A STUDY OF THE RELATIONSHIP BETWEEN SYSTEM TESTABILITY AND MODULARITY

Article *in* OR Insight · March 2017

DOI: 10.1002/inst.12140

CITATIONS
3

READS
58

2 authors:

Mahmoud Efatmaneshnik
University of New South Wales - Canberra
75 PUBLICATIONS **270** CITATIONS
SEE PROFILE

Mike Ryan
UNSW Canberra
244 PUBLICATIONS **1,212** CITATIONS
SEE PROFILE

26th Annual INCOSE International Symposium (IS 2016)
Edinburg, Scotland, UK, July 18-21, 2016

A Study of Relationship between of System Testability and Modularity

Mahmoud Efatmaneshnik
School of Engineering and Information Technology
University of New South Wales - Canberra
m.efatmaneshnik@adfa.edu.au

Michael Ryan
School of Engineering and Information Technology
University of New South Wales - Canberra
m.ryan@adfa.edu.au

Abstract. Modularity is one approach to design that, amongst other design attributes, facilitates testability. However, the relationship between the type of modularization and system testability is under-researched. In this paper we present a probabilistic model of testability that enables us to study the relationships between modularity and testability. In particular, we focus on the costs and benefits associated with the particular modularizations selected for the sole purpose of system testability. We show that adopting appropriate testing architecture through careful modularization can greatly enhance efficiency and effectiveness of system testing.

Testability and Test Reliability

Testability is commonly defined as the degree to which a component or system can be tested in isolation (Rodriguez, Llana, and Rabanal 2014, McGregor and Srinivas 1996, Kusiak and Huang 1997, Baudry et al. 2003). (Wang, Wu, and Wen 2006) defines testability of a logic circuit as the relative to effort required to test it. Design for testability techniques improve quality of the product in addition to reducing the costs of testing (Kusiak and Huang 1997, Wang, Wu, and Wen 2006). Testing a system requires interacting with it (Rodriguez, Llana, and Rabanal 2014), naturally testability has been related to other attributes: observability and controllability (Wang, Wu, and Wen 2006). Observability is the degree to which internal state of a system can be inferred from its inputs and outputs relations. Controllability on the other hand is the degree to which the internal state of a system is determined by the inputs. VLSI logic circuit testing literature extracts measures of testability from composites of observability and controllability measures (Kantipudi 2010, Wang, Wu, and Wen 2006).

In this paper we go one step further and study the effects that any level of testability might have on the way on which system modules are tested during assembly. Instead of looking at creating a measure of testability, we study its use. SEVOCAB (SEVOCAB 2015) gives the following entries on testability:

(1) extent to which an objective and feasible test can be designed to determine whether a requirement is met (ISO/IEC 12207:2008 Systems and software engineering--Software life cycle processes, 4.52)

(2) degree to which a requirement is stated in terms that permit establishment of test criteria and performance of tests to determine whether those criteria have been met (IEEE 1233-1998 (R2002) IEEE Guide for Developing System Requirements Specifications, 3.18)

(3) degree to which a system or component facilitates the establishment of test criteria and the performance of tests to determine whether those criteria have been met (ISO/IEC/IEEE 24765:2010 Systems and software engineering--Vocabulary)
(4) (4) degree of effectiveness and efficiency with which test criteria can be established for a system, product, or component and tests can be performed to determine whether those criteria have been met (ISO/IEC 25010:2011 Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE)--System and software quality models, 4.2.7.5)

Testability can be a property of a requirement, a system, or any structural constituent of the system, i.e. subsystem, component, etc. In this paper we define one aspect of testability as the reliability of the test, or the confidence in the outcome of the test, or the probabilistic accuracy of the test in having the correct outcome. From this perspective, testability for a particular system, subsystem, component, interface, or module is controlled by the type of test subject, as well as the object or method through which the test is accomplished. A large computer program might be more difficult to test than a small program. Also a C++ code, for example, might be more testable than a program in machine code, because of the availability of the platform for testing. Testability can therefore also be considered to be related to the ease of testing a system. This ease is both an intrinsic property of the design (thus a proper characteristic of the product) and a property correlated to the testing strategy which is used to reach a chosen test criterion (thus, a joint characteristic of the product and the process).

The test cost, or the time required for testing, is another aspect of testability, which again is determined by the size of the tested system, test reliability as well as topology/architecture of testing. A low-cost test implies high testability—that is, the high probability of test accuracy facilitates quick identification of faults. A third aspect of testability in this paper is represented by the reliability of system after the test relative to that reliability before the test. The test cost and test's achieved reliability in the system are respectively efficiency and effectiveness of the test.

This paper studies the effect of test architecture on test efficiency and effectiveness. The architecture is in turn modeled by way components and interfaces are set up to create subsystems, or modules. In the next section we present a simple model of integration and test. Based on this model a simulation is set up to estimate, in a probabilistic sense, the expected time/cost of tests as well as the expected system reliability after the test. The role of test architecture on these two testability outcomes are discussed in last section.

A Model of System Integration and Testing

In this section we develop a model for system testing and testability attributes. Figure 1 shows the schematic of a system integration and test. Several components, C, are integrated into a system or subsystem or a module, S1. Then a test is performed. This S1 performs as intended then it will passed on to the next level of testing, whatever that might be, for example, integration with another system/subsystem/module. If the system test identifies a fault in one of the components or their integration then the

integration must be repeated in one way or another. The probability that the test identifies a fault is testability (TR). Again this model does not intend to suggest how to improve testability of a particular system by increasing its observability and controllability—we are interested in how the testability relates to the location of tests when assembling modules.

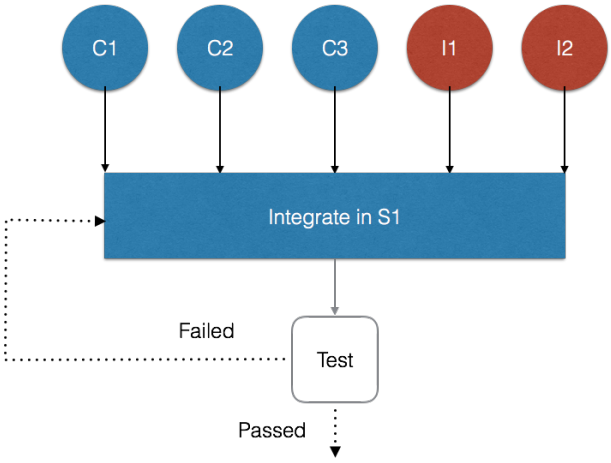

Figure 1. The integration-test procedure. A perfect test identifies any fault in the constituent element and their integration.

Figure 2 shows the interaction of test reliability with component reliability. A component/interface/unit reliability is the probability that it satisfies all its intended requirements. The reliability is denoted by R in the Figure 2. The probability that a failure wont be detected after the test is simply the *latent defect probability* P_{LD} = (1-R)×(1-TR). However this probability is for a hypothetical case when detecting a failure does not require it to be fixed, which defeats the purpose of testing. When the test outcome prompts rework as shown in Figure 1 and in more detail in Figure 3, then the probability that a failure won't be detected is less than or equal to R_T = (1-R)×(1-TR) and the reliability of the system after test (R_T) is always more than R regardless of how bad the test might be (or how small TR is).
Assume an integration process that contains n component/interfaces each with reliability, R_i i=1,...n. The reliability of this system after integration without any testing is:

$$R_S = \prod_{i=1}^{n} R_i \quad (1)$$

And if there is any kind of testing the reliability is

$$R_{ST} \geq \prod_{i=1}^{n} R_i \quad (2)$$

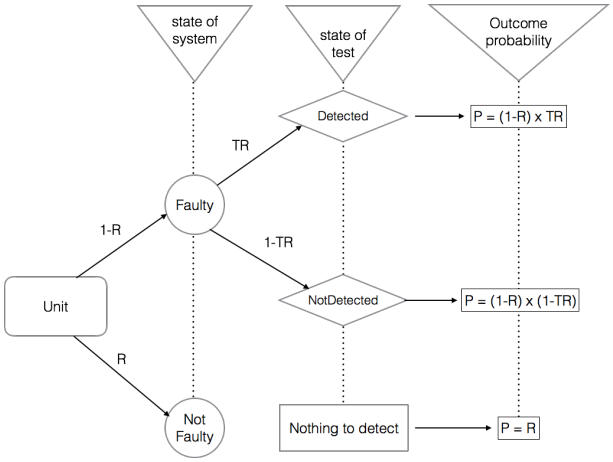

Figure 2. Interaction of testability with component reliability.

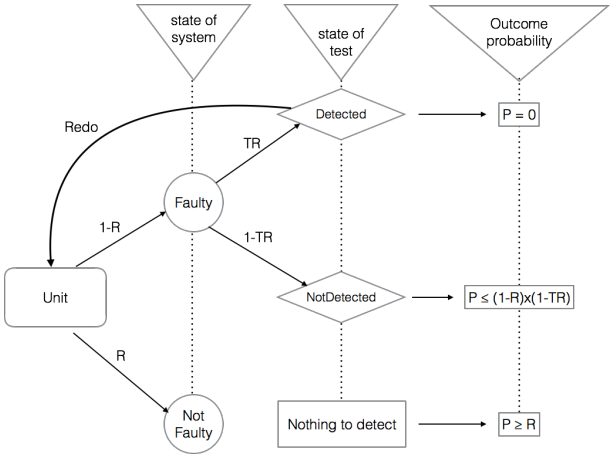

Figure 3. Interaction of test reliability with component reliability with repeated testing.

We have solved R_{ST} and the expected number of tests—that is, an indicator of the total test cost/time, with the Matlab simulation code in Figure 4. In this simulation all the reliabilities of n components/interfaces in S have same reliability R and Figure 5 shows the result of the simulation for R= 0.9. Each time a failure is detected, the test will be repeated after the re-integration. Assuming all tests have the same required time/cost as 1 unit per item per test, the expected time/cost of testing (TST) for n items is Figure 6. Figure 6 is particularly interesting, because it shows that the expected time/cost of testing increases log linearly with test reliability. This is because a higher test reliability means detection of more defects and thus a higher number of re-integration and subsequent tests. Although this leads to fewer latent defects, it increases the cost/time of test. It is evident from these two figures that while the unit reliabilities rapidly increase with improved test reliability (or test reliabilities TR), the expected cost/time of test that is required to achieve R_{ST} (unit reliability), exponentially increases with the size of the system (n). One solution to this problem is to test smaller portions of the system (system modules) first and then perform a global test. In the next section we study the effect of topology of lower level tests on unit reliability and testing cost, given the same test reliability and component reliabilities.

```
k= 1000000;
test_time = 0;
latent = 0;
for i = 1:k
    f = rand(1,n) > R;%a one means a fault
      test_time= test_time + 1;
       while any(f)
         m = nnz(f);%no of actual defects
          d = rand(1,m) < TR; %detected defects
            if any(d)
                 j = find(d);
                    f = rand(1,n) > R;
                 test_time=  test_time + 1;
            else
               f = 0;
               latent = latent + 1;
            end
    end
end
RST = 1 – (latent/k);
TST = n * test_time / k;%for the unit
```

Figure 4. Matlab simulation code to solve for R_{ST} for the simple case when all the reliabilities of n items in S have same reliability R.

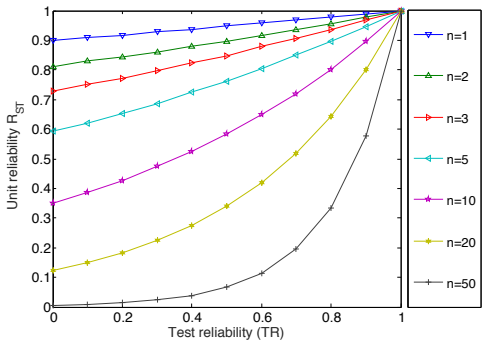

Figure 5. The relationship between unit's reliability after all testing is accomplished (R = 0.9 for all components/interfaces).

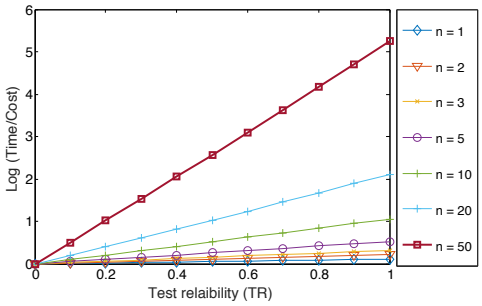

Figure 6. The relationships between test reliability and total expected time/cost of testing, for R = 0.9 for all components/interfaces.

The Architecture of Testing

In this section we study the effect of topology and architecture of testing in a probabilistic sense on the system reliability. Here we consider two level of testing, one (or a number of tests) at the component/module level and one at the system level. Figure 7 demonstrates a two-level testing regime where each component/interface is tested individually followed by integration and a system level test. Figure 8 shows the case where components are grouped into modules for the first level of testing. We are interested to study the effect of different topologies on test cost as well as the system level reliability.

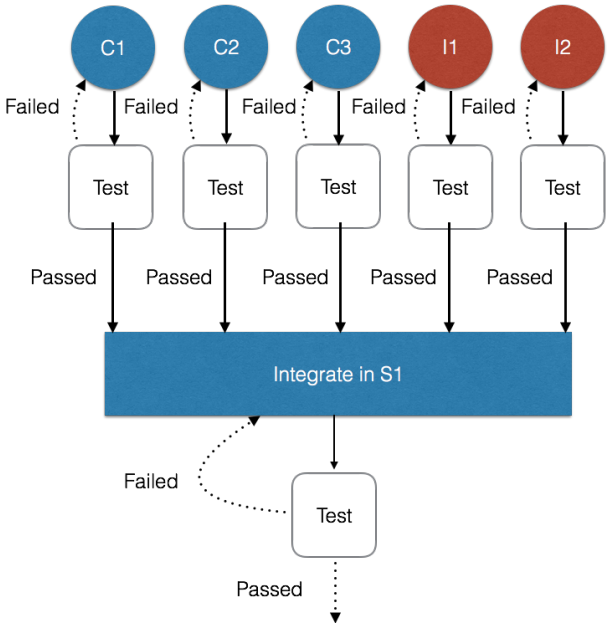

Figure 7. A two level testing architecture with no modularization.

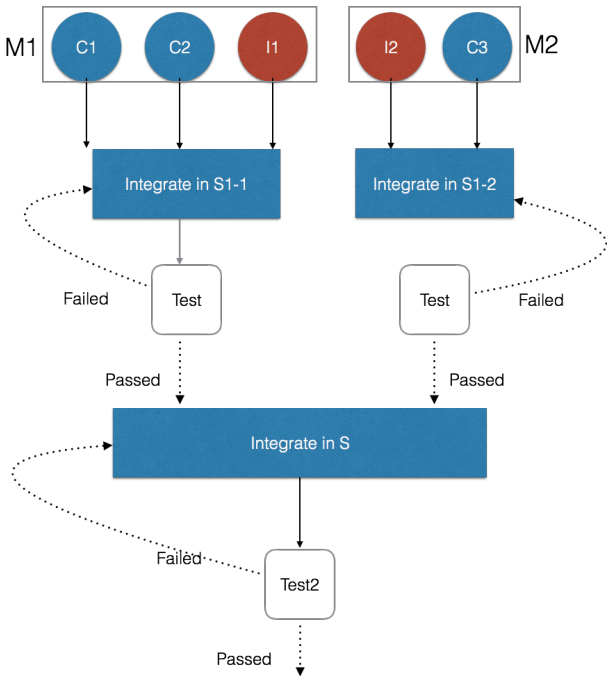

Figure 8. A two-level testing architecture with a two-module decomposition.

In Table 1 we have listed all the possible testing topologies for a hypothetical system of size n =10, which is resulted from all possible modularizations of the system into two levels. Each bracket [•] denotes a test. With the help of the same simulation code in Figure 4 we determined the resulting unit reliabilities and expected testing cost/time that achieves those reliabilities. For this simulation, we have used the unit reliability (R_{ST}) outcome of a low-level test as the component reliability (R) of the next level test. Figure 9 and 10 show the results. The test reliability value has been held constant throughout the testing procedure.

From these two figures we can see that for all two-layer testing architectures (sample no≥2), the achievable unit reliabilities are mostly identical for any given test reliability value. However, as expected the resulted unit reliabilities for two-layer architectures show large improvements relative to that of a single testing architecture (sample no 1). The main finding of this simulation comes from the expected time/costs of the two-layer architectures. Although they vary greatly from architecture to architecture, it is evident that modularization reduces the cost of testing more than 50% relative to non-modular architecture (one-layer testing architecture). Nevertheless the time/cost is much more sensitive to the topology of the testing than it is to the unit reliability. In particular, samples no 2, 7, 15, 24,30, and 35 show a consistent pattern of local cost minimization; thus they are a more efficient architecture for testing; because the same unit reliability results at the lowest cost. Interestingly, these sample numbers have the minimum standard deviation of number of modules for their modularization number. For example when creating 2 modules, a modularization vector of MV = [5,5] has a variance of zero, and for 3 modules MV=[3, 2, 2] has the lowest variance amongst all 3 modules architectures. These are the most balanced modularization in terms of number of components/interfaces inside each module. We have reported a similar heuristic in (Efatmaneshnik and Ryan 2015) for the integration/assembly of components into modules, where we showed low cost of integration for balanced modularization under homogeneous uncertainty of integration failure. The result did not hold when the

uncertainties where heterogeneous (Shoval and Efatmaneshnik 2015), which suggests that the same principle might hold here.

Table 1: 41 modularization vectors (M vector) for the testing of a system of size 10. 'm' denotes the number of modules. Each bracket [n] denotes a test of module of size n.[[a] ,[b]] denotes two tests of system sizes a and b and one test of a system of size two, and so on.

	no	**m**	**no**	**m**	**no**	**m**	**no**	**m**	**no**	**m**
Sample	**1**	1	**2**	2	**3**	2	**4**	2	**5**	2
M vector	[10]		[[5],[5]]		[[6],[4]]		[[7],[3]]		[[8],[2]]	
Sample	**6**	2	**7**	3	**8**	3	**9**	3	**10**	3
M vector	[[9],[1]]		[[4],[3],[3]]		[[4],[4],[2]]		[[5],[3],[2]]		[[5],[4],[1]]	
Sample	**11**	3	**12**	3	**13**	3	**14**	3	**15**	4
M vector	[[6],[2],[2]]		[[6],[3],[1]]		[[7],[2],[1]]		[[8],[1],[1]]		[[3],[3],[2],[2]]	
Sample	**16**	4	**17**	4	**18**	4	**19**	4	**20**	4
M vector	[[3],[3],[3],[1]]		[[4],[2],[2],[2]]		[[4],[3],[2],[1]]		[[4],[4],[1],[1]]		[[5],[3],[1],[1]]	
Sample	**21**	4	**22**	4	**23**	4	**24**	5	**25**	5
M vector	[[5],[2],[2],[1]]		[[6],[2],[1],[1]]		[[7],[1],[1],[1]]		[[2],[2],[2],[2],[2]]		[[3],[2],[2],[2],[1]]	
Sample	**26**	5	**27**	5	**28**	5	**29**	5	**30**	6
M vector	[[3],[3],[2],[1],[1]]		[[4],[3],[1],[1],[1]]		[[5],[2],[1],[1],[1]]		[[6],[1],[1],[1],[1]]		[[2],[2],[2],[2],[1],[1]]	
Sample	**31**	6	**32**	6	**33**	6	**34**	6	**35**	7
M vector	[[3],[2],[2],[1],[1],[1]]		[[3],[3],[1],[1],[1],[1]]		[[4],[2],[1],[1],[1],[1]]		[[5],[1],[1],[1],[1],[1]]		[[2],[2],[2],[1],[1],[1,[1]]	
Sample	**36**	7	**37**	7	**38**	8	**39**	8	**40**	9
M vector	[[3],[2],[1],[1],[1],[1],[1]]		[[4],[1],[1],[1],[1],[1],[1]]		[[2],[2],[1],[1],[1],[1],[1],[1]]		[[3],[1],[1],[1],[1],[1],[1],[1]]		[2],[1],[1],[1],[1],[1],[1],[1],[1]]	
Sample	**41**	10								
M vector	[[1],[1],[1],[1],[1],[1],[1],[1],[1],[1],[1]]									

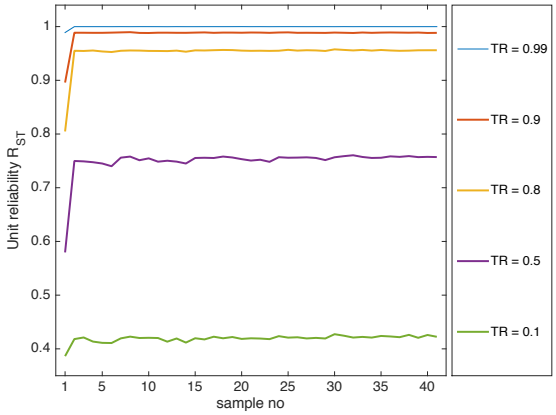

Figure 9. Unit reliability achievable for each sample architecture and for different test reliability values, TR.

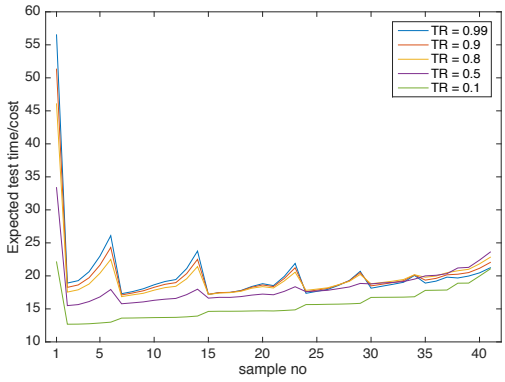

Figure 10. Expected cost/time required for sample architectures and for different test reliability values, TR.

Conclusion

There are a number of reasons for the use of modularity when designing a system—modularity in design, in assembly, in deployment, in operation, in maintenance and in retirement. Amongst other design attributes, we have shown in this paper that modularity also facilitates testability. In this paper we present a probabilistic model of testability that enables us to study the relationships between modularity and testability. In particular, we focus on the costs and benefits associated with the particular modularizations selected for the sole purpose of system testability. We find that, although the expected time/costs vary greatly from architecture to architecture, it is evident that modularization into a two-level architecture reduces the cost of testing more than 50% relative to non-modular architecture (one-layer testing architecture). Additionally, the time/cost is much more sensitive to the topology of the testing than it is to the unit reliability. In particular, the lower cost seems to result for the most balanced modularization in terms of number of components/interfaces inside each module under homogenous component reliability and test reliability conditions.

References

Baudry, Benoit, Yves Le Traon, Gerson Sunyé, and Jean-Marc Jézéquel. 2003. "Measuring and improving design patterns testability." Software Metrics Symposium, 2003. Proceedings. Ninth International.

Efatmaneshnik, Mahmoud, and Michael J Ryan. 2015. "On optimal modularity for system construction." *Complexity*:n/a-n/a. doi: 10.1002/cplx.21646.

Kantipudi, Kalyana R. 2010. "Controllability and Observability."

Kusiak, Andrew, and Chun-Che Huang. 1997. "Design of modular digital circuits for testability." *Components, Packaging, and Manufacturing Technology, Part C, IEEE Transactions on,* 20 (1):48-57.

McGregor, John D, and Satyaprasad Srinivas. 1996. "A measure of testing effort." Proceedings of the 2nd conference on USENIX Conference on Object-Oriented Technologies (COOTS)-Volume 2.

Rodriguez, Ismael, Luis Llana, and Pablo Rabanal. 2014. "A General Testability Theory: Classes, properties, complexity, and testing reductions." *Software Engineering, IEEE Transactions on,* 40 (9):862-894.

SEVOCAB. 2015. "Software and Systems Engineering Vocabulary." pascal.computer.org/.

Shoval, Shraga, and Mahmoud Efatmaneshnik. 2015. "Upper and Lower Bound for Integration Cost of System with Heterogeneous Uncertainties." Asia and Pacific Conference on Systems Engineering (APCOSE), Seuol, S Korea.

Wang, Laung-Terng, Cheng-Wen Wu, and Xiaoqing Wen. 2006. *VLSI test principles and architectures: design for testability*: Academic Press.

Biography

Dr Mahmoud Efatmaneshnik is a researcher at the Capability Systems Research Center, University of New South Wales, Canberra at the Australian Defence Force Academy. He hold BE in aerospace, ME in Manufacturing and PhD in Systems Engineering. Hi areas of interest are complexity management, complexity measurement, modeling and simulation.

Doctor Mike Ryan is a senior lecturer at the University of New South Wales, Canberra at the Australian Defence Force Academy. He holds bachelor, masters and doctor of philosophy degrees in engineering, and his research interests include project management, systems engineering, requirements engineering and military communications and information systems. He is the author or co-author of eleven books, three book chapters, and over 150 technical papers.