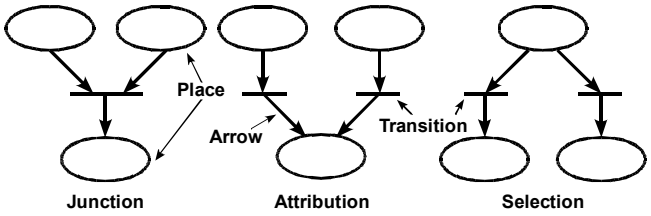

Figure 2. Information transfer modes.

A data-flow description of a system is hierarchically composed of operators. Each operator has an elementary ITM. This model corresponds to the data-flow representation of the operator. According to the level of testability analysis, this elementary model can be more (or less) detailed. The basic representation of an operator associates a functional module to each output. "Fig. 3" illustrates the notion of elementary ITM using graphic representations.

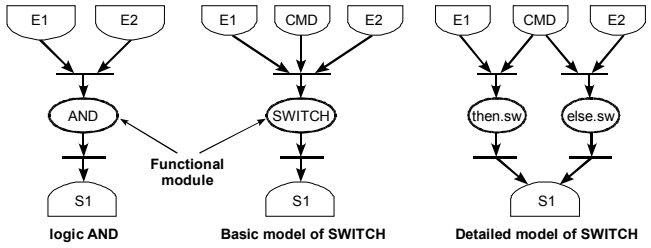

Figure 3. Elementary ITM of logic AND and SWITCH operators.

In this representation, inputs and outputs are depicted by semicircles, modules by circles, and transitions by bars.

The system ITM is obtained by concatenating the elementary ITMs.

B. *Information flows*

SATAN technology identifies flows from a system ITM. A information flow is an information path that carries information from one or several inputs, through modules and transitions, to one output. Several information flows can be associated to an output.

"Fig. 4" represents the SCADE specification of a system. The testability analysis, proposed by SATAN, shows it contains twelve information flows. Two information flows (F_1 and F_2) are depicted using bold lines. These flows are described below by a set of modules and output.

F_1 = {NOT_1, NOT_2, then.SWITCH_2, OR_2 | O2}

F_2 = {NOT_1, NOT_2, then.SWITCH_2, OR_2,
then.SWITCH_4, OR_3, then.SWITCH_5 | O1}

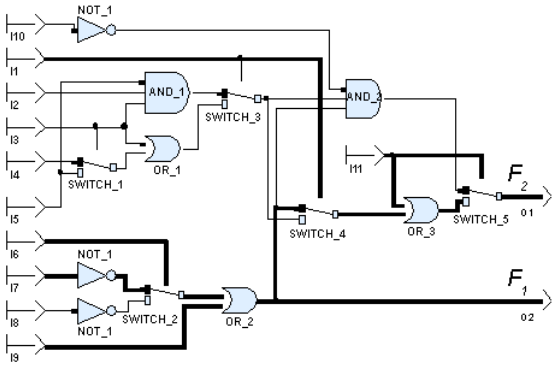

Figure 4. Graphical representation of a information flow

C. *Test strategies*

A test strategy defines the way to conduct test activities and to analyze test results. Test strategies allow the selection of a set of relevant information flows for testing purpose. Flows are chosen according to the following criterion: every place in the graph must be activated at least once in order to ensure the coverage of all operators. SATAN supports three strategies: *Start-Small* (progressive structural strategy) [9] suitable for the progressive detection of faults during system validation, *Multiple-Clue* (cross-checking strategy) [9] suitable for diagnosis during maintenance and *All-paths* [9] which chooses all flows contained in the ITM. We will focus on the Start-Small strategy in this paper.

The Start-Small strategy gradually covers the modules by choosing flows with an increasing number of covered modules. The main idea of this strategy is to minimize test and diagnosis efforts. The first information flow to be tested contains the minimal number of modules. The next one contains a minimal number of modules that are not activated yet. In this strategy, a new flow is tested only if all faults detected in previous flows are corrected, as depicted in "Fig. 5". Test cases are defined for each selected flow.

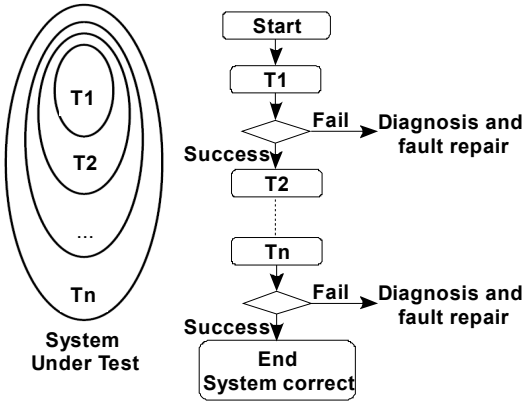

Figure 5. Start-Small strategy illustration

Start-Small strategy selects seven relevant information flows instead of the twelve initially determined for the system specification depicted in "Fig 4".

Choosing a relevant testing strategy depends on industrial practices and system development stage. In the next section, we present the AIRBUS system validation and verification process in order to point out the detailed specification validation activities.

III. AIRBUS VALIDATION AND VERIFICATION (V&V) PROCESS

In this section, we focus on AIRBUS flight control and Auto flight systems V&V activities. Flight control systems allow the pilot to control the aircraft in flight. Auto flight systems allow maintaining the flight path defined by the crew. They also control the aircraft and the engines. The V&V cycle of these systems relies on a generic V-cycle process for which validation activities are added due to the modeling process at system level "Fig. 6".

Three main levels can be identified in this V&V process. The "*Aircraft level*" is composed of aircraft level requirements definition, aircraft simulation, ground and flight tests activities. The *"System level*" represents the system specifications and design definition stage. The "*Equipment level"* corresponds to the implementation of system specifications and designs in real equipment.

We will focus on systems validation activities in the rest of the paper.

A. System validation cycle

System validation activities are mainly based on testing and traceability activities. The model validation activities *(5)* consist in executing tests on a desktop simulator dedicated to flight control system. This simulator embeds system functions code automatically generated from the SCADE model. The traceability process relies on a documents cascade and on a SCADE model managed by DOORS tool [4]:

- *SRD* (System Requirements Document) *(1)* specifying system requirements refined from aircraft requirements.
- *DFS* (Detailed Functional Specification) *(2)* document describing the system detailed functions that meet the system requirements.
- *TRD* (Test Requirements Document) *(3)* defining for each DFS function the tests data (functional tests description, tests vectors and expected tests results).
- SCADE model (or detailed specification) *(4)* corresponding to a formal implementation of the detailed functions and from which the embedded code is automatically generated.

The traceability activities, based on DOORS tools, consist in checking that:

- All the system requirements described in the SRD are considered in the system functions *(L1)*;
- The SCADE model implements only once all the systems functions specified in the DFS *(L2)* (neither under-specification nor over-specification);
- The tests defined in TRD cover all the functions of the DFS *(L3)* and the functions of the SCADE model *(L4)* (no missing tests).

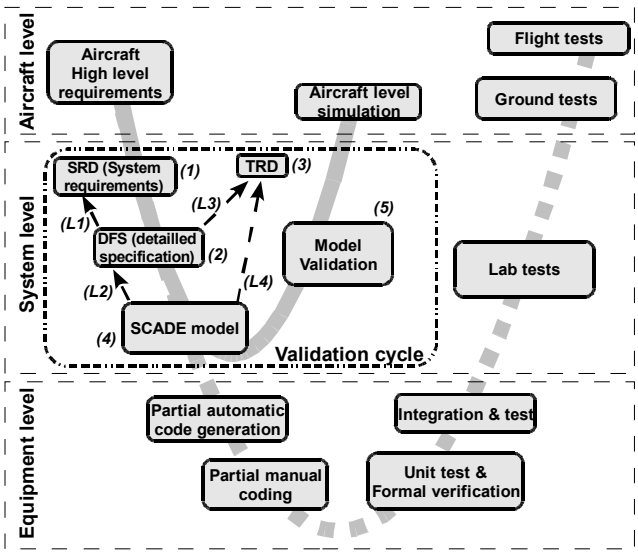

Figure 6. AIRBUS Flight control systems V&V process

In order to alleviate drastically the coverage analysis activities described above, we propose to rely on the information flows determined by SATAN on the SCADE model.

Indeed, if we can demonstrate that there is a clear link between information flows and DFS requirements, the relationship between SCADE model and DFS becomes obvious. Another important issue of this approach in the validation process is the definition of relevant set of tests ensuring the coverage of the SCADE model. Enough tests have to be identified to cover all the system requirements, but redundant tests must be avoided for cost-efficiency reasons.

The next Section IV defines a methodology based on testability analysis to support system coverage analysis.

IV. TESTABILITY ANALYSIS METHODOLOGY

Testability analysis information is useful only if it reflects the development process. Before describing the testability process (Section D), we present three testability approaches defined for AIRBUS needs.

A. Output variable approach

This first approach consists in identifying all the SCADE output variables involved in the DFS. The part of the SCADE model related to each output variable is extracted. Thereby, information flows are determined for each extracted part of the SCADE model. This approach allows a local testability view of the system for all output variables.

B. Set of output variables approach

This second approach consists in identifying the set of output variables per function. The part of the SCADE model related to this set of output variables is extracted and information flows are determined. It allows the testability assessment for each function defined in the system.

C. Component approach

In this third approach, we propose to split the SCADE model into independent parts from a data-flow point of view (called hereafter components). This approach consists in extracting each component and performing the testability analysis. It allows the analysis of system independent parts separately.

D. AIRBUS systems testability analysis process

The analysis process defined for AIRBUS systems is based on DFS document, TRD document and SCADE model. It is composed of three main activities: the testability analysis, the coverage assessment of the requirements against the SCADE model, and the test cases against the SCADE model.

The testability analysis stage is depicted in "Fig. 7". It consists in applying one of the three testability approaches defined in Sections A, B and C. Relevant information flows are selected using Start-Small strategy (Section II).

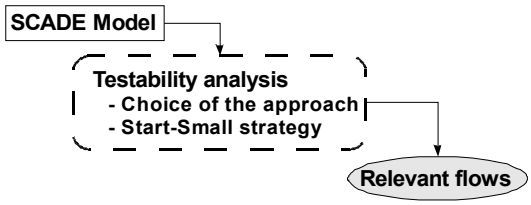

Figure 7. Testability analysis stage

The two other stages of this process are depicted in "Fig. 8". Five main activities can be identified in these stages.

a) Requirements output variables identification: This activity consists in using the DFS document to identify SCADE ouput variables involved in each system requirement definition.

b) SCADE model coverage analysis: Information flows are associated with output variables in this step. This coverage activity allows highlighting a link between requirements and information flows. It also points out the presence of:

- *Orphan flows*: information flow is said to be orphan if it has not any association with a requirement. The identification of these flows points out the presence of SCADE output variables which are not defined in the DFS. The presence of these flows points out either a possible over-specification of the SCADE model or the implementation of derived requirements in the SCADE model.
- *Missing flows*: This situation is due to the absence of flows associated with some output variables. The detection of these outputs means that no SCADE model part corresponds to these output variables. It highlights the SCADE model is incomplete: the implementation of some DFS requirements is missing.

In addition, for each DFS requirement, the minimum number of tests to be defined can be deduced from the number of information flows: at least one test per flow. It allows evaluating the testing effort.

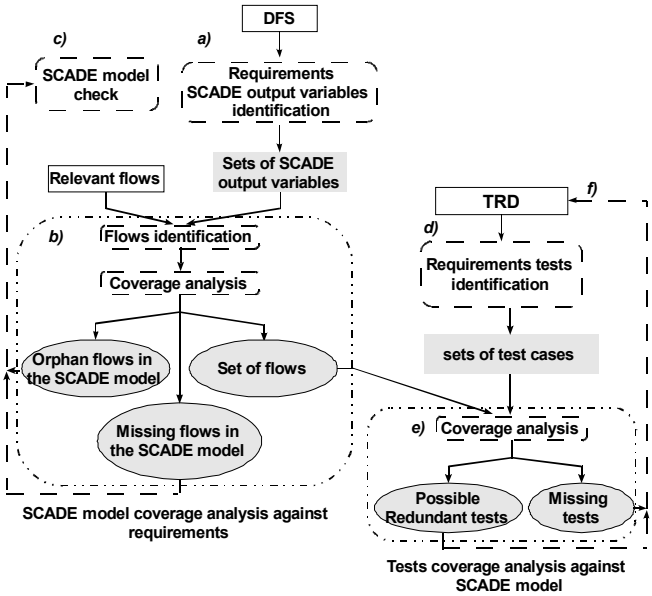

Figure 8. AIRBUS systems testability analysis process

c) SCADE model check: It consists in analyzing orphan and missing flows information and may lead to modifications in the SCADE model: deletion of the SCADE part related to orphan flows; addition of the SCADE part related to the unimplemented DFS requirements highlighted by the missing flows.

d) Requirements test cases identification: This activity consists in using the TRD to identify test cases associated with each requirement described in the DFS.

e) Tests coverage analysis: This step highlights the link, for each requirement, between relevant information flows and identified test cases. When a relevant flow associated with a requirement cannot be linked to a described test in the TRD, a missing test is identified. When a same flow is linked to different tests, potential redundant tests are identified. It is also pointed out that the number of determined flows is dependant of the ITM modeling (Section II). In the current study, the ITM modeling principle is based on branch and decision coverage criteria.

Otherwise, the relation between relevant flows and test cases can be used to define tests scheduling. Indeed, Start-Small strategy (Section II) proposes an order of execution of the tests related to the selected flows. This order aims at minimizing diagnosis effort.

In the following section, we apply our process to an academic example and present the results of its use for two industrial case studies.

V. CASE STUDIES

In this section, we depict two experiments: the pumping system which is academic and two Auto-flight systems

which are AIRBUS operational systems. For interpretation purpose: *(1)* means the "Output variable approach", *(2)* means the "Set of output variables approach" and *(3)* means the "Component approach".

A. A house pumping system

This academic case study controls the water supply of a house. Two pumps are used to specify this system: the first one brings up water from a well to fill a tank; the second pump supplies water to the residence. Three main functions can be defined for controlling this system: one manages the first pump; another controls the second pump and the last one performs the system's global status. These functions can be described by the following requirements:

- R1: the system shall actuate the first pump when the water-level in the tank decreases and reaches the "filling level";
- R2: The pumping shall stop when the water-level in the tank raises and reaches the "pumping idle level";
- R3: The first pump shall stop running when the temperature of the pump is abnormally high (only a push button can actuate it).
- R4: The second pump shall be actuated by the decrease in pressure due to the turning on of a tap in the residence. Its shutdown is provoked by the turning off all the taps in the residence;
- R5: the system shall warn and idle the second pump when water reaches the "warn level" (the water-level is lower than the "filling level");
- R6: The pump shall be idled when the water flow rate is too low;
- R7: The pumping shall stop when its temperature is abnormally high (only a push button can actuate it).
- R8: the system shall indicate its global status by taking into account the two pumps.

"Fig. 9" below shows the SCADE model of the house pumping system.

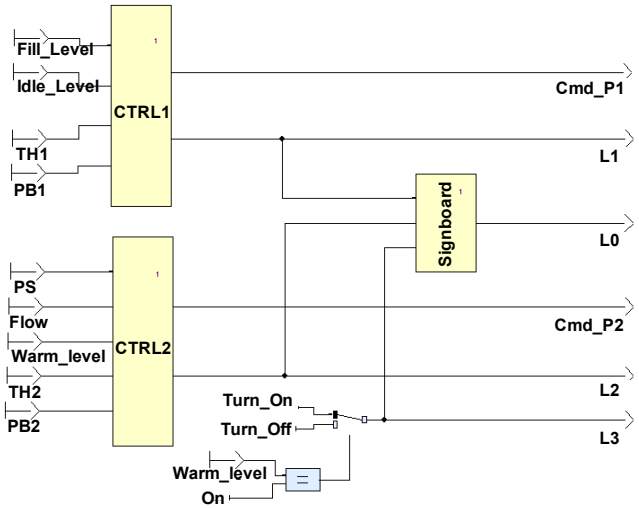

Figure 9. Formal specification of pumping system

The "Tab. I" gives the result of testability analysis process on this case study. It shows that the number of tests (48) is higher than the number of information flows (45, 45 or 27) selected using respectively (1), (2) or (3).

TABLE I. PUMPING SYSTEM ANALYSIS RESULT

Functional requirements	Output variables	Number				Input variables
		Defined tests	Selected flows			
			(1)	(2)	(3)	
R1 R2	Cmd_P1	6	**5**	**5**	**5**	*Fill_level*; *Idle_level*; *TH1*; *PB1*.
R3	Cmd_P1 L1	6	**6**	**6**	**4**	*Fill_level*; *Idle_level*; *TH1*; *PB1*.
R4	Cmd_P2	6	**4**	**4**	**4**	*PS*; *Flow*; *Warn_level*; *TH2*; *PB2*.
R5	Cmd_P2 L3	6	**3**	**3**	**3**	*PS*; *Flow*; *Warn_level*; *TH2*; *PB2*.
R6 R7	Cmd_P2 L2	24	**13**	**13**	**9**	*PS*; *Flow*; *Warn_level* *TH2*; *PB2*.
R8	L0	48	**14**	**14**	**2**	*Fill_level*; *Idle_level*; *Warn_level*; *TH1*; *PB1; TH2; PB2*.

We observe that the number of selected information flows is the same for (1) and (2) in this case study. This is mainly due to the structure of the SCADE model. Indeed, each output variable corresponds to a function description. In addition, the number of selected flows decreases using (3).

The requirement R8 is verified in combination with the seven other requirements. Indeed, L0 represents the state that is checked whatever the pump behavior.

An example of functional tests defined for requirement R1 is described as follows: *"the water-level reaches the filling level (Fill_level = true and Idle_level = false); the pump temperature is normal (TH1 = false and PB1 = false); as result the first pump is activated (Cmd_P1 = true)"*.

The experiment result analysis outlines interesting points:

- For this academic example, we found no orphan information flows; neither missing implemented requirement in the SCADE model. Indeed, this specification is a final version which has been tuned;
- In order to introduce missing test problem, we deliberately omitted to define tests verifying the second pump is running without output rate (R5). As a result, three information flows have not been linked to any test;
- For redundant tests, we identify several tests verifying the second pump behavior when its

temperature is too high (R7). These tests are linked to the same information flows. Consequently, they can be considered redundant because they all address the same pump behavior.

B. Industrial examples

Our testability analysis process has been experimented with two systems LM1 and LM2 provided by AIRBUS. These examples are extracted from an AIRBUS program Auto Flight systems. LM1 and LM2 contain respectively 69 and 146 nodes. The following table "Tab. II" exposes the result of testability analysis process application on these systems.

TABLE II. LM1 AND LM2 TESTABILITY ANALYSIS RESULT

	Number					
	Functional requirements	Output variables	Defined tests	Selected flows (1)	(2)	(3)
LM1	34	58	70	**615**	**198**	**84**
LM2	57	92	114	**534**	**202**	**127**

We observe that the number of selected information flows decreases from (1) to the (3). Indeed, the description of some output variables can share SCADE model parts.

In (1), these common parts are analyzed for each output variable. Thereby, an important number of information flows is determined using this approach.

Regarding (2), common parts using by a set of output variables related to a function are analyzed once during testability analysis. Common parts related to different functions are analyzed several times. So, the number of information flow decreases compared to (1) but is still high.

In (3), all SCADE model related parts are analyzed together. Each common part is explored only once during testability analysis. As a result, information flows determined by this approach represent the lowest number compared to the two others.

Considering these approaches, (3) principle sticks to the AIRBUS testing process of system validation.

The experiment result analysis of LM1 and LM2 outlines the following points:

- No orphan information flow has been detected after running our testability analysis process. No missing flow is identified in the detailed specification.
- Considering (3), for LM1 (resp. LM2), the number of tests (70 (resp. 114)) is lower than the number of flows (84 (resp. 127)). At least, 14 (resp. 13) tests are missing.
- During tests coverage analysis process, we identified that several tests are linked to the same information flows. Consequently, they can be considered redundant.

These experiments show the testability analysis methodology described in Section IV can support efficiently the AIRBUS system detailed specification validation process. Nevertheless, the applicability of the method in terms of scalability must be confirmed on larger operational systems.

VI. CONCLUSION AND FUTURE WORK

This paper deals with the definition of a methodology based on testability principles to support traceability activities and test design in the system validation process. This method consists in identifying links between the detailed specification, the SCADE model, the test data and the information flows. Such an approach makes easier coverage analysis activities and tests design ((L2), (L3) and (L4) in "Fig 6"). The validation phase is thus shortened generating important cost and effort reduction.

In the future, our work will focus on two main objectives. The first one consists in adjusting testability metrics for synchronous data-flow specification [3]. These metrics are: controllability which expresses how easy the input values of an internal component can be controlled through the input values of the system; and observability which expresses how easy the results of an internal component can be observed at the outputs of the system. We mainly plan to take into account the temporal aspects. Until now, our testability approach is based on a static view of the system (one execution cycle). This work will allow analyzing systems on several execution cycles. It will also contribute to reflect, using metric, the verification complexity introduced by these temporal aspects in the specifications.

The second objective of our future work aims at splitting information flows in two categories: the initial flows related to system initialization phase; and the nominal flows. This classification will be useful for test data generation process. It allows the identification of information flows to be exercised in a given phase of system execution during the validation process of system detailed specification.

REFERENCES

[1] H. V. Do, M. Delaunay, C. Robach, "Integrating testability into the development process of reactive systems", *IASTED SE 2007*, Innsbruck, Austria, February 2007.

[2] C. Robach: "Test et testabilité de système informatiques", *PhD Thesis*, 1979.

[3] N.Halbwachs, P. Caspi, P. Raymond and D. Pilaud: "The synchronous dataflow programing language LUSTRE", *Proceedings of the IEEE*, 79(9): 1305-1320, September 1991.

[4] Esterel Technologies SA. *SCADE Technical Manual*, 2005.

[5] R. S. Freedman. Testability of Software Components. *IEEE Transactions on Software Engineering*, 17(6):553–564, Jun 1991.

[6] Y. Le Traon and C. Robach. Testability Measurements for Data Flow Design. *In Proceedings of the Fourth International Software Metrics Symposium*, pages 91–98, Albuquerque, New Mexico, Nov 1997.

[7] T. J. McCabe. A complexity measure. *IEEE Transactions on Software Engineering*, 2(4):308–320, December 1976.

[8] B. A. Nejmeh. Npath: A complexity measure of execution path complexity and its applications. *Communications of the ACM*, 31(2):188–200, February 1988.

[9] C. Robach and P. Wodey. Linking design and test tools: an implementation. *IEEE Transactions on Industrial Electronics*, 36:286–295, 1989.

[10] J. M. Voas and K. W. Miller. Software testability: The new verification. *IEEE Software*, 12(3):17–28, May 1995.

[11] RTCA/DO-178B, "*Software Considerations in Airborne Systems and Equipment Certification*", December 1, 1992.