ONE CRITERION FOR SOFTWARE SAFETY

Assoc. Prof. Assen Vassilev Krumov, Ph.D.,
VVTU"T.Kableshkov", Sofia

Abstract: In the paper one criterion in two versions for software safety is suggested. The first version can be applied on line during the functioning of the computer system for detection of dangerous failures and transition to a safe state of the controlled object. The other version of the criterion can be used off line during the design of the software. This version is connected with the problem of practical stability of the software and a theorem concerning the practical stability is suggested and proven in the paper.

1.Introduction

In the resent years the problem of the software safety becomes more the and more acute, due to the widespread application of computer control in the safety critical systems- aircraft, spacecraft, nuclear and chemical plants, in the medical equipment, where a computer or software fault can cause big damage and lost of human lives.

To minimize the risks of dangerous software failure several strategies can be applied, namely it is possible :

a) to apply hazard analysis - define in which software modules the failure is dangerous and increase the reliability [1] of these modules;

b) to apply fault tolerant programming.

The mentioned above strategies have well known limitations even when they are applied simultaneously and the probability of a dangerous failure in many cases can't be made acceptable.

In the paper an additional method for improvement of software safety is suggested. It consists of synthesis of criteria of software safety which can be applied on line and off line. These criteria define the domains of nondangerous output signals for each particular application. The off line criterion can be used during the design of the software. The on line criterion is applied during the functioning of the computer system and if a dangerous failure is detected a transition to a safe state of the controlled object is executed.

The off line criterion of software safety is connected with the problem of practical stability of the software considering it as a limitation of the selected norm of output signals. A criterion of the practical stability of the software is suggested and proven with a theorem, considering the software as a discrete nonlinear system. Examples are shown. The on line criterion gives the sufficient conditions for dangerous failures of the computer system (not only software), while the off line criterion gives the sufficient conditions for stability and safety.

The described in the paper criteria were applied in a computer system for D.C. motor testing.

2.Software safety model and on line criterion for safety

Software failures f can be divided in two: dangerous f_d and non dangerous f_n: $f = f_d + f_n$, and accordingly the probabilities for failure are: $P_f = P_d + P_n$. The reliability of the software can be assessed by P_f and the safety - by P_d. This means that the main goal for improvement of the safety should be the reduction of the number of dangerous failures f_d and their reliability P_d. The specific measures for reduction of P_d include:

1.Hazard analysis and definition of the software modules, the failure of which can be dangerous.

2.Application of the fault tolerance strategy [2] for the software dangerous modules or for the whole system , using different versions of software to perform the same task. When applied this strategy improves not only the software safety , but also the software reliability . According to many authors the application of the fault tolerance strategy can't exclude all the errors in the software because the same sort of errors can be found in different versions of software , maid by different programmers, especially when their origin is in the specification of the software. That's why it is necessary to apply the fail safe strategy even when fault tolerance strategy is used.

3.The fail safe strategy is widely used in the hardware control systems but not in the software technology. This strategy consist of two steps: a) on line detection of the failure of the software during the execution of the program using a priori defined criterion for each module or for the whole system; b)on line transition of the controlled object to a safe state.

Concerning the step a) the following criterion for safety can be suggested.

Definition 1: If the norm of the output $\|\mathbf{X}(t)\|$ does not belong to the allowable domain Ω of non dangerous output signals:

(1) $$\|\mathbf{X}(t)\| \notin \Omega$$

then can be considered that a dangerous failure of the software has occurred .This criterion can be applied on line and can also be used as a basis for creation of a similar off line criterion, which should be applied during the design stage of the software.

In each specific case a proper norm $\|\mathbf{X}(t)\|$ should be sought. In the simplest cases the norms can be:

(2) $$\sup_{0 \le t \le \infty} |x(t)| < \mathrm{MAX}, \quad \text{or}$$

(3) $$\inf_{0 \le t \le \infty} |x(t)| > \mathrm{MIN},$$

where MAX and MIN are the allowable limits. Considering the on line application of the criterion with (2) the speed of the vehicle can be limited on line and with (3) the current of the excitation of D.C. motor should be stopped from dangerous falling. In another particular case Ω contains a set of allowable vectors of limited number of bits (i.e. codes) and $\mathbf{X}(t)$ can be compared with each element of Ω .In the general case however it is difficult to formulate a criterion for assessment of $\|\mathbf{X}(t)\|$, but in the most concrete cases it is possible. The assessment of the norms $\|\mathbf{X}(t)\|$ can be done by another processor or by the same processor if the problem of hardware reliability is solved.

Concerning the step b) there are also in the general case theoretical and practical difficulties , but in the railway signaling systems this problem is solved- a red light appears in the semaphore and the train is stopped. What should be done in an aircraft or in a nuclear plant depends mainly on the concrete failure of the software and the system which is controlled . In some cases the condition (1), (2), (3) can be verified using hardware. For example for the limitation of the minimal current of excitation of a computer controlled drive with D.C. motors, a minimum current relay can be used.

3.Off line application of the criterion of software safety

The described criterion (1) of software safety can be applied off line during the design of the software , considering the software as a discrete nonlinear system :

(4) $\mathbf{X}_{n+1}(t+1)= f [\mathbf{X}(t), t, \mathbf{X}_n(t+1), P]$, $\mathbf{X}(0) \in Q_0$, $t,t+1 \in [0,T]$

where $\mathbf{X}_{n+1}(t+1)$ and $\mathbf{X}_n(t+1)$ are two consecutive iterations in the process of reaching the solution in the moment (t+1), $\mathbf{X}(t)$ is the vector solution in the moment t; the set Q_0 contains the initial values of the vector $\mathbf{X}$, i.e. $\mathbf{X}(0)$; P is a set of parameters used in the software and influencing the solution- $p=\{p_1,p_2,...p_j\}$, $p \in P$. The vector $\mathbf{X}(t)$ describes the status of the system and the sequence of vectors $\mathbf{X}_{tr}$= $[\mathbf{X}(0), \mathbf{X}(1),....\mathbf{X}(t), \mathbf{X}(t+1)]$ is called trajectory of the system with software model (4).In the general case the right part of (4) is not an analytical function , but a system of procedures, subroutines and functions of the algorithmic language which is used.

In many cases Ω is simply connected domain Q , which can be defined in the following way:

(5) $Q\{ \mathbf{X}: \|\mathbf{X}\| \leq R_Q\}$, $Q_0 \in Q$,

where R_Q is the radius of the allowable domain. In this case the criterion (1) coincides with the following definition of practical stability of the software , suggested in this paper on analogy with the definition of practical stability of analog systems [3,4].

Definition 2: For given sets (5) the discrete nonlinear system (4) modeling the software , with first iterations $\mathbf{X}_0(1),...\mathbf{X}_0(t+1) \in Q$, initial value $\mathbf{X}(0) \in Q_0$ is practically stable for given Q_0, ε and time interval [0,T] if after (k+1) iterations

(6) $\|\mathbf{X}_{k+1}(t+1) - \mathbf{X}_k(t+1)\| < \varepsilon$

and $\mathbf{X}_{k+1}(t+1) \in Q$ for $(t+1) \in [0,T]$.

An additional explanation is that when after (k+1) iterations the condition (6) is satisfied , then $\mathbf{X}_{k+1}(t+1)$ is approximating $\mathbf{X}(t+1)$:

$\mathbf{X}(t+1)=\mathbf{X}_{k+1}(t+1)$, $(t+1) \in [0,T]$

Obviously the necessary number of iterations k+1 is different for each moment in the interval [0,T].The requirement $\mathbf{X}_0(1),\ldots \mathbf{X}_0(t+1) \in Q$ means that the first iterations should belong to the set Q , which can be easily fulfilled.

The inequality (6) will be reached and the computational process will be convergent for a given t if the operator (4) written as $\mathbf{X}_{n+1}= f[\mathbf{X}_n]$ satisfies the Lipschitz condition:

(7) $\|f(x_1) - f(x_2)\| \le q\|x_1 - x_2\|$, where $q<1$,

and considering in our case x_1,x_2 as a vectors.

The following criterion giving sufficient conditions for practical stability of software (and hence for software safety) in accordance with **Definition 2** is suggested in the following theorem:

Theorem: The sufficient condition for practical stability of software described with equation (4) is:

(8a) $\| \mathbf{X}_{n+1}(t+1)\| = \|f[\mathbf{X}(t), t, \mathbf{X}_n(t+1), P]\| \le R_Q$,

where the norm of the right part of (8a) is evaluated taking into account that:

(8b) $\mathbf{X}(0) \in Q_0$, $t,t+1 \in [0,T]$, $\mathbf{X}(t) \in Q$, $\mathbf{X}_n(t+1) \in Q$, $n \in [0,k+1]$

Proof: It will be made using the method of mathematical induction. For this purpose it is necessary to be proven that:

1.The discrete process is stable in the initial moment ,i.e. $\mathbf{X}(0) \in Q_0$ and each first iteration $\mathbf{X}_0(1),\ldots \mathbf{X}_0(t+1) \in Q$.

2.If the system (4) is stable for the moment t, i.e. $\mathbf{X}(t) \in Q$ and for iteration n in the moment t+1, i.e. $\mathbf{X}_n(t+1) \in Q$ then the n+1 iteration in the moment t+1 is also stable and belongs to Q. However this second condition is fulfilled if the criterion (8a),(8b) is satisfied. Then if using the induction method it follows that $\mathbf{X}_{k+1}(t+1) \in Q$, which means that the theorem is proven according to **Definition 2** if the Lipschitz condition (7) is satisfied.

The proven criterion (8a), (8b) for practical stability and safety of software is sufficient but not necessary because the right part of (8a) can be estimated in most cases only by some majorant value M of the norm:

(9) $\| \mathbf{X}_{n+1}(t+1)\| = \|f[\mathbf{X}(t), t, \mathbf{X}_n(t+1), P]\| \le M \le R_Q$,

which gives a pessimistic assessment of the stability.

When the right part of (4) or (9) is not simply analytically represented but is a software procedure or subroutine then the norm of the right part can be found as numerical solution of the optimization problem of nonlinear programming. If the norm (2) is used then the optimization criterion will be:

(10) $\sup_{0\le t\le T} |f[\mathbf{X}(t), t, \mathbf{Xn}(t+1), P]|$

The norm in (10) must be calculated taking into account the conditions (8b). The varied parameters should be : the numerical step ; t in the interval [0,T]; n in the interval [0,k+1], parameters in the set P. The method of seeking and finding the optimum can be chosen between the known methods of Monte Carlo, the gradient methods, ets. The so defined norm should be called numerical norm of the software procedure. It can be calculated with certain amount of accuracy and be used for investigation of the practical stability and safety of the software for safety critical applications.

4.Example

Let us consider that the software system (4) is:

(11) $x_{n+1}(t+1) = c_1 . \int_0^T [c_2 . x(t) + c_3 . x_n(t+1) + c_4 . P]\, dt$

The norms in (11) will be sought for Banach space with a norm $\|x\| = \sup_{0\le t\le T} |x(t)|$,

as follows:

$$\|x_{n+1}(t+1)\| \le c_1 . \int_0^T \| c_2 . x(t) + c_3 . x_n(t+1) + c_4 . P \|\, dt \le$$

$$\le c_1 . T[c_2 \| x(t) \| + c_3 \| x_n(t+1) \| + c_4 \| P \|] \le c_1 T[c_2 . (R_Q) + c_3 . (R_Q) + c_4 . \delta] \le R_Q,$$

where δ is the norm of P. Hence the condition for stability is :

(12) $(c_2 + c_3) . R_Q + c_4 . \delta \le R_Q / (c_1 . T)$

From (12) R_Q can be found:

(13) $R_Q = c_4 \delta / [1/(c_1 . T) - c_2 - c_3]$

The discrete computational process will be limited and practically stable and hence safe for a set Q defined by R_Q calculated from (13). On the other hand for a given R_Q one of the constants c_1 ,c_2, c_3, c_4 can be found as a function of the other three.

References

1.Trachtenberg M. "A general theory of software reliability modeling", IEEE Ttans. on Reliability, vol.39, No.1, 1990, April.

2.Chadleigh M." Software and safety : how compatible are they", Information and software technology , vol.32, No.5, June 1990.

3.J. La Saal, S. Lefschetz " Stability by Liapunov's direct method", Academic press, London, New York, 1961.

4.Krumov A. " A new criterion for practical stability of nonlinear dynamic systems", International conference on electrical machines, Budapest, 5-10, IX, 1982.