ESTIMATING OF CRITICAL SOFTWARE LATENT FAULTS PRESENCE WITH REQUIRED TRUSTWORTHINESS

Boris Konorev[1], Vladimir Sergiienko[1,2], Vyacheslav Kharchenko[1], Grygoriy Zholtkevych[3]

[1] N.E. Zhukovsky National Aerospace University "KhAI", Kharkiv, Ukraine
[2] I&C System Certification Center, Kharkiv, Ukraine
[3] V.N. Karazin Kharkiv National University, Kharkiv, Ukraine

e-mail: bkonorev@gmail.com, vov3d@mail.ru,
V.Kharchenko@khai.edu g.zholtkevych@gmail.com

ABSTRACT

The estimation of latent faults probability is a key indicator for quantitative assessment of critical systems reliability and safety. The article presents the method of latent faults probability estimating for critical software with a required (controlled) trustworthiness. Compositional set-theoretical model of residual and latent faults is proposed for framework assessment of latent faults. The procedure of the experimental calibration of faults sensitivity and diversity degree of inspection methods of source software faultlessness is presented. The "dotted" injections method using faults profile for the level of programming languages is proposed.

1. INTRODUCTION

Possible damage of the software fault propagation on the system level during operation of I&C systems critical to safety can be in the range "material losses - damage to the environment - a threat to the health and life of human being". Latent faults (which are not detected during testing) in critical software are considerable risk factors, as they may lead to failure of I&C systems during performing functions critical to safety. Therefore, the estimation of latent faults probability - one of the most important tasks for qualification testing and risk management of critical to safety systems maintenance.

At the moment, for the tests of reliability and functionality of critical software the model-checking approach is used more widely [Peled (2009), Baier (2008)]. Model checking can be implemented during static analysis [Moiseev (2013)]. Multi-model model-checking approach may be used to confirm the absence of latent faults [Konorev (2009), Konorev (2011), Konorev (2012)]. This approach consists in critical software source code validation by use of the invariants - oriented models. The invariant is an invariable attribute of software. The invariants - oriented models are generating at static software source code analysis to verify the correctness of each software invariant [Brukhankov (2010)]. Each such model of verifiable software project contains the necessary data to monitor a specific invariant. After that invariants measurement methods (using invariants - oriented models as input data) implement an algorithm of inspection of specific invariants values. Each invariants measurement method either confirms invariant consistency (unchangeability), or detects abnormality. Evidence of the approach is based on the usage of diversity principle - various (diverse) invariants measurement methods are used for latent faults detection. In general, the methods have different sensitivity to faults (diversity degree - the degree of differences between them). To evaluate the fault sensitivity and diversity degree of methods the

procedure of artificial faults injection can be used [Cotroneo (2013)]. Faults can be injected both to the executable code, and at the level of the source codes [Lanzaro (2013)].

In the article the approach of latent faults probability estimation is proposed, which is based on an assessment of faults sensitivity and diversity degree of invariants measurement methods.

The basis for probability estimation of latent faults is a set-theoretical model of latent and residual software faults, the model allows getting a framework assessment of latent faults probability. The procedure of experimental calibration of sensitivity and diversity degree of invariants measurement methods is used for model parameterization, the procedure is based on the use of test faults "dotted" injection.

The result of the experimental calibration of invariants measurement methods is the definition of the partial sensitivity (as the probability of latent fault detection) of the j-th method to the i-th faults type from injectable test faults subset, expressed as a set of residual faults of the i-th type for the j-th method .

As a result of processing of data, collected during the calibration, diversity degree of invariants measurement methods can be determined, an assessment of source code test coverage completeness is performed, residual faults area and the border area of latent faults location are established.

2. MODEL OF RESIDUAL AND LATENT SOFTWARE FAULTS

To assess the effect of using different invariants measurement methods the set-theoretical model of the residual and latent software faults for the composition of invariants measurement methods is proposed (see Fig. 1). Latent fault is software code or documentation incorrectness which was not found during development and testing and could lead to one or more failures of a system component or an entire system. Efforts should be focused exactly on the removal or determining the location of this faults type during qualification testing, that includes software independent verification. Estimation of latent faults probability is encouraged to obtain indirectly from the residual faults data. Residual faults for invariants measurement methods are faults, which are not detected by these methods. Location area of residual faults can be set experimentally by procedure of invariants measurement methods calibration.

The model, presented by Venn diagram (Fig. 1), illustrates a possible positional relationship of residual and latent faults sets and allows estimating the advantages due to the use of invariants measurement methods for a variety of positional relationship (superposition) of residual faults sets.

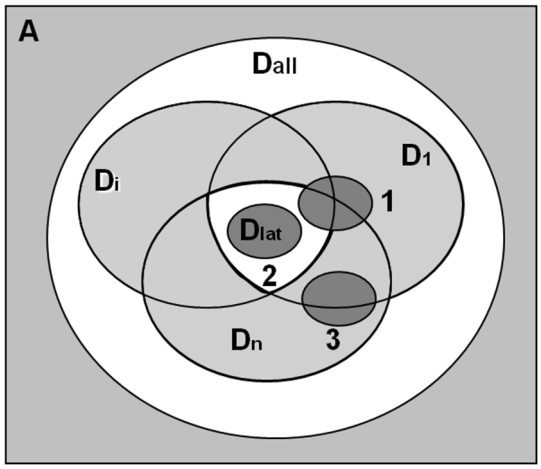

Figure 1. Model of latent faults for invariants measurement diverse methods, where D_{all} – the set of possible faults in the address space of the software A; D_i - the set of faults, which are not detected by measurement method of the i-type invariants; D_{lat} - the set of latent faults.

Assessment indicator of achievable effect for invariants measuring methods composition implementation at independent verification is the quantity (%), on which the boundary value of latent faults probability P_{lat} decreases.

$$I = \frac{\left| D_{lat} \setminus \bigcap_{i=1}^{n} D_i \right|}{\left| D_{lat} \right|} * P_{lat}, \quad (1)$$

where n – number of used methods.

The intersection of residual faults subsets is a frame assessment of the latent fault probability and can be used as estimation result.

The main benefit (advantage) of using invariants measurement methods is determined by the diversity degree of invariants measurement methods in terms of a specific project. It is possible to define 3 alternatives:

a) theoretically possible case of latent faults probability decrease to 100% (I=1, position 3 in Fig.1):

$$\left(\bigcap_{i=1}^{n} D_i \right) \cap D_{lat} = \varnothing, \quad (2)$$

b) ultimate unfavorable case (I=0, position 2 in Fig.1):

$$D_{lat} \subset \bigcap_{i=1}^{n} D_i, \quad (3)$$

c) common case(position 1 in Fig.1)

$$\left(\bigcap_{i=1}^{n} D_i \right) \cap D_{lat} \neq \varnothing, \quad (4)$$

If latent faults (elements of the set D_{lat}) are not detected during independent verification, though the indicator of achievable effect I=0, subset $\bigcap_{i=1}^{n} D_i$ can be used as boundary area of latent fault location (it is frame assessment of latent fault probability). The area (set) is established, where faults absence is guaranteed - $\overline{\bigcap_{i=1}^{n} D_i}$. The efforts should be focused on the area $\bigcap_{i=1}^{n} D_i$ analysis (searching the faults inside this area) to improve or refine the assessment. The relative benefit of using measurement invariants methods V_{rel} is defined as:

$$V_{rel} = 1 - \frac{\left| \bigcap_{i=1}^{n} D_i \right|}{\left| \bigcup_{i=1}^{n} D_i \right|}. \quad (5)$$

Thus, the proposed set-theoretical model allows defining the area of latent faults and to establish an absolute and relative gains during usage of diverse invariants measurement methods.

3. PROCEDURE OF EXPERIMENTAL CALIBRATION

In order to estimate the probability of latent faults presence for specific software project, experimental data on the fault sensitivity and the diversity degree of invariants measurement methods must be obtained in accordance with the proposed model of residual and latent faults (shown in Fig. 1). Parameterization of the model is getting through the procedure of experimental calibration of invariants measurement methods by the method of test faults "dotted" injection. The method is the "dotted" (single) injection of a specific type test fault in the selected location of software, sensitivity binary assessment of invariant measurement methods (detection/undetection of injected fault) and returning to its initial state (removal of the test fault). "Dotted" injection means a single fault injection, this guarantees the absence of mutual influence and the emergence of a secondary faults due to interference and mutation of injected faults.

To assess the completeness of test coverage and methods sensitivity during each step (layer) of calibration the following actions must be performed:

Generating of test faults profile with the use of results of syntactic and semantic parsing in the mode of software source code static analysis;

Consecutive choice of fault type and injection of a single fault of selected type into checked software;

Checking by all invariants measurement methods;

Fixation (registration) of detection/undetection of injected test faults for each method;

Backout to the software initial state (the state before fault injection) and repeat the procedure from step 2.

The procedure is performed cyclically for all possible fault types for a specific project.

Quantity of injected faults (number of steps) for each fault type is determined on the assumption of required accuracy (reliability, uncertainty) results.

Graphic representation of the calibration model is shown in Figure 2.

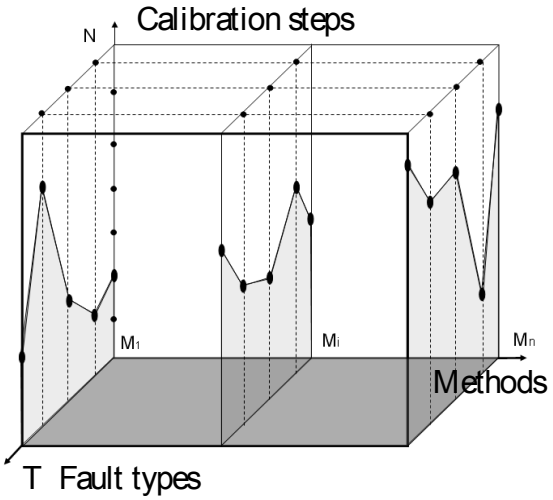

Figure 2. Calibration model.

Experimental calibration of fault sensitivity of each method and integral sensitivity of diverse methods composition is produced in the context of calibration tests space (Fig. 2), which represents Cartesian product:

$$P_{cal} \subseteq N \times M \times T, \quad (6)$$

where $N=\{n_i\}$ – set of test faults injections that are implemented during the calibration;

$M=\{m_j\}$ – set of calibrated invariants measurement methods;

$T=\{t_k\}$ – set of injected test faults types or test faults profile (TFP).

TFP for a specific project is determined by the used language constructs composition. Each construction can be distorted by certain fault types, which injection will not lead to violation of the syntax and semantics of the program (i.e., the fault injection will not cause an error during compilation). These types of faults are forming test faults profile of a specific project.

P_{cal} defines the set (tuple) of the outcomes - the results of experiments during calibration. P_{cal} item value with the coordinate $(n_i; m_j; t_k)$ possesses the value 0 if the test fault was not found, or 1 if the test fault was detected.

Injection of a tuple $T=\{t_k\}$ during method $M = \{m_j\}$ calibration is a step or layer of calibration in the P_{cal} space.

Each calibration step (increment along the axis N) can be represented as a "secant plane": the injection of faults is performed from the test faults profile and verification is fulfilled by all methods.

At each step of "secant plane" the following elements may be defined:

Disjunction of undetected test faults for each M_i

$$D_{fault\_i} = \bigcup_{j=1}^{m} D_j , \tag{7}$$

Conjunction of undetected faults for each fault profile type for all M_i.

$$D_{t\_k} = \bigcap_{i=1}^{n} D_i , \tag{8}$$

Partial sensitivity $P_{part\ ji}$ of each (the j-th) method to the i-th fault type as a set of detected faults by the j-th method (sensitivity to the concrete faults type).

As a result of calibration the following characteristics are determined:

–Integral sensitivity of each method for chosen test faults profile TFP:

$$P_j = \bigcup_{i=1}^{k} P_{part_{ji}} , \tag{9}$$

where k – total number of fault types;

–In pairs for all M_i (for all pairs) absolute and relative diversity degree :

$$m_{ijabs} = \left|P_i \Delta P_j\right| = \left|P_i \cup P_j\right| \setminus \left|P_i \cap P_j\right| , \tag{10}$$

$$m_{ij} = 1 - \frac{\left|P_i \cap P_j\right|}{\left|P_i \cup P_j\right|} , \tag{11}$$

where i and j take the values from 1 to n; n - number of used methods.

Using specified data the list of invariants measurement methods can be defined, this set ensures achievement of set point of test coverage completeness and validity (with a specified trustworthiness) of latent faults probability estimation.

4. FAULTS "DOTTED" INJECTION METHOD

The proposed method of faults "dotted" injection recommends the places (answers the questions) in which the test faults of TFP should be injected and the way how the injection procedure should be implemented.

Detailed algorithm of action sequences "fault injection - analysis of fault impact on invariants consistency - returning of software to its initial state" is shown in Figure 3.

1. Fault type is selected alternately from TFP.
2. The injection places (constructions) for each fault type are established, – only those places are chosen from full spectrum of software project operations, where injection of the fault does not lead to a violation of syntax and semantics program (i.e., the injection of the fault will not lead to an error during compilation). In other words, the faults in the software, which are detected during compilation, are not taken into account during the choice of faults for injection.
3. All possible injection places form a list of points for injection, - each established place gets its identification ordinal number.
4. If the number of possible injection places is very large (for a real project it can be tens of thousands), the number of injections n is determined by required reliability of the results.
5. The injection place is chosen randomly from the general list of points and injection is implemented (replacement of the correct construction with incorrect). The point, which has already been exposed to distortion, is excluded from the list (to prevent re-injection of the fault in the same place).
6. The modified code analysis (checking) is implemented by all invariants measurement methods which were determined for a specific project.
7. Check results are fixed in binary form (a fault is detected / not detected).
8. Verifiable code returns to a state, when the fault was not injected.
9. Steps 5-8 are executed number of times established on the 4th step.
10. If TFP is not covered, the procedure must be implemented from step 1 (the procedure should be performed for all fault types from TFP).

5. PROCESSING OF CALIBRATION RESULTS

Processing of collected during the calibration data allows calculating a number of metrics.

Numerical expression of partial sensitivity of method to each faults type is determined as:

$$S_{part\,ij} = \frac{n_{det\,ij}}{n_i} * 100\%, \qquad (12)$$

where n_i - is general number of measurements of the i-th faults type; $n_{det\ ij}$ - number of measurements, in which the j-th method detected the i-th faults type.

Note: it is possible that two methods have equal partial sensitivity to some fault type, but at the same time they found faults in different injection places. In this case, to increase the degree of test coverage it is necessary to use both methods.

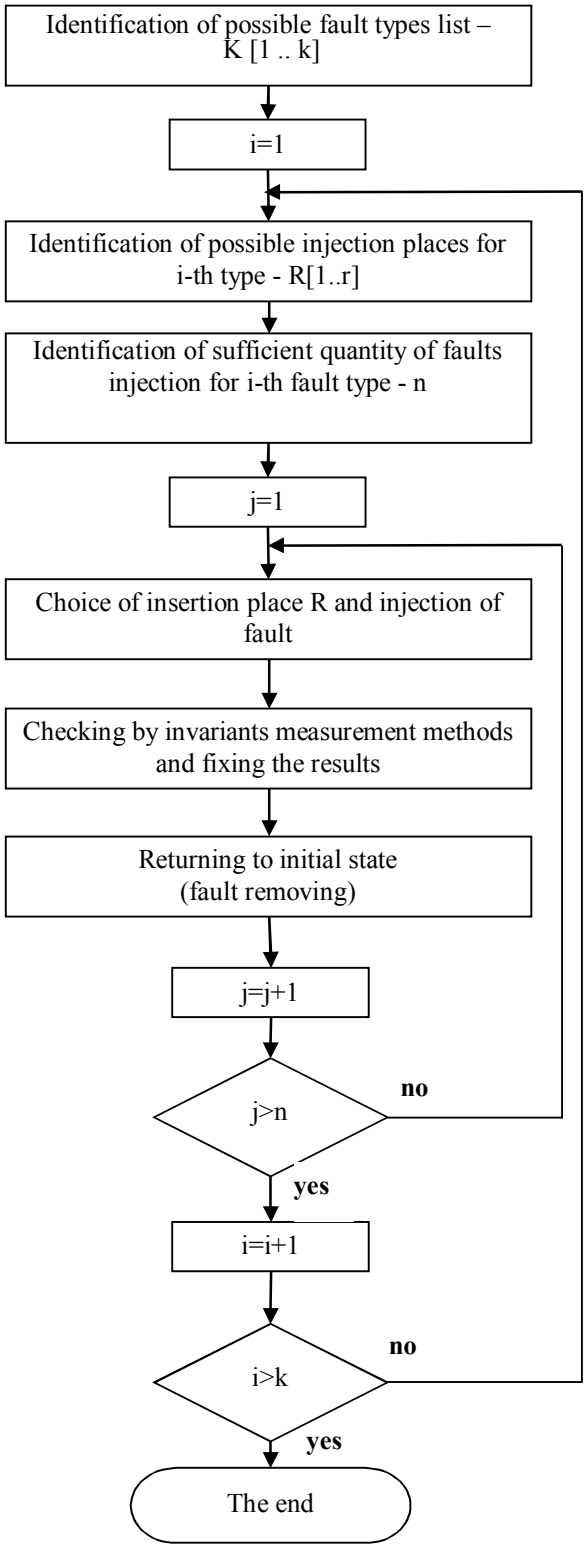

Figure 3. Test faults “drop” (dotted) injection algorithm.

The minimum required number of measurements n is calculated taking into account the established requirements of result reliability. At the same time, it is guaranteed that the discrepancy between true and calculated medium measured value does not exceed legitimate value.

Percentage of detected faults for each type $S_{det\,i}$:

$$S_{det\,i} = \frac{n_{det\,i}}{n_i} * 100\%, \quad (13)$$

where $n_{det\,i}$ – number of measurements, in which at least one method detected i-th type of fault.

Percentage of detected faults for each method S_j:

$$S_j = \frac{\sum_{i=1}^{k} n_{det\,ij}}{\sum_{i=1}^{k} n_{det\,i}} * 100\%, \quad (14)$$

where k – number of fault types.

As a result, the probability of latent faults existence P_{lat} is defined as:

$$P_{lat} = \frac{\sum_{i=1}^{k} (S_{det\,i} * r_i / 100)}{\sum_{i=1}^{k} r_i} * 100\%, \quad (15)$$

where r_i – number of language structures in code, that can contain the i-th type of fault.

6. CASE STUDY

Testing of the proposed technique was performed in the framework of the Science and Technology Center in Ukraine project # 4726 «Safety-critical software independent verification and latent faults assessment based on diverse measurement of invariants». Model-checking verification has been implemented in static software source code analysis on IDE Eclipse platform. Designed Mobile instrumental complex enabled to check a number of invariants. These invariants were the following:

–Semantic invariants of software variables (physical dimension, variation interval, the accuracy of the representation);

–Control flow invariants: reducibility of control thread, the potential accessibility and "liveliness" (demand) of operators;

–RAM usage in a particular software project: memory leaks, repeated memory release;

–The structure of the threads of program control (the execution logic);

–Specific invariants for FPGA-based I&C systems (“list of sensitivity“, “signal race", absence of "latches", etc) [Kharchenko (2012)].

7. CONCLUSIONS

The proposed technique allows evaluating latent faults probability with controlled (required) trustworthiness during independent verification of critical software. Trustworthiness is the controlled value during application of experimental calibration. It is determined by the test faults profile completeness and the accepted amount of injected faults.

The composite set-theoretical model of residual and latent faults is provided. This model allows evaluating of residual faults, to assess the area of latent faults location and test coverage completeness.

"Dotted" injection method of test faults is presented with the use of faults profile, which takes into account the specifics of a project at the level of the programming language.

The proposed procedure of experimental calibration of faults sensitivity and diversity degree of invariants measurement methods is based on the faults "dotted" injection method, it allows eliminating the possibility of interference (superposition) and mutation of faults in address field of software.

The proposed method is developed in framework of the invariant-oriented model-checking approach for critical software verification. Its application can be extended to other domains.

REFERENCES

1. Peled D., Pelliccione P., Spoletini P. (2009). Model Checking. *Wiley Encyclopedia of Computer Science and Engineering*. pp. 1904–1920.
2. Baier C., Katoen J.-P. (2008). *Principles of Model Checking*. The MIT Press. 984 p.
3. Moiseev M. (2013). Static Analysis Approach for Defect Detection in Multithreaded C/C++ Programs. In: *Software Engineering for Resilient Systems (SERENE 2013)* Springer LNCS, Vol. 8166. pp. 169-183.
4. Konorev B., Kharchenko V. (Eds) (2009). *Invariant-oriented assessment of space systems software quality*. Kharkiv: National Aerospace University KhAI. - 221 p. [in Russian]
5. Konorev B., Sergiyenko V., Chertkov G. (2011). The Evidential Independent Verification of Software of Information and Control Systems, Critical to Safety: Functional Model of Scenario. In. *IEEE East-West Design & Test Symposium (EWDTS'2011)* Kharkiv: KNURE, Ukraine. pp. 263-266.
6. Konorev B., Sergienko V., Zholtkevych G. , Chertkov G., Alexeev Y. (2012). Concept of Critical Software Independent Verification Based on Invariant-oriented Model-checking Approach *Radio-electronic and Computer Systems*. – Vol. 57, № 5. – pp. 184 – 190.
7. Brukhankov S., Konorev B., L'vov M., Zholtkevych G. (2010). About Static Analysis of Variables Physical Dimensions for Critical-mission Software. *Radio-electronic and Computer Systems*. – Vol. 47, # 6. – pp. 186 – 191.
8. Cotroneo D. & Madeira H. (2013). Introduction to Software Fault Injection. In: *Innovative Technologies for Dependable OTS-Based Critical Systems. Challenges and Achievements of the CRITICAL STEP Project*. Milan: Springer. pp. 1-15.
9. Lanzaro A., Natella R., Barbosa R. (2013). Tools for Injecting Software Faults at the Binary and Source-Code Level. In: *Innovative Technologies for Dependable OTS-Based Critical Systems. Challenges and Achievements of the CRITICAL STEP Project*. Milan: Springer. pp. 85-100.
10. Kharchenko V. (Ed.) (2012). *Case-assessment of critical software systems*, Volumes 1-3 / Volume 3. Safety. Kharkiv: National Aerospace University KhAI. – 302 p. [in Russian]