The Scientific Journal of University of Benghazi

www.sjuob.uob.edu.ly

Testing Mitigation in Safety Critical Systems: Trade-Off Analysis of Various Testing Criteria

Salwa M. Elakeili*[1], Hoijin Yoon [2], Anneliese Andrews [3]

[1]*Department of Software Engineering, University of Benghazi, Benghazi, Libya.*
[2]*Department of Computer Science, Hyupsung University, <hjyoon@uhs.ac.kr>*
[3]*Department of Computer Science, University of Denver. <andrews@cs.du.edu>*

Received: 3/3/2020; accepted: 9/6/2020.

الملخص

هذه هي الورقة الأولى لمجموعة من الأوراق التي تتناول تكلفة الفشل. في أنظمة السلامة الحرجة (SCSs) تعتبر مقايضه تكلفة الفشل (بسبب نقص الاختبارات) مقابل تكلفة الاختبار أمرًا ضروريًا. عادةً ما تكون معايير الاختبار الأقوى أكثر تكلفة للاختبار ، ولكنها أيضًا أكثر فاعلية في العثور على العيوب التي إذا لم يتم كشفها أثناء الاختبار قد تؤدي إلى فشل تشغيلي يتسبب في تكلفتها. . تقدم هذه الورقة نموذجًا لتكلفة المقايضة التي تقيم معايير مختلفة لاختبار التخفيف من حالات الفشل مقابل تكلفة التخفيف المعيب. يستخدم أسلوب تأثير الفشل وتحليل الحرجية (FMECA) لتحديد تكلفة الفشل ، وقياس العائد على الاستثمار (ROI) لمعايير اختبار التخفيف.

الكلمات المفتاحية:

أنظمة السلامة الحرجة ، اختبارات التخفيف، نموذج خطأ، تأثير وضع الفشل وتحليل الحرجية.

Abstract

This is the first paper of a set of papers dealing with cost of failures. In Safety Critical Systems (SCSs), trade -off the cost of failures (due to lack of testing) against the cost of testing is essential. Usually stronger test criteria are more costly to test, but also more effective at finding faults which if not exposed during testing could lead to operational failures incurring their cost. This paper presents a trade-off cost model that evaluates various criteria for testing mitigation of failures against the cost of defective mitigation. Failure Mode Effect and Criticality Analysis (FMECA) are used to quantify cost of failure, measure the return on investment (ROI) for mitigation test criteria.

Keywords: Safety-Critical Systems (SCSs), Model Based Testing, Failure Mode Effect and Criticality Analysis (FMECA), Fault Model, Mitigation Tests.

1. INTRODUCTION

The trade-off issue with thinking which criterion would be cost-effective is applied in Safety Critical Systems (SCSs). A larger set of tests would be the best, because the loss of failures must be a critical factor in SCSs no matter how high the cost of testing is. SCSs are getting important and popular these days. They are systems whose failure could result in loss of life, damage to the environment or significant property damage [1]. Medical devices, aircraft flight control, weapons and nuclear systems are examples of SCSs. A model explaining how to measure the cost of failures and testing is developed to analyze the trade-off relations. The cost of failures is calculated by analyzing Failure Mode Effect and Criticality Analysis (FMECA) and then applying a standard money value for each severity level of failure. The standard is MUL-STD 882D. The cost of testing is converting to a currency of money per hour. Our measurement is not about how many faults are detected by how many tests, but about how much the costs of failure is saved by how much is the cost of testing? This is because the criteria for SCSs generate fail-safe tests. The safety mitigation test criteria are employed for this analysis. Therefore, some existing works develop how to select test cases for mitigation [2]. Mitigation of SCSs is one of the critical parts of our process. A mitigation fault could harm human's life seriously. So people think that all the possible mitigation tests should be executed for the safety. It is not easy to execute mitigation behaviors, since mitigation behaviors are supposed to start when a specific failure happens. For this, a specific environment where the failure would happen need to be set up before running tests. The time setting up the environment is included in test execution time[3]. It would be expensive if failures happen in SCS dynamically. SCS expects an almost perfect testing, but the cost of executing tests is very high for SCS. For the higher cost effectiveness, criteria select different sets of tests for saving the cost of testing. However, some criteria might miss some of tests that would cover a certain failure. To resolve this issue, we wondered which set of test cases, namely which criterion is appropriate in terms of the cost-effectiveness. An equation to measure the cost-effectiveness is built. Section II explains background and related work. Section III, the safety mitigation test criteria that this paper employs is explained, and then the measurement model using FMECA and MIL-STD 822D is developed. Section IV concludes that there is a certain trade-off relationship between costs of testing and cost of failures in the criteria.

2. BACKGROUND AND RELATED WORK

A. Model Based Testing (MBT)

According to[4] MBT is an approach to generate test cases using a model of the system under test (SUT). The model provides an

*Correspondence:

Salwa M. Elakeili
Department of Software Engineering
University of Benghazi
salwa.elakeili@uob.edu.ly

abstract view of the SUT by focusing on specific aspects. [5] provide a survey on MBT. They define six dimensions of MBT approaches (a taxonomy): model scope, characteristics, paradigm, test selection criteria, test generation technology and test execution. In[5], the authors classify MBT notations as State Based, History Based, Functional, Operational, Stochastic, and Transition based. Transition based notations, which are used in this paper, are graphical node-and-arc notations that focus on defining the transitions between states of the system such as variants of finite state machines (FSMs), extended finite state machines (EFSMs), and communicating extended finite state machines (CEFSMs).

B. Fault Modeling and Analysis

To make systems low risk and fail-safe, software for safety critical systems (SCSs) must deal with the hazards identified by safety analysis. There are over 100 different hazard analysis techniques in existence. The most common analysis methods for SCSs are Preliminary Hazard List Analysis (PHL), Preliminary Hazard Analysis (PHA), Subsystem Hazard Analysis (SSHA), System Hazard Analysis (SHA), Fault Tree Analysis (FTA), Event Tree Analysis (ETA), Failure Mode and Effects Analysis (FMEA), Failure Mode Effects and criticality Analysis (FMECA), Fault Hazard Analysis (FHA), Functional Hazard Analysis (FuHA), Hazard and Operability Analysis (HAZOP), Cause Sequence Analysis, and Common Cause Failure Analysis[6]. These techniques aid in the detection of safety flaws, design errors, and weaknesses of technical systems. FTA is a top-down deductive analysis technique used to detect the specific causes of possible hazards[7], [8]. The top event in a fault tree is the system hazard. FTA works downward from the top event to determine potential causes of a hazard. It uses Boolean logic to represent these combinations of individual faults that can lead to the top event[8]. FMEA is a bottom-up method of analyzing and evaluating safety problems in a system. The FMEA technique consists of identifying and listing all possible failure modes, evaluating effects on the whole system for each failure mode, and identifying all potential causes that may lead to each failure mode[9]. FMECA is composed of two separate analyses, the FMEA and the Criticality Analysis (CA). The CA classifies or prioritizes their level of importance based on failure rate and severity of the effect of failure. Unlike FMEA which is only qualitative, FMECA includes a quantitative evaluation of the criticality of each failure mode. The criticality indices are calculated by multiplying three components- probability of occurrence, severity and detection[10].

C. Integration of Safety Analysis Techniques and Behavior Models.

Several studies have tried to bridge the gap between fault tree and system modeling.

1. **Safety analysis:** Safety analysis improves the probability of uncovering possible faults in safety-critical software. [11] introduce an approach that integrates fault trees and statecharts via a set of transformation steps that maintain semantics of both models. A set of conversion rules that transform gates of fault trees into statechart notation is presented. The integrated model shows how systems behave when a failure occurs. It aids in the identification of system constraints in order to mitigate failures or correct functional and safety specifications. Thus safety analysis is included into the software design process at an early stage. [12] present rules and algorithms to bridge the gap between hazard analysis and system specification by transforming hazards from FTs to a state machine diagram. The algorithms aid the engineer to develop the primary events of the FT by matching them with elements of the state machine diagram, provide transformation rules, and deal with implicit transitions of the state machine diagram. The integrated model focuses on the causes of the hazard and shows direct paths to the causes which help to identify test scenarios. The authors El Ariss et al [11] and kim et al [12] both propose an approach to integrate fault trees and statecharts. They differ in how integration is done. In addition, in[11] the authors consider a FT notation involving time or counters. In[12] each transformed state machine diagram captures both explicit and implicit causes that trigger a hazard with respect to normal behavior. Kaiser and Gramlich [13] provide an approach to integrate behavioral states and events into State/Event Fault Trees (SEFTs). Temporal order of events can be expressed by SEFTs, as are gates with memory (e.g. priority AND). The component concept developed for component fault trees (CFTs) has been further developed for SEFTs: each system may be decomposed into subcomponents. Components are transformed into Deterministic and Stochastic Petri Nets (DSPNs) for quantitative probabilistic analysis.

2. Safety testing: Testing safety-critical software differs from testing non-safety-critical software in many ways. Before testing safety-critical software systems, the needs to conduct a safety analysis for the system to find possible safety breaches, what may cause them and test desired behavior in the presence of failures are essential. S´anchez and Felder [14] proposed a fault-based approach for generating test cases to overcome the limitations of specification-based approaches that derive from the incompleteness of the specification of undesirable behavior, and from the tendency of specifications to focus on the desired behavior, rather than potential faults. Minimum cut sets of the FT are used to determine how undesirable states can occur in a system. These sets are transformed to equivalent statechart components. These components are integrated into the behavioral model of the system and transformed to EFSMs to flatten the hierarchical and concurrent structure of states and to eliminate broadcast communication. The problem is that flattening a statechart into an EFSM model makes it grows exponentially causing scalability problems. Similarly, [15] transform fault tree events into elements of a statechart behavior model. They verify system correctness and criticality using a model checker.

D. Mitigation Modeling

Safety critical systems (SCSs) have requirements that mandate that safety faults have to be identified removed and mitigated. Mitigating failures allows a system to continue operations at a reduced level rather than failing completely. Many mitigations follow a common pattern, like a safety-shutdown, trying alternatives, and omitting functionality that has become dangerous, etc. While there is no work on mitigation models for SCSs, exception handling patterns have been defined for process modeling. [16] identify several, like presenting other alternatives, inserting behavior, skipping some tasks or aborting the current processing. They focus on the composition of the exception handling tasks with normal tasks to identify higher level patterns. Exception handling is a common approach to fault tolerance in software systems. Avizienis et al [17] illustrate taxonomy of error handling and fault handling (fault tolerance) techniques such as rollback, rollforward, and compensation.

3. APPROACH

A. Test Generation Process: In [2], the authors developed an approach to test proper mitigation for failures in SCSs. This approach takes a behavioral model and its testing criteria and creates a set of behavioral tests. It also uses a set of mitigation models and associated coverage criteria to generate a set of mitigation tests. Weaving rules describe how to weave the

mitigation tests into the behavioral test suite at selected points of failure. Four testing criteria were developed in [2] to select combinations of points of failure, type of failure, and specific test paths through the mitigation model. Their cost varies.

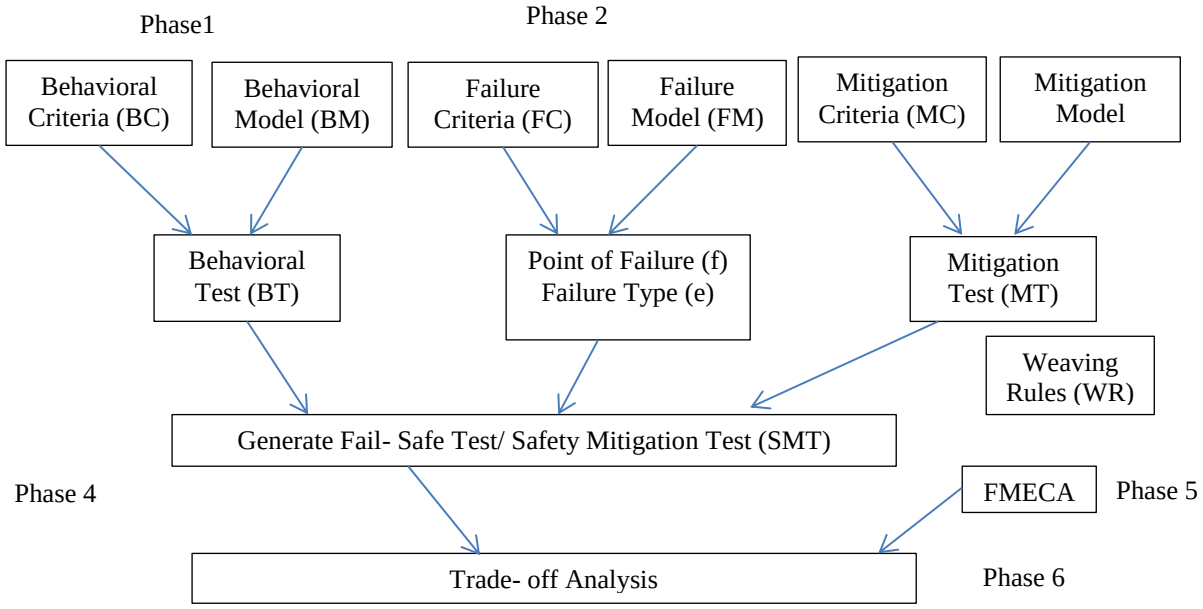


Figure 1: Items for Trade-off Analysis

This paper investigates the trade-off between the cost of testing failure mitigation vs. the cost of an (improperly mitigated) failure occurring. Figure 1 shows the items for trade-off analysis.

- Construct a behavioral test suite (BT) from the behavior model (BM), using behavior test criteria (BC).
- Construct mitigation test suites (MT) from mitigation models (MM), using mitigation criteria (MC).
- Select positions of failure (p) in test suite (BT), and type of failure (e) (failure scenarios). Select (p,e) using failure coverage criteria (FC).
- Construct a safety mitigation test suite (SMT) using the behavioral test suite (BT), point of failure (p), type of failure (e) and mitigation test suite (MT) according to weaving rules (WR).
- Use FMECA to identify and analyze all potential failure modes of the various parts of a system and their effects and consider effects of their severity
- Employ trade-off analysis to demonstrate the relation between cost of testing and cost of failure as well as which criterion is more cost-effective than the others.

Each phase is described in the following subsections.

B. Phase 1: Behavioral Model BM and Behavioral Test BT:

The approach is illustrated using EFSM. EFSMs have been widely used in areas ranging from aircraft, train control, and medical applications [18]. An EFSM is defined as[19]: E=(S, X, Ev, V) where S is a set of states, X is a set of transitions, Ev is a set of events, and V is a store represented by a set of variables. Transitions have a source state source(x) $\in$ S, a target state target(x) $\in$ S and a label lbl(x). Transition labels are of the form $e_1[g]\backslash a$ where $e_1 \in$ Ev, g is a guard, i.e. a condition that guards the transition from being taken when an e_1 is true, and *a* is a sequence of actions. All parts of a label are optional. Test criteria such as edge-coverage, prime-path coverage, etc. [20] can be defined. Using any of a number of test path generation techniques, test paths can then be generated that fulfill these coverage criteria. Let BT = {t_1,…. t_l} be the set of such paths.

C. Phase 2: Determine Points of Failure

Let the set of failures F be defined as {f1, f2, f3,….. fk}. A failure is injected into the system by manipulating parameters that indicate to the software under test (SUT) that a particular failure has occurred (obviously, a failure event like a gas leak is not wanted to occur). This is modeled by inserting a failure injection action directly at the point of failure in the test suite. A point of failure is a particular state in a test path at which the failure is injected. Let $C_{test}(T)$ be the concatenation of test paths in BT that is $C_{test}(T) = t_1 \circ t_2 \circ t_3 \ldots t_l$. Let len (t) be the number of nodes in t. Then I = len ($C_{test}(T)$) = $\sum_{i=0}^{l}$ len (ti) . The position of failure p is a position in the behavioral test suite where a failure is injected. It indicates a point of failure ($1 \leq p \leq$ I). Failure type e ($1 \leq e \leq |E|$) is also selected to apply at the point of failure p. Hence we are selecting (p, e) such that node-failure coverage criteria are met. Note that not all combinations (p, e) are applicable since not all failures are possible or relevant in every node. Therefore, a node-failure applicability matrix A (i,j) is defined as follows.

$$A(i,j) = \begin{cases} 1, if\ failure\ type\ j\ applies\ in\ node\ i\ in\ S \\ 0,\ otherwisw \end{cases}$$

Let *s* = node (p) that is *s* is the behavioral node in the test suite at position p. obviously there has to be at least one state in the behavioral model where a given fault applies. Hence no row in the failure applicability matrix can have all zeros. Hence for a given failure type j there must be some node *s* such that A(s, j) is true. Failure coverage criteria FC for selecting (p, e) need to be defined next. What combinations of test suite positions and failure types (p, e) (failure scenarios)
should one require?

Criteria 1: All combinations, i.e. all positions p, all applicable failure types e (test everything). This is clearly infeasible for all but the smallest models. It would require |I| ×|F| pairs if A contains all"1"s.

Criteria 2: All unique nodes, all applicable failures. This only requires $\sum_{j=1}^{k} \sum_{i=1}^{|s|}(A(i,j) = 1)$ combinations i.e. the number of one entries in the applicability matrix. When some nodes occur many times in a test suite only one needs to be selected by some scheme. This could lead to not testing failure recovery in all tests. A stronger test criterion is to require covering each test as well.

Criteria 3: All tests, all unique nodes, all applicable failures. This criterion requires that when unique nodes need to be covered they are selected from tests that have not been covered. A weaker criterion is not to require covering all applicable failures for each selected position.

Criteria 4: All tests, all unique nodes, some failures (only one failure per position, but covering all failures). Some failure means that collectively all failures must be paired with a position at least once, but not with each selected position as in Criteria 3. The authors in [2] demonstrate an example to explain the four testing criteria.

D. Phase 3: Generate Mitigation Test (MT)

Safety critical systems (SCSs) require mitigation of failures to prevent adverse effects. This can take a variety of actions. Mitigation patterns have been defined in [2]. These mitigation patterns can be expressed in the form of mitigation models. For example, try other alternatives is shown in Figure 2.

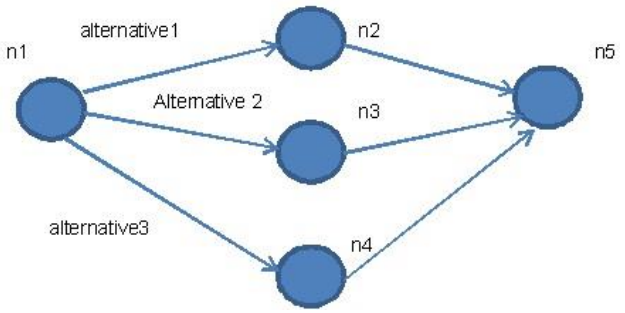


Figure 2: Try Other Alternatives: Mitigation Models

Each failure f_i is associated with a corresponding mitigation model MM_i where i = 1. . . k. The models are of the same type as the behavioral model BM (e.g. an EFSM). Graph based [20], mitigation criteria MC_i can be used to generate mitigation test paths

$MT_i = mt_{i1}, \ldots\ldots, mt_{iki}$ for failure f_i. Figure 2 shows an example of a mitigation model of type "Try other alternatives". Assuming MC as "edge coverage", the following three mitigation test paths fulfill MC: MT= {mt_1, mt_2, mt_3} where mt_1= {n_1, n_2, n_5}, mt_2= {n1, n3, n5}, mt3= {n1, n4, n5}. Mitigation models can be very small for some failures and the mitigation can be an "empty action".

E. Phase 4: Generate Safety Mitigation Tests using Weaving Rules

Assume that $t \in BT$, $p \in I$, $e \in E$ and $mt \in MT_e$. A safety mitigation test $smt \in SMT$ using this information and the weaving rules $wr_e \in WR$ are built as follows:

- Keep path represented by t until failure position p.
- Apply failure of type e (f_e) in p.
- Select appropriate $mt \in MT_e$.
- Apply weaving rule wr_e to construct smt.

Table I shows the equivalent weaving rule for each mitigation pattern.

Let t = {s_1 …. s_b … node (p) … s_f …. s_k}

Table 1: Mitigation Patterns and Weaving Rules

Mitigation Pattern	Weaving Rule Name
Alternative	Fix- option 1
	SMT = s_1……..node(p) mt node(p)……..s_k
Retry	Rollbackward – option 3
	SMT = s_1… node(p) node $(p)^r$ … s_k
Fix and Proceed	Rollbackward – option 1
	SMT = s_1… node (p) mt s_b … s_k
End activity	Rollforward – option 1
	SMT = s_1… node (p) mt s_f … s_k
End All	Fix and Stop – option 2
	SMT = s_1… node (p) mt
Rollback	Rollbackward – option 1
	SMT = = s_1… node (p) mt s_b… s_k
Ignore	No user action required – option 4
	internal compensate
Go to fail-safe	Fix – option 2
	SMT = s_1… node (p) mt s_g

F. Phase 5: Use FMECA

One of the reasons that drive us to use FMECA instead of FMEA is to benefit from the more detailed risk-ranking information from the Criticality Analysis. FMECA is used in this paper to analyze the criticality quantitatively and qualitatively. Quantitative criticality analysis is a series of calculations to rank hardware items and failure modes according to a formula that covers;

- Expected Failures: With an exponential distribution, an expected failure is calculated by multiplying the failure rate (λp) by the time duration of the mission phase or operation time (om), but it is estimated differently for other distributions.

  Expected Failures= (λp) × om
- Mode Ratio of Unreliability: It is the portion of the unreliability to each failure mode. The total of each for failure mode in one item should be 100%.
- Probability of Loss: It is the probability that a failure of the item under analysis will cause a system failure. According to MIL-STD 1692A, it is 100% in actual loss, and it would be between 10% and 100% in probable loss and between 0% and 10% in possible loss.
- Mode Criticality: it is calculated by multiplying λp, t, Mode Ratio and Probability of loss together.
- Item Criticality: it is calculated as the sum of Mode Criticality. To use Qualitative Criticality Analysis to evaluate risk and prioritize corrective actions we use :
- Rate the impact of the potential effects of failure.

Rate the probability of occurrence for each potential failure mode. The impact scale and occurrence scale from MIL-STD 882D are used in this paper. MIL-STD 882D suggests that the occurrence can be ranked from A to E, ("frequent to improbable."). It also identifies that the impact can be categorized from 1 to 4, (catastrophic to negligible.) The impact can be measured in dollars for various risk exposures [21]. As mentioned in[21] the impact cost is divided into three components: the cost caused by property damage, the cost caused by

recovering damage, and the cost from business loss. They are named as I_{Damage}; $I_{Recovery}$; and $I_{Business}$ respectively.
According to MIL-STD 882D, the impact and the occurrence levels is defined as shown in Table II. MIL-STD 882D suggests mishap severity categories as shown in Table III. Mishap severity categories are defined to provide a qualitative measure of the most reasonable credible mishap resulting from personnel error, environmental conditions, design inadequacies, procedural deficiencies, or system, subsystem, or component failure or malfunction. This cost range of each level in the measurement process is used in a domain safety critical system. How much a failure costs with both of the quantitative criticality analysis and the qualitative critical analysis together is measured. Criticality of failure i, in the quantitative view is Cr (i). The cost of failure type i, $C_{failure}$(i) in the qualitative view is a minimum cost, Lo (i), or a maximum cost, Up(i), according to Table III. Finally, to measure the minimum cost caused by failure, I, which is called CFmin(f), multiply Cr (i) by Lo (i). And the maximum cost is measured by multiplying Cr (i) and Up (i). The maximum cost is named as CFmax (f).

Table 2: Levels of Occurrence and Impacts

occurrence	Impact:		
	Damage	Recovery	Business
Frequent:	Catastrophic	Catastrophic	Catastrophic
Probable:	Critical	Critical	Critical
Occasional:	Marginal	Marginal	Marginal
Remote:	Negligible	Negligible	Negligible
Improbable:	Impact= damage + recovery + business		

Table 3: Suggested Mishap Severity Categories

Description	Category	Environmental, safety, and health result criteria
Catastrophic	I	Could result in death, permanent total disability, loss exceeding $1M, or irreversible severe environmental damage that violate law
critical	II	Could result in permanent partial disability, injuries or occupational illness that may result in hospitalization of at least three personal, loss, exceeding $200k but less than $1M or reversible environmental damage causing violate of law
Marginal	III	Could result in injuries or occupational illness resulting in one or more lost work day(s), loss exceeding $10k but less than $200k, or mitigationable environmental damage without violate of law where restoration activities can be accomplished
Negligible	IV	Could result in injuries or illness not resulting in a lost work day, loss exceeding $2k but less than $ 10k, or minimal environmental damage not violating law

G. Phase 6: Trade-Off Analysis

To measure the cost-effectiveness of test criteria, the Return on Investment (ROI) is computed as in[22].

$$ROI = \frac{Benefit - Investment}{Investment} \qquad (1)$$

In Equation 1, Investment is the sum of (a) the cost of testing selected tests $C_{test}(T_{sel})$ and (b) the cost of failures not tested $C_{failure}$ (F_{miss}). Benefit is the sum of (c) the savings of not testing save(T_{miss}) and (d) the savings of avoiding cost of failures $i \in F_{sel}$ ($C_{failure}(F_{sel})$.
Each criterion selects its own set of tests, and the selected tests $C_{test}(T)$ are executed. The cost for testing with the selected tests is a part of investment. Our analysis named it $C_{test}(T_{sel}$. (b) cost of failures not tested $C_{failure}(F_{miss})$. Omitting some tests would cause damage or loss of failure that they would have covered. It can be measured as a part of investment since we need to save the same amount of money for the risk. Our analysis named it $C_{failure}(F_{sel})$. (c) The cost of testing is proportional to the number of tests. A criterion selects a part of all possible tests, and it saves the cost of testing the omitted tests. It is measured as a part of benefit. Our analysis named it save(T_{miss}). (d) The tests selected by a criterion would cover their associated failures through testing activity. The loss or damage that might be caused by the failures would not happen. The saving of this loss of damage is part of the benefit. Our analysis named it $C_{failure}(F_{sel})$. Therefore, the equation (1) is transferred to the equation (2).

Benefit = B = C_{test} (T_{sel}) + $C_{failure}$ (F_{miss})
Investment= I = save (T_{miss}) +$C_{failure}$ (f_{sel})
ROI = $\frac{B-I}{I}$ (2)

The investment pays off if ROI is positive. That means that the higher ROI value is the more effective the criterion is. Four different aggregation criteria from our previous work are employed. The criteria result in four different tests. Their ROI

with equation (2) are measured to show the trade-off relation between the cost of testing and the cost-effectiveness in SCSs.
Equation (2) includes two kinds of functions; $C_{test}(T)$ and COST(T). The function, COST(T), measures how much money it would cost to handle failures that a set of tests, T, covers $C_{failure}(i)$.
In practice, COST(T) has COSTmin(T) and COSTmax(T) since the cost of failures was suggested to have a minimum value and a maximum value as described in Table IV. COSTmin(T) or COSTmax(T) is calculated from CFmin(T,i) or CFmax(T,i), which is the cost of the expected loss or damage through not covering some parts of failure, i, with a set of tests, T. Table 4 describes these functions.
The other function in Equation (2) is Ctest(T), the cost of testing is defined as the time of testing, and then multiply the time by financial unit cost. The time of testing consists of four parts of time according to[23], and it is described in Table 5. From the table, Cost of Testing is calculated as below.

a) $C_{test}(T) = (AT + ST + ET + RT) \times Salary$ (3)

Table 4: Functions to Calculate Cost (T)

|FM|	The number of failure modes
T	A set of test cases
R(T,i)	Ratio of the number of missed pairs to the number for all the pairs of failure i in a set of test cases, T $\frac{the\ number\ of\ missed\ points\ (p,e) of\ failure\ \ f\ in\ T}{the\ number\ of\ possible\ pairs\ (p,e)}$
CMmin(T,i)	Minimum expected cost by missing weaving points, which is measured to R(T,i) CMmin (T,j) = R (T , i) x CF min(i)
CMmax (T,i)	Maximum expected cost by missing weaving points, which is measured to R(T,l) CMmax (T,j) = R (T , i) x CF max(i)
Costmin(T)	Minimum expected cost Costmin(T)= $\sum_{i=1}^{\|FM\|}$ CFmin (T, i)
Costmax(T)	Maximum expected cost Costmax(T) = $\sum_{i=1}^{\|FM\|}$ CFmax (T, i)

However, AT was canceled out with the same reason why CV of S-EVOMO[24] was canceled according to the sensitivity analysis. This is because it could be the same value for each criterion. ST reflects the process selecting tests of the aggregation criteria that we employ. The criteria take the same ST_m and ST_p. That means that these two

values can be omitted in comparing the criteria. Finally, only ST_a makes a difference between them. ST_a is a proportion of the number of tests. If we have the time for aggregating one behavioral test and one mitigation test at a point of (p,e) and name the time as agg, STa is calculated by multiplying the number of tests and agg. ET is surely related to the number of test cases or the total length of test cases. ET is a multiplication of (ET_s+ET_e) and the number of tests. For SCSs, the focus on ET_s is required. SCSs support huge domain systems such as railroads, medical, spacecraft, automobile, and etc. Their scale is bigger than the traditional system. Setting up the environment of testing would take more time and need more effort because of its complexity and scale. That is the point where the trade-off is examined. The last item, RT, is the lease significant according to[24]. RT is omitted in calculating C_{test} (T) of a set of tests, T, as shown in Equation 4.

$C_{test}(T) = (|T| \times agg + |T| \times (ET_s + ET_e)) \times Salary$ (4)

One of the important factors is money/time. We rely on a figure of $100 per person-hour, obtained by adjusting an amount cited in[21] by an appropriate cost of living factor [25]. In Equation (4), money/time is 100/hour. Finally, Equation (2) works with Table III and the equation (4). We measure ROI of a set of test cases selected by each criterion, and check which criterion is more effective in terms of testing costs and cost from damages. The four criteria and equation 2 are applied to show that their ROIs vary.

4. CONCLUSION

For measuring the cost-effectiveness of testing in SCSs, this paper focuses on the following two issues, time and cost. The first conclusion is the measurement needs to consider costs of failures. FMECA and MIL-STD 922D is adopted. According to FMECA, Criticality and Severity of failure were calculated. Based on MIL-STD 922D, a cost of failure for each severity level was calculated. The second is test execution time in SCS which is longer than in traditional systems. The ways in which the cost effectiveness is affected by changing the test execution were analyzed. These two issues brought up two research questions, RQ1 and RQ2. To address them, this paper measured cost effectiveness of four test criteria that would select tests covering both behavioral models and mitigation models. These safety mitigation test criteria were employed and the detail of their methods was explained in Section III. The cost effectiveness is measured as ROI, which was described in Equation (2).

Table 5: Elements of Cost of Testing

AT	Analysis Time: Time of identifying failure modes + cost of building mitigation models for failure modes : $AT= AT_f + AT_m$ AT_f : time of identifying failure modes and setting up values for each failure mode through FMECA AT_m : cost of building mitigation models for failure modes
ST	Test Selection Time :Time of working out the test input, and identifying the correct output or system behavior :largely depending on the chosen test strategy :ST=STm+ STp+ STa STm= time of generation mitigation tests from mitigation models STp= time of identifying (p,e) STa= time of aggregating mitigation tests and behavioral tests
ET	Time to execute test cases : The ways to set up testing environment can be quite different depending on which application we use. Our domain is safety critical software ET = (ETs + ETe) X |T| ETs: time of setting up the environment for one test (loading, compiling, entering data…) ETe: time of executing a test under the environment |T| : The number of tests
RT	Result Analysis Time : RT = RTr + RTo+ RTc RTr: tester's time in collecting test outputs RTo: Time in understanding the correct output RTc: Testers time to compare the test output to specification

5. REFERENCES

1. Knight J. Safety Critical Systems: Challenges and Directions, In Proceedings of the 24th International Conference on Software Engineering (ICSE '02). ACM, New York, NY, USA, 2002: 547-550. DOI=http://dx.doi.org/10.1145/581339.581406.
2. Elakeili S, Andrews A, Boukhris S. Fail Safe Test Generation in Safety Critical Systems. 15th IEEE International Symposium on High Assurance Systems Engineering. 2014; vol. (0): 49-56. DOI: 10.1109/HASE.2014.16
3. Hyunsook D, Gregg R. Using sensitivity analysis to create simplified economic models for regression testing. In Proceedings of the International Symposium on Software Testing and Analysis. ACM. 2008: 51–62.
4. Nguyen C, Marchetto A, Tonella P. Combining Model-Based and Combinatorial Testing for Effective Test Case Generation. In Proceedings of the 2012 International Symposium on Software Testing and Analysis. ISSTA 2012. New York, NY, USA: ACM. 2012: 100–110. [Online]. http://doi.acm.org/10.1145/04000800.2336765
5. Utting M, Pretschner A, Legeard B. A taxonomy of Model-Based Testing Approaches. Softw. Test. Verif. Reliab. 2012; vol. 22 (5): 297– 312. [Online]. Available: http://dx.doi.org/10.1002/stvr.456
6. Ericson C. Hazard Analysis Techniques for System Safety. John Wiley sons, Inc. 2005
7. Tribble A, Miller S. Software Intensive Systems Safety Analysis. Aerospace and Electronic Systems Magazine, IEEE; 2004; vol. 19 (10): 21 – 26.
8. Leaphart E, Czerny B, D'Ambrosio J, Denlinger C. Survey of Software Failsafe Techniques for Safety-Critical Automotive Applications. SAE Technical Paper 2005-01-0779, 2005, https://doi.org/10.4271/2005-01-0779. Paper 2005-01-0779, 2005, https://doi.org/10.4271/2005-01-0779.paper.
9. Wang D, Pan J. An optimization to Automatic Fault Tree Analysis and Failure Mode and Effect Analysis Approaches for Processes. In International Conference on Computer Design and Applications (ICCDA), 2010; vol. 3: V3–153 –V3–157.

10. Bonnabry P, Despont-Gros C, Grauser D, Casez P, Despond D, Pugin C et al.. A risk analysis method to evaluate the impact of a computerized provider order entry system on patient safety. Journal of the American Medical Informatics Association, 2008; vol, 15 (4): 453–460.
11. El Ariss O, Xu D, Wong W. Integrating Safety Analysis with Functional Modeling. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, 2011; vol. 41, (4): 610 –624.
12. Kim H, Wong W, Debroy V, Bae D. Bridging the Gap between Fault Trees and UML State Machine Diagrams for Safety Analysis. In 17th Asia Pacific Software Engineering Conference (APSEC); 2010:196 –205.
13. Kaiser B. and Gramlich C. State-Event-Fault-Trees A Safety Analysis Model for Software Controlled Systems. Reliability Engineering & System Safety. 2007; vol. 92 (11): 1521 - 1537.
14. S´anchez M. and Felder M. A Systematic Approach to Generate Test Cases based on Faults. In Argentine Symposium in Software Engineering, Buenos Aires, Argentina. 2003
15. Nazier R, Bauer T. Automated Risk-Based Testing by Integrating Safety Analysis Information into System Behavior Models. In IEEE 23rd International Symposium on Software Reliability Engineering Workshops (ISSREW). 2012: 213–218.
16. Lerner S, Christov S, Osterweil J, Bendraou U, Kannengiesser U, Wise A. Exception Handling Patterns for Process Modeling. IEEE Transactions on Software Engineering, 2010; vol. 36 (2): 162–183.
17. Avizienis A, Laprie JCB, Randell B, and Landwehr C, Basic Concepts and Taxonomy of Dependable and Secure Computing. IEEE Transactions on Dependable and Secure Computing; 2004, vol. 1 (1): 11–33.
18. Derderian K, Hierons R, Harman M, Guo Q, Estimating the Feasibility of Transition Paths in Extended Finite State Machines. Automated Software Engineering, vol. 17 (1); 2010: 33–56.
19. Androutsopoulos K, Clark D, Harman M, Li Z, Tratt L. Control Dependence for Extended Finite State Machines. In Fundamental Approaches to Software Engineering. Springer. 2009: 216–230.
20. Ammann P, Offutt J. Introduction to Software Testing. 1st ed. 32 Avenues of the Americas, New York, NY 10013, USA: Cambridge University Press. 2008
21. Jones C. Applied Software Measurement: Assuring Productivity and Quality. New York, NY, USA: McGraw-Hill, Inc. 2003
22. Müller M, Padberg F. About the Return on Investment of Test-Driven Development. In EDSER-5. 5th International Workshop on Economic-Driven Software Engineering Research. 2003: 26.
23. Leung H. and White L. A Cost Model to Compare Regression Test Strategies. In Software Maintenance. Proceedings. Conference on IEEE. 1991: 201–208.
24. Do H, Rothermel G. Using Sensitivity Analysis to Create Simplified Economic Models for Regression Testing. In Proceedings of the 2008 International Symposium on Software Testing and Analysis. ACM. 2008: 51–62.
25. Medikonda B, Ramaiah P, Gokhale A. FMEA and Fault Tree Based Software Safety Analysis of a Railroad Crossing Critical System. Global Journal of Computer Science and Technology. 2011; vol. 11 (8): 57–58.