Механика Транспорт Комуникации

ISSN 1312-3823 (print)
ISSN 2367-6620 (online)
том 14, брой 1, 2016 г.

Научно списание **http://www.mtc-aj.com** **статия № 1280**

СОФТУЕР ЗА КРИТИЧНИ ЗА БЕЗОПАСНОСТТА ПРИЛОЖЕНИЯ – ИЗИСКВАНИЯ, ПРОБЛЕМИ И РЕШЕНИЯ

Мария Христова
mhristova@vtu.bg

Висше транспортно училище „Тодор Каблешков”
София 1574, ул. „Гео Милев” № 158
БЪЛГАРИЯ

Ключови думи*: критични по безопасност системи, критичен по безопасност софтуер, грешки в софтуера, отказоустойчивост, надеждност, надеждност на софтуера*

Резюме: *Предмет на проучване са проблемите на софтуера, критичен за безопасността в различни приложения. Предложена е следната логическа конструкция: не трябва да се допускат грешки на никой от етапите и нивата на изграждане на софтуера. За целта се привличат създадените от науката подходи, формални методи и средства за програмиране, както и действащите стандарти за критичен софтуер. Но в сложни системи грешки продължава да има. За да се открият и отстранят грешките преди пускането на системата в експлоатация, се прилагат подходи, методи и средства за верификация и off-line тестване. Въпреки това, в експлоатация остават немалко грешки, някои от които водят до ограничаване на функционалността, а други могат да предизвикат опасни инциденти. Неблагоприятното влияние на едните и опасното въздействие на другите може да се ограничат, ако грешките се открият по време на работа (on-line), последствията им се стопират и грешките се отстранят или се толерират, за да не се проявяват.*

Прави се изводът, че проблемите на критичния за безопасността софтуер са свързани с нормите за допустим риск, методите за изграждане на безопасен и безгрешен софтуер, методите за изграждане и структурите на безопасни след отказово поведение (fail-safe) и отказоустойчиви (fault-tolerance) критични по безопасност системи (SCS), оценката за опасността след отказ, възможностите тя да се намали, както и моделите на надеждността и безопасността, оценяващи влиянието на грешките на софтуера.

1. ПОСТАНОВКА

Предмет на проучване в този труд са проблемите на **софтуера**, **критичен за безопасността** в различни приложения.

Критична по безопасност система (**S**afety **C**ritical **S**ystems - SCS), *е система при която безопасността на човека зависи от правилното функциониране на системата.*

SCS управляват особено отговорни технологични процеси, чиито откази могат да доведат до заплаха за здравето и/или загуба на живот, на големи материални и

човешки ценности и/или недопустимо увреждане на околната среда. Неизпълнението на предвидените функции на SCS може да има животозастрашаващо въздействие.

Целта на статията *е да се структурира известната информация за софтуер, критичен за безопасността, и се предложат насоки за решения на оставащите открити въпроси.*

2. СПЕЦИФИЧНИ ИЗИСКВАНИЯ КЪМ СОФТУЕРА ЗА SCS

2.1 Задачи на софтуера за SCS

Софтуерът е централен компонент в SCS. Задачите на софтуера в системите, свързани с безопасността, са три:

1. Да постигне дефинираната от спецификацията ***функционалност***.
2. Да отговори на изискванията за ***безопасност.***
3. Да ***контролира и координира***.

Първата задача се поставя за всички системи, включително за системи, които не носят отговорност за безопасността - и в системите с пакетен режим, и в системите за реално време.

Втората задача е специфична и се отнася само за софтуер, релевантен на безопасността. Софтуерът трябва да е проектиран така, че да няма откази, от които произтича опасност. Ако софтуерът на всички нива – от концепцията, през спецификацията, структурата и кодирането са перфектни и съобразени с изискванията за безопасност, втората задача не би следвало да се поставя. Но това не е така, поне в сложните системи.

Третата задача е да координира и контролира действието на различни части на системата: сигналите на вход и изход се филтрират, потребителският интерфейс на много системи е софтуерно базиран, а цялостното функциониране на системата се контролира от софтуера [1].

Системата трябва да е така проектирана, че опасностите да се откриват и отстраняват преди да предизвикат инцидент. Тя трябва да включва защити, които минимизират инцидентите и последствията от злополуките. Ето защо към хардуера, софтуера и преносния тракт на SCS системите, които обработват и пренасят информацията, се поставят повишени изисквания за надеждност и недопустимост на грешни управляващи въздействия след откази.

Разработването на критичен за безопасността софтуер е предизвикателство, защото той не само трябва да осигурява необходимите функции, но и да го направи по начин, който да гарантира, че системата е безопасна и сигурна за експлоатация.

2.2 Подходи за решение

Приемливо ниво на безопасност на софтуера за критични приложения може да се постигне само, когато се отдава първостепенно значение на изискванията към системата в нейната спецификация. Една грешка в софтуерната спецификация или липсата на важни изисквания може да причини отказ на системата, от който да последва погрешно решение.

Инженерите, на които е поверена безопасността на системата, трябва да работят съвместно със софтуерни специалисти, за да се идентифицират онези грешки, които могат да причинят вреди или да предизвикат нежелани събития. Проблеми възникват, когато едни специалисти трябва да комуникират със специалисти от други дисциплини. Разногласията могат да доведат до непоследователни или липсващи изисквания. На следващ етап решаващо значение за безопасността имат инженерните анализи и непрекъснатото управление по време на цялата разработка и жизнен цикъл на системата. Анализът на безопасността се извършва по редица утвърдени методи, като

„дървото на опасните откази” [2], Failure Mode Effects Analysis (FMEA) [3], Мрежи на Петри и др.

Разработването, проектирането и внедряването на критичен за безопасността софтуер изисква приложението на специфични методи и техники в софтуерния системен инженеринг. Цената на отказа на SCS-система е достатъчно висока, за да оправдае за разработка на нейния софтуер методите, които за други системи не са икономически ефективни.

Естествено, работата по софтуера започва със съставянето на ***спецификацията,*** която съдържа функционалните и безопасносни изисквания за конкретното приложение.

Рисково базираната спецификация [4]е подход, който е широко използван от разработчици на SCS системи. Тя се фокусира върху тези събития, които биха могли да причинят най-големи щети или, за които има вероятност да се появят често.

Основаващата се на риска спецификация като подход включва:

- разбиране на рисковете, пред които е изправена системата;
- откриване на техните първопричини и
- генериране на правила за управление на тези рискове.

Безопасност в софтуерни системи се постига чрез разработване и разбиране на ситуациите, които биха могли да доведат до откази, свързани с безопасността. Стремежът е софтуерът да се проектира така, че да няма такива откази. Това не е толкова лесно, а и не е достатъчно. Надеждните системи могат да бъдат опасни и обратно - ненадеждните да са безопасни.

Според [4] има поне четири причини, поради които надеждни софтуерни системи не винаги са безопасни:

1. Никога няма 100% сигурност, че софтуерната система изчерпателно е изпълнила изискванията за липсата на опасни откази. Неоткрити грешки могат да бъдат пасивни в продължение на дълъг период от време и софтуерни откази могат да се появят след много години на надеждна работа.

2. Когато хардуерни компоненти са близо до състояние на физически отказ, те се държат нестабилно и генерират сигнали извън диапазоните, които могат да бъдат обработвани от софтуера. Софтуерът или отказва, или погрешно интерпретира тези сигнали, което може да доведе до опасност.

3. Неправилно функциониране на системата може да е резултат от спецификацията, а не от грешки на кодирането. Спецификацията може да е непълна, ако не съдържа описание на необходимото поведение на системата в някои критични ситуации.

4. Системните оператори могат да генерират входове, които в някои ситуации, може да доведат до опасен отказ на системата. Един невероятен пример за това е приведен в [4]. Ходовата част на самолета се е срутила, докато е на земята. Причината: техник е натиснал бутон, който задейства софтуера за вдигане на колесника. Софтуерът перфектно изпълнява възложената му функция, но трябва да му е разрешено да я извърши само, когато самолетът е във въздуха.

Преди да бъде одобрен и сертифициран за използване, софтуерът трябва да отговаря на строги изисквания, формулирани в стандарти.

3. СЪСТОЯНИЕ НА ИЗСЛЕДВАНИЯТА ПО ПРОБЛЕМИТЕ НА КРИТИЧНИЯ ПО БЕЗОПАСНОСТ СОФТУЕР

3.1 Маркери в развитието на науката по критичния софтуер

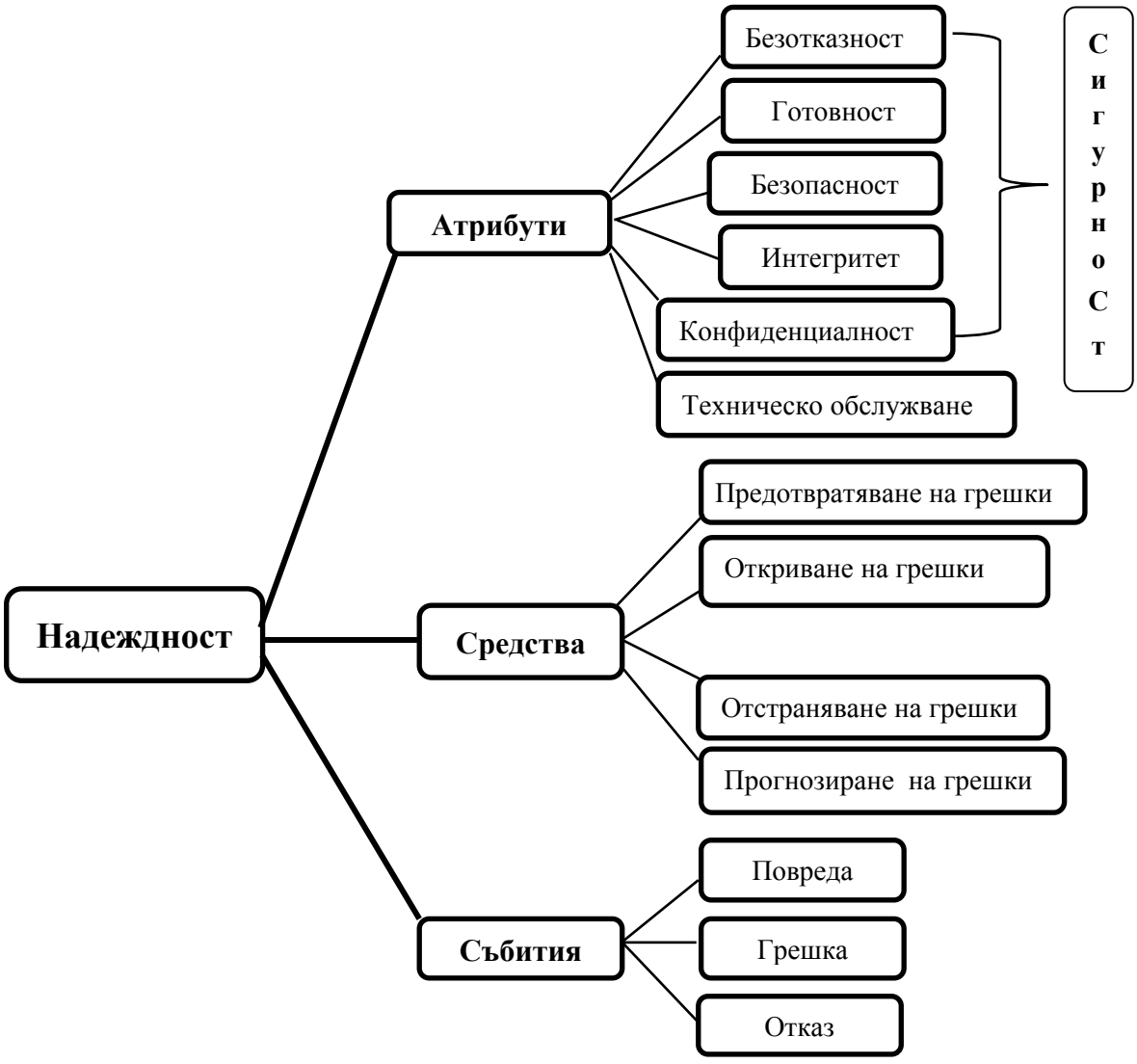
Развитието на науката за надеждността на програмното осигуряване бележи началото си от 70-те години на миналия век. Може да се приеме, че началото на

научното поставяне на проблема за надеждността на софтуера е монографията на G. J. Mayers „*Software Reliability: Principles and Practices*" [5]. Дотогава, а отчасти и след това, някои учени упорито твърдят, че не може да се говори за надеждност на нещо, което е идеално, не се поврежда, не се чупи, не се износва, не остарява. Софтуерът е точно такава компонента на компютъра.

Доказва се, че надеждността може да се третира по същите подходи и принципи, както хардуера, че и във формалните методи за моделиране на надеждността на софтуера трябва да се разчита на математическата теория на вероятностите, въпреки че между надеждността на софтуера и на хардуера има големи и принципни разлики.

Към началото на тази епоха от 70-те години, свързана с надеждността на софтуера, се отнасят повечето модели на софтуерна надеждност [6, 7, 8, 9]. В тези модели намират израз връзката между грешките в софтуера, изявата им като откази, интензивността на тяхното отстраняване и показателите за надеждност, по-специално, породената от грешките интензивност $\lambda_\varepsilon(t)$ на потока откази и вероятността за безотказна работа на съответната система. Броят на моделите на софтуерна надеждност днес са повече от сто и продължава да расте.

Един заслужаващ маркиране проблемен кръг е свързан с понятията в сферата на софтуерната надеждност. За понятийния апарат фундаментално значение има



Фиг.1. Понятийно пространство „надеждност – сигурност - отказоустойчивост"

публикацията [10] „*Basic Concepts and Taxonomy of Dependable and Secure Computing*", написана от Algirdas Avizienis, Jean-Claude Laprie и Brian Randel - колосите в тази специфична наука. Въпреки че в проблемното поле на софтуерната надеждност имаше интензивна научна дейност и се бяха появили множество научни публикации, до 2004 г., когато излиза този труд, съществува твърде голяма дисперсия и размитост в понятийния апарат на надеждностно-безопасносното пространство.

За да се сложи ред в понятията и таксономията, още през 1995 г. Съвместният комитет на IEEE CS „*Основни понятия и терминология*" формира работна група по „10.4 Dependable Computing & Fault Tolerance". Цитираният труд е резултат от положените усилия на работната група и сполучлив опит за минимален консенсус по понятията, необходими за техническото взаимодействие между специалности в различни сфери. Понятийният апарат и таксономията на надеждни и сигурни компютри беше разширена, усъвършенствана и опростена. Направен е опит за изясняване на връзката между надеждност, безопасност и сигурност на човешките грешки (вкл. злонамерени), релевантни на компетентността, на способността за оцеляване, високо-доверителните системи и др. На фиг. 1 е дадена структурна схема на най-често ползваните понятия.

Съществено значение за развитието на проблематиката в тази сфера имат и европейските стандарти, адресирани към SCS в железопътния транспорт: *EN50126* „The specification and demonstration of Reliability, Availability, Maintainability and Safety – *RAMS"* и *EN50128 „*Software for control and protection systems*",* стандартите за критичен софтуер в авиацията, космоса, медицината и др.

3.2 Откъде идват и какви са проблемите на критичен за безопасността софтуер?

Ако софтуерът е перфектен – и по спецификация, и по алгоритмизация, и по кодиране, и по документиране, проблеми за безопасността нямаше да има. Когато обаче на някои от тези етапи са допуснати грешки и те се активират, следствието е нарушена работоспособност и появата на откази. Да се активират грешки означава да се подадат подходящи данни и при конкретното изпълнение алгоритмът да мине през маршрут, елемент (данни, код, условен преход), в който е грешката. Нарушената работоспособност може да предизвика поведение, застрашаващо безопасността.

Ето защо проблемите на софтуера за SCS се свеждат практически до необходимостта от безгрешното изграждане на целия софтуер и перфектно програмиране на функционалността на системата. Това обаче в сложни системи е невъзможно. Някои от грешките водят до нарушаване на работоспособността, без да влияят върху дефинираните като опасни последствия. С откриването и отстраняването им надеждността нараства, но безопасността остава същата. Подобряването на надеждността не е задължително да подобри безопасността [11].

Логиката на софтуерната проблематика в изучавания контекст е следната:

Не трябва да се допускат грешки на никой от етапите и нивата на изграждане на софтуера. За целта се привличат създадените от науката подходи, формални методи и средства за програмиране. Съблюдават се стандартите за критичния за безопасността софтуер. Тъй като, въпреки това, в сложни системи грешки има, трябва да се направи необходимото допуснатите грешки да се открият и отстранят преди пускането на системата в експлоатация. За целта се прилагат съвременните подходи, методи и средства за верификация и *off-line* тестване на програмата.

В сложни системи, въпреки верификацията, винаги има, при това немалко, остатъчни грешки. Някои от тях само нарушават нормалното функциониране и водят до спиране на управлявания процес или ограничена функционалност. Други могат да

предизвикат опасни инциденти. Последните са предмет на особено внимание по методите на рисково базираните спецификации.

Неблагоприятното влияние на едните и опасното въздействие на другите остатъчни грешки могат да се ограничат, ако:

а) грешките се открият по време на работа (*on-line*), въздействието им се стопира и те се отстранят;

б) грешките се толерират чрез средствата на отказоустойчивото програмиране, за да не се проявяват.

4. КЛАСИФИКАЦИЯ НА ПРОБЛЕМИТЕ В КРИТИЧЕН ЗА БЕЗОПАСНОСТТА СОФТУЕР

От направените дотук проучвания могат да се направят обобщения за проблемите на критичния за безопасността софтуер.

1. Съставяне на ***рисково базирана спецификация,*** която е съобразена с възможните опасности след откази на системата за управление
2. Изисквания към и избор на ***езици за програмиране,*** които са най-подходящи за критичен за безопасността софтуер;
3. Методи и средства за откриване и отстраняване на ***грешките.***
4. Количествено оценяване на ***влиянието на грешките*** върху интензивността на отказите, и оттам, върху надеждността и безопасността.
5. Методи и алгоритми за ***толериране на грешките*** и потискане на тяхното влияние върху функционалността и безопасността на системите.

От проучванията може да се направи изводът, **че софтуерните проблеми на *Safety Critical Systems*** са в две пространства:

Функционална безопасност

Функционалната специфика произтича от предназначението на системата. Функциите на различните по предназначение системи са описани в техните технико-експлоатационни изисквания, спецификации, стандарти и др. съпътстващи документи. На тази основа учени, изследователи и проектанти създават техническите средства, а специализирани програмисти разработват софтуер, който изпълнява дефинираните изисквания – за системи на въздушния транспорт, железопътния транспорт, атомната енергетика, медицината и т.н.. Това е работа на професионалисти, разработчици и проектанти, в т.ч. програмисти с експертен опит в съответната сфера.

Разбира се, функционалността е свързана с безопасността. Ако не се познават или се нарушават някои принципи, ако не всички условия за безопасност са предвидени в проекта, не са намерени правилните реакции на входните въздействия и т.н., системата може да допуска опасни управляващи команди и когато работи нормално, както е зададено от проектанта. Причините за опасни изходни въздействия в тези случаи се дължат на неправилни решения от незнание, неумение, недостатъчна компетентност, недоглеждане, липса на опит и т.н. и в крайна сметка - дефицит на перфектност.

Надеждностно-безопасностно пространство, което е общото за всички системи, независимо от тяхното предназначение и функционалност, и като проблем понякога доминира в този клас системи. В това пространство се влиза като се предположи, че спецификацията, по която системата е изпълнена, е перфектна и че когато SCS е работоспособна, тя изключва опасните управляващи въздействия. Опасностите се създават след откази.

Проблемите в това пространство ***на критичния софтуер*** за SCS са свързани с:

- нормите за допустим риск, най-често нормиран в международни стандарти за системите, в които се прилага;

- методите за изграждане и структурите на *fail-safe и fault-tolerance SCS*, с които с постига съответствие със стандартите;
- методите за изграждане на безопасен и безгрешен софтуер;
- оценката за опасността след отказ и възможностите тя да се намали;
- вероятностните модели на надеждността и безопасността на съответната структура, оценяващи влиянието на грешките на софтуера и др.

При цялото богатство на литературни източници и научни публикации в това второ пространство има какво още да се прибави. Важни елементи на това пространство са *формалните методи за синтез и анализ, езиците за програмиране, валидацията и сертификацията на софтуера, съпровождането на софтуера през жизнения му цикъл*. Непреходно актуални са методите и средствата за разработка на безгрешен софтуер, за откриване и отстраняване на грешките. Има дефицит в количествените модели, отчитащи влиянието на софтуерните грешки върху надеждността и в частност върху опасните откази на системите, а без количествени оценки не може да се прецени доколко ефективно е дадено решение.

ЗАКЛЮЧЕНИЕ

В критичния за безопасността софтуер са постигнати сериозни и не подлежащи на съмнение научни и практически резултати, които позволяват да се изграждат и експлоатират релевантни по безопасност системи за управление. От направените в този труд проучвания на богатото на литературни източници пространство се извеждат обобщенията, че остават непреходно актуални редица открити въпроси, които заслужават вниманието на науката. Авторски амбиции са по-специално изследванията на зависимостите между остатъчните грешки в софтуера, надеждността на системите и възможностите за подобряване на безопасността им чрез тяхното откриване и толериране чрез диверситета на софтуера.

ЛИТЕРАТУРА

[1] Elakeili, S. M., Fail-Safe Test Generation of Safety Critical Systems Electronic Theses and Dissertations,Paper 180, 2015

[2] Vesely W.E., F.F. Goldberg, N.H. Roberts, D.F. Haasl, Fault Tree Handbook (NUREG-0492), 1981

[3] Threat Effects Analysis: Applying FMEA to Model Computer System. Annual Reliability and Maintainability Symposium - February ISBN: 1-4244-1461-X Library of Congress 78-132873 IEEE 2008 Proc. Ann. Reliability & Maintainability Symp. 2008

[4] Sommerville I., Safety Engineering, pp. 1-36, 2013

[5] Mayers G.J. Software Reliability: Principles and Practices, New York, Wiley, 1979 г.

[6] Asad Ch. A., M. Irfan, M. J. Rechman, An approach for software reliability model selection - IEEE Computer Society Press, 2004

[7] Shooman M.L. Operational Testing and Software Reliability Estimation During Program Developments - IEEE Computer Society, 1973

[8] Coutinho J. deS, Software Reliability Growth - IEEE Symposium on Computer Software Reliability, 1973

[9] Musa J. D., Okumoto, K., Software Reliability Models: Concepts, Classification, Comparisons, and Practice - Electronic Systems Effectiveness and Life Cycle Costing, 2000

[10] Avizienis, Jean-Claude Laprie и Brian Randel „*Basic Concepts and Taxonomy of Dependable and Secure Computing*", написана от Algirdas IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 1, NO. 1, JANUARY-MARCH 2004

[11] Fujiwara T., M. Kimura, Y. Satoh, S. Yamada, A Method of Calculating Safety Integrity Level for IEC61508, Conformity Software, 17th IEEE Pacific Rim International Symposium on Dependable Computing , 2011

SOFTWARE FOR SAFETY CRITICAL APPLICATIONS – REQUIREMENTS, PROBLEMS AND SOLUTIONS

Mariya Hristova
mhristova@vtu.bg

Todor Kableshkov University of Transport
1574 Sofia, Geo Milev str. 158
BULGARIA

Key words: *safety critical systems, safety critical software, errors in the software, fault tolerance, reliability, reliability software*

Abstract*: The paper is dedicated to a study on the problems of safety critical software in various applications. The following logical structure is proposed: there must be no mistakes made in any of the stages and levels of software development. For that purpose, some approaches, formal methods and tools of programming created by science as well as operating standards for critical software are used. However, in complex systems errors continue to exist. To detect and remove errors before putting the system into operation, approaches, methods and tools for verification as well as off-line testing are applied. Nevertheless, a number of mistakes still remain in operation, some of which result in limiting functionality while others can cause dangerous incidents. The unfavourable effect of the former and the dangerous effects of the latter can be limited if errors are found during operation (on-line); the consequences are stopped and they are removed or tolerated not to appear. It is concluded that the problems of safety critical software are connected with the standards of admissible risk, methods for building safe and fault-free software, methods for building structures of fail-safe and fault-tolerance SCS, the assessment of post-failure danger, the possibility to reduce danger as well as models of reliability and safety evaluating the impact of software errors.*