计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2018,27(1):66−71 [doi: 10.15888/j.cnki.csa.006143]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于构件的可信软件系统冗余机制及可靠性分析①

郁　湧[1,2], 黄宇鑫[1], 陈　浩[1]

[1](云南大学 软件学院, 昆明 650091)
[2](云南省软件工程重点实验室, 昆明 650091)

摘　要: 在高可信软件的设计和开发中, 软件容错是提高系统可信性的一种实现技术之一. 容错性就是指软件在故障出现时保证提供服务的能力, 对退化故障进行容错的一种处理方式就是依靠冗余技术. 本文在分析结构冗余及其对可信性的影响的基础上, 在基于构件的可信软件系统中提出了对核心构件进行冗余的机制, 包括单个构件的双模冗余结构、组合构件的双模冗余结构和构件的三取二冗余及其扩展结构, 并给出了其故障检测和判断方法. 同时, 在各种冗余结构的基础上对系统可靠性能进行分析.

关键词: 可信软件; 基于构件的软件系统; 软件容错; 结构冗余; 可靠性分析

引用格式: 郁湧,黄宇鑫,陈浩.基于构件的可信软件系统冗余机制及可靠性分析.计算机系统应用,2018,27(1):66–71. http://www.c-s-a.org.cn/1003-3254/6143.html

Redundancy Mechanism and Reliability Analysis of Trusted Software System Based on Component

YU Yong[1,2], HUANG Yu-Xin[1], CHEN Hao[1]

[1](School of Software, Yunnan University, Kunming 650091, China)
[2](Key Laboratory for Software Engineering of Yunnan Province, Kunming 650091, China)

Abstract: In the design and development of high confidence software, the software fault tolerance is one of the techniques to improve the credibility of the system. Fault tolerance is the ability of software to guarantee the service when the fault occurs. And a processing method for fault tolerance is to rely on redundancy technology. Based on the analysis of the structural redundancy and its influence on the credibility of the system, this paper proposes a redundancy mechanism for the core components of the component-based trusted software. The redundancy structure includes dual redundant structure for single component and composite components, 2 out of 3 redundant structure and its extension. And the fault detection and the judgment method are given. At the same time, the reliability of the system is analyzed on the basis of various redundant structures.

Key words: trusted software; software system based on component; software fault tolerance; structural redundancy; reliability analysis

当今, 以高速通信、海量存储和高性能计算为核心的信息基础设施已经广泛深入地渗透到经济、政治、军事和社会文化生活的各个层面, 成为现代生产力发展和人类文明进步不可或缺的强大工具. 在众多应用背景的推动下, 软件的复杂度和规模都在以前所未有的速度在不断延伸, 在金融、国防、政府和通信等关键领域的各种复杂应用需求背景下, 软件是否可信已经成为衡量软件系统的重要指标. 然而, 作为计算机技术的核心和基础之一的软件系统, 其生产现状和质量一直不能令人满意, 尤其是应用于航空航天、核电及国防等领域的安全关键软件系统, 其失效常常会对人类和环境造成严重的乃至灾难性的后果. 早在

① 基金项目: 国家自然科学基金 (61462091)
收稿时间: 2017-03-21; 修改时间: 2017-04-13; 采用时间: 2017-05-02; csa 在线出版时间: 2017-12-22

1991 年, Laprie 就从安全关键系统的研究出发提出了软件可依赖性 (Dependability) 的概念[1]. 1997 年美国国家科学技术委员会在《高可信系统的研究挑战》中明确提出了高可信性 (High Confidence) 的概念[2]. 我国学者陈火旺、王戟等认为高可信软件在系统提供服务时应能满足一系列可靠安全性、实时性、可靠性、容错性、保密性等关键性质[3]. 可信软件作为软件领域最具挑战性和价值的研究课题之一, 引起了国内外学者的高度重视.

软件容错是提高系统可信性的一种实现技术之一, 其相关的研究方面主要分为两大类: 软件冗余和时间冗余[4-6]. 软件冗余是在系统设计时, 增补一些部件或模块, 使得即使其中一个部件发生故障, 而整个系统照样完成规定的任务. 从冗余的范围来看, 分为元件冗余、部件冗余、子系统冗余等. 从部件联接形式来分, 可分为并联、旁联、表决系统等. 多版本编程利用完成同一功能的不同实现之间的多样性互补容错, 也是一种常见的软件冗余容错方法. 而时间冗余方面则是基于失败重做 (Retry-on-failure) 的思想, 如在系统进行设计和实现时设置检查点和回滚机制, 当发生故障就回滚到适当的检查点重新执行[7]. Reis 等人在编译器级别通过指令复制和合并对软件进行版本冗余从而可以在设置的同步点检查指令的一致性, 具有较好的容错效果和执行效率, 但其实现较为复杂[6]. 文献[8]通过动态监控和回滚技术, 建立了合适的还原点来对系统进行监控, 使系统能够及时恢复到预先的还原点. 文献[9]为提高软件的可靠性和生存能力, 分析了模块化对可靠性的影响提出一种基于进化计算的可进化模块冗余软件混合容错模型. 文献[10]设计了一种基于网络控制的可编程控制器冗余系统, 使得双机软件冗余系统更加稳定. 文献[11]给出了一种基于三取二冗余结构的安全计算机系统.

在现代软件工程技术中, 系统构件化已经成为软件技术总体发展趋势之一[12], 基于构件的软件开发技术尤其得到了广泛发展[13]. 为了能够提高基于构件的软件系统的可信性, 本文在分析系统结构冗余和可信性关系的基础上在基于构件的软件系统中提出一种构件结构冗余的方法并对其可靠性进行性能分析.

1 软件结构冗余及其对可信性的影响

软件系统的可信性质是指该系统需要满足的关键性质, 包括可靠性 (reliability)、可靠安全性 (safety)、保密安全性 (security)、生存性 (survivability)、容错性 (fault tolerance) 等等, 当软件一旦违背这些关键性质会造成不可容忍的损失时, 称这些性质为系统的高可信性质[3]. 构建的软件系统不够可信的原因就是故障的存在, 故障的存在说明软件系统内部有缺陷的部件. 软件系统故障的种类很多, 退化故障就是其中常见的一种. 当系统内的一个部件发生失效, 不再工作了, 则认为发生了退化故障. 软件系统中的退化故障可能是活动的, 也可能是休眠的; 可能是瞬时性的, 也可能是永久性的故障. 为了降低软件故障的发生率, 如果能够识别或者在进行系统设计时确定系统中可能发生故障的关键部件, 预先进行调节, 并确保失效部件的影响不会带来使系统发生失效的输出, 系统本身就不会失效.

软件系统需要对退化故障进行处理, 系统的容错是处理退化故障的方法之一. 容错性就是指软件在故障出现时保证提供服务的能力, 对退化故障进行容错的一种处理方式就是依靠冗余. 所谓软件的冗余技术主要就是指在软件设计和实现中, 除了完成系统本身所需的功能外, 为了能够提高系统的性能及可靠性等而额外增加一些合理的部件和程序代码的技术.

基于构件的可信软件系统中, 构件是具有一定规模、相对独立、可替换的单元, 它具有较稳定的组成模式, 完成一项确定、可区分的功能, 并遵从和提供一套接口以及这些接口的实现. 构件是软件系统的构成要素, 同时也是软件的承载体, 一个构件应该包括两个部分: 接口和实现. 其中接口部分定义了构件所提供的功能并规范了功能的使用方法; 而实现部分包括了构件所能提供的一系列相关操作. 在基于构件的可信软件结构设计中, 构件冗余就是对可能会出现故障的关键构件进行多个备份; 但是, 冗余并不意味着简单的备份, 冗余意味着一个构件有多个功能相同的构件是可用的, 超过提供服务所需的部件数量, 当其中一些构件发生失效时, 其他的冗余部件可以继续提供服务, 从而保证软件系统运行的可信性. 在系统中, 如果两个构件 A_1 和构件 A_2 所实现的功能和对应接口完全一样, 则称构件 A_1 和构件 A_2 是相互冗余的构件, 相互冗余的构件的规约和消息传递机制必修一致. 一个存在冗余构件的系统在合理调用的情况下不仅不会影响系统的实现, 而且会提高系统的容错能力, 满足系统可信性的要求. 相同冗余构件的运行要在确定的系统环境和相同输入的情况下, 才能得以正确运行; 否则, 存在外界的影响会产生错误的结果. 当一个构件产生故障时, 继续提供服务的构件需要进行数据和环境的检测和重新配置来保证系统运行的一致性.

为了保证基于构件的可信软件系统中的构件冗余结构的可靠性在运行过程中能够得到合理有效地验证, 需要具有一个称之为信任根的构件, 信任根构件是系统的可信启动模块, 可以存放构件的标识和编号、构件的信息摘要以及构件冗余信息的存储等重要数据, 同时也可以用来对冗余构件的运行情况就行实时检测, 从而判断各个构件运行是否正常. 基于冗余机制的可信软件框架如图 1 所示.

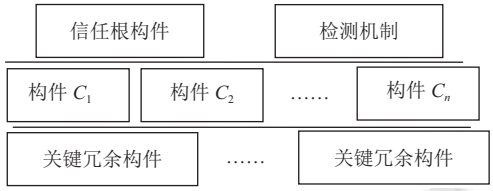

图 1 基于冗余机制的可信软件框架

2 单个构件的双模冗余结构及性能分析

一个具有双模冗余结构的系统是指在系统中存在两个完全冗余的构件; 在此系统中, 两个相同构件并行运行, 并将结果进行检测. 两个在相同输入和相同环境的情况下运行, 所产生的运行结果应该是一样的. 一种普遍使用的简单双模冗余结构如图 2 所示.

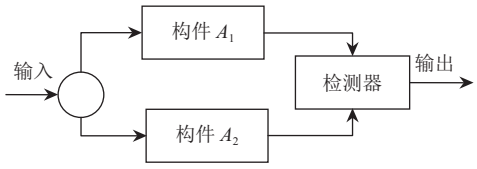

图 2 构件的双冗余结构

在基于构件的软件系统中, 对于系统中的关键或者核心构件, 为了保证运行过程中所产生的故障能够被及时发现, 提高系统容错阶段的检测能力, 可以对其进行双模冗余结构设计.

在进行双模冗余结构的设计时, 两个完全冗余的构件 A_1 和构件 A_2 的输入必须相同, 即在进行消息传递时, 需要把同一消息传递给冗余的构件. 双模冗余结构的错误检测是通过比较两个冗余构件的输出结果来完成的. 如果输出不同, 那么就是发生了错误. 但是, 通过错误检测无法确定是哪个构件发生了故障. 如果输出的结果完全相同, 则说明系统没有发生故障, 此时只需要把其中一个输出传递给下一个构件即可.

一个双模冗余结构会对系统的可信性产生影响, 因为单个构件的运行结果无论是否正确都不可能完成故障和错误检测, 错误状态产生而未被检测出来, 将会给系统带来无限的负面影响, 而双模冗余结构是一种最简单提供了错误检测的能力的方法.

双模冗余结构的可靠性分析: 假设两个完全冗余的构件 A_1 和构件 A_2 独立运行, 两个构件产生故障的概率相同都为 p, 由于构件 A_1 和构件 A_2 建立的是个并联系统, 则系统运行中检测出故障的概率为$1-(1-p)^2$.

两个完全冗余的构件 A_1 和构件 A_2 运行中, 一个产生故障而另一个没有产生故障而整体被检测成产生故障的概率为$2p(1-p)$.

从双模冗余结构的性能分析结果可以看出, 该结构与不用冗余结构相比可能会增加系统整体被检测出故障的概率, 因为无法判断哪个构件产生故障, 所以一个构件故障而另一个没有故障产生时的结果会判定为系统产生故障.

对于双模冗余结构, 如果在系统运行过程中能够收集系统产生故障的可能环境或情况, 在对应情况下才进行两个完全冗余的构件的调用, 否则只需要进行其中一个构件的调用, 这样就可以提高运行效率.

3 组合构件的双模冗余结构及性能分析

软件系统中的构件可能具有多种关系, 比如并联、串联等, 对于多个需要进行冗余处理的关键构件, 为了能够得到更好的效果, 可以对其进行冗余的组合. 本论文主要对两个构件的并联、串联的组合关系进行分析, 对多个构件的关系可以进行相似处理.

3.1 两个串联构件的双模冗余方式及性能分析

对于两个串联构件的冗余主要有两种方式, 如图 3 和图 4 所示, 图中相同冗余构件用不同的下标表示, 如构件 A_1 构件 A_2 和构件 A_3 是三个相同的冗余构件, 构件 B_1 和构件 B_2 是两个相同的冗余构件, 其他以此类推.

对于串联构件双模冗余方式一, 两个构件串联运行之后再检测其运行结果是否相同来确定是否有故障发生, 把两个构件看成一个整体来检测, 检测数量会变少, 但是就算检测出存在故障, 也不知道是哪个构件造成的故障.

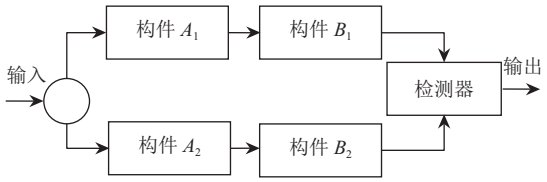

图 3 串联构件双模冗余方式一

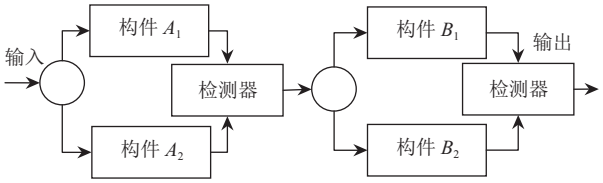

图 4 串联构件双模冗余方式二

若构件 A_1 和构件 A_2 的故障概率为 p_1, 构件 B_1 和构件 B_2 的故障概率为 p_2, 且构件 A_1 和构件 A_2、构件 B_1 和构件 B_2 的运行都是独立的, 则构件 A_1 和构件 B_1 串联时产生故障的概率为:

$$p_1(1-p_2)+p_2(1-p_1)+p_1p_2 = p_1+p_2-p_1p_2$$

此时, 构件 A_1、构件 A_2、构件 B_1 和构件 B_2 中只要有一个构件产生故障, 检测器就是认为系统发生故障, 其概率为:

$$1-(1-(p_1+p_2-p_1p_2))^2$$

对于串联构件双模冗余方式二, 两个串联的构件分别运行之后就检测其运行结果是否相同来确定是否有故障发生, 把两个构件分开来进行检测, 检测数量会增加, 可以根据检测结果知道是构件 A_1 或 A_2 还是构件 B_1 或 B_2 造成了系统故障.

对应方式 2 的串联构件双模冗余, 若构件 A_1 和构件 A_2, 构件 B_1 和构件 B_2 分别进行检测, 运行中检测出故障的概率分别为$1-(1-p_1)^2$和$1-(1-p_2)^2$.

对于两个冗余检测之间是串联关系, 此时, 构件 A_1 和构件 A_2 检测出故障就不用运行构件 B_1 和构件 B_2, 因此构件 B_1 和构件 B_2 也不需要检测. 只有构件 A_1 和构件 A_2 运行无故障时才需要运行构件 B_1 和构件 B_2 并对其进行检测. 因此, 运行中检测出产生故障的概率为: $2p_1(1-p_1)+(1-2p_1(1-p_1))(1-(1-p_2)^2)$.

从上可知道, 串联构件采取哪种冗余方式与串联的两个构件产生故障的概率有关, 可以根据情况进行选择.

3.2 两个并串联构件的双模冗余方式及性能分析

对于两个并联构件的双模冗余主要有两种方式, 如图 5 和图 6 所示.

对于并联构件双模冗余方式一, 两个并联关系的构件 A_1 和构件 B_1 以及并联关系的构件 A_2 和构件 B_2 先并联运行, 之后再把两组并联构件运行的结果进行检测来确定是否有故障发生, 此时把两个并联关系的构件看成一个整体来检测, 检测数量会变少, 但是只要构件 A_1、构件 A_2、构件 B_1 和构件 B_2 中一个构件发生故障, 都会认为系统出现故障. 就算检测出存在故障, 也不知道是构件 A_1 或 A_2 还是构件 B_1 或 B_2 造成的故障.

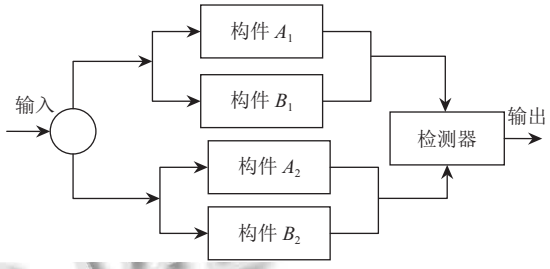

图 5 并联构件双模冗余方式一

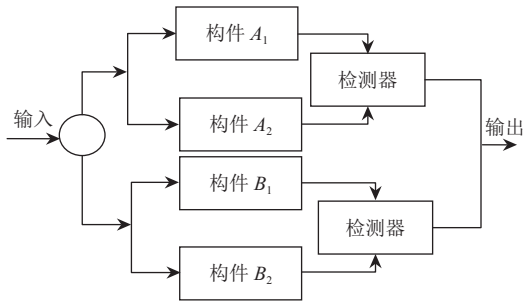

图 6 并联构件双模冗余方式二

若构件 A_1 和构件 A_2 的故障概率为 p_1, 构件 B_1 和构件 B_2 的故障概率为 p_2, 且构件 A_1、构件 A_2、构件 B_1 和构件 B_2 的运行都是独立的. 虽然构件 A_1 和构件 B_1 是并联关系, 但是其中只要一个产生故障, 均认为系统产生故障, 因此构件 A_1 和构件 B_1 并联时产生故障的概率为:

$$1-(1-p_1)(1-p_2) = p_1+p_2-p_1p_2$$

此情况下, 两组并联构件运行结果不一致, 经检测器检测认为出现故障的概率为:

$$1-(1-(p_1+p_2-p_1p_2))^2$$

对于并联构件双模冗余方式二, 先对构件 A_1 和构件 A_2 以及构件 B_1 和构件 B_2 进行冗余处理, 构件 A_1 和构件 A_2 以及构件 B_1 和构件 B_2 的故障检测互不影响, 根据各次检测的结果来确定是那组构件产生故障.

若构件 A_1 和构件 A_2 的故障概率为 p_1, 构件 B_1 和构件 B_2 的故障概率为 p_2, 且构件 A_1、构件 A_2、构件 B_1 和构件 B_2 的运行都是独立的, 则构件 A_1 和构件

A_2 并联时产生故障的概率为:

$$1-(1-p_1)^2=2p_1-{p_1}^2$$

构件 B_1 和构件 B_2 并联时产生故障的概率为:

$$1-(1-p_2)^2=2p_2-{p_2}^2$$

此情况下, 只要一个检测器检测出故障, 均认为出现故障, 其概率为:

$$(2p_1-{p_1}^2)+(2p_2-{p_2}^2)-(2p_1-{p_1}^2)(2p_2-{p_2}^2)$$

4 构件的三取二冗余结构及其扩展

4.1 三取二冗余结构

构件的双模冗余结构只能检测出系统是否出现故障却不能判断具体哪个构件出现故障, 而三取二冗余结构是一种基于三取二表决原理的三模冗余架构, 如图 7 所示.

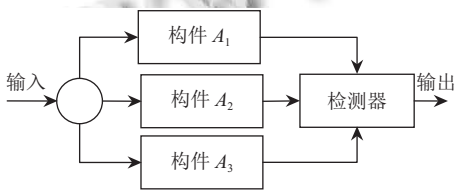

图 7 构件的三取二冗余结构

三取二冗余结构不仅能够检测出系统的故障, 而且能够按照表决原理来确定哪个出现故障, 其检测和处理有 2 种方式.

方式一是先只运行其中的两个冗余构件, 如果运行结果一致则直接运行下一构件; 如果运行结果不一致, 说明至少其中一个构件产生故障, 此时再运行第三个冗余构件, 若三个冗余构件运行结果中有两个结果是相同的, 则把它当成正确结果传输给下一个构件, 否则就认为产生了故障.

方式二是在该结构中, 同时运行三个冗余构件, 当且仅当 2 个以上的构件同时出现故障时 (发生概率较低) 才会认为出现故障, 即三个冗余构件运行结果中, 只要有 2 个及以上结果一致就把该结果当成正确结果传输给下一个构件.

三取二冗余结构不仅可以有效保证冗余构件的退化故障而导致的错误能够被检测出来, 而且在故障产生的情况下判断正确的运行结果. 这一特点与双模冗余结构中的错误检测形成鲜明对比, 如果三个冗余构件中只有一个产生故障, 三取二冗余结构都可以检测出来并给出正确的运行结果, 因此, 三取二冗余结构可以对软件的故障结构进行屏蔽.

在三取二冗余结构中, 假设三个完全冗余的构件 A_1、构件 A_2 和构件 A_3 独立运行, 它们产生故障的概率相同都为 p, 构件 A_1、构件 A_2 和构件 A_3 建立的是并联系统, 则系统运行中检测出故障的概率为$1-(1-p)^3$, 此时出现故障被检测出来的概率将会大幅增加.

三个完全冗余的构件 A_1、构件 A_2 和构件 A_3 运行中, 三取二冗余结构被认为产生故障的概率为 $p^3+3p^2(1-p)$.

经分析可知, 当一个构件产生故障的概率 $p<0.5$时, 三取二冗余结构被认为产生故障的概率 $p^3+3p^2(1-p)$就会小于 p. 一般来说, 一个构件产生故障的概率 p 是一个很小的数, 因此, 三取二冗余结构可以大幅增加系统的可靠性.

4.2 三取二冗余结构的扩展

为了提高系统可靠性, 也可采用四重化冗余结构和二乘三取二冗余结构. 四重化冗余结构主要使用四个完全相同的构件来搭建冗余结构, 通过一定的逻辑关系来使可信性全面提高的一种技术, 其如图 8 所示.

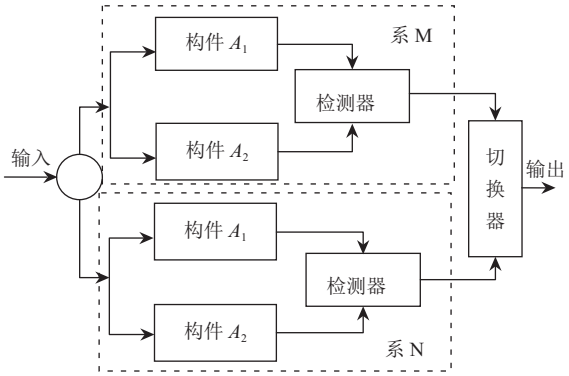

图 8 四重化冗余结构

四个构件被分为两组, 每组有两个构件和一个检测器, 整个四重化冗余结构即由这两个相同的系组成, 每个系的运行与检测方式与双模冗余结构相同. 在系统运行过程中, 只有一系有计算输出而另一系为备用, 当工作的系失效之后, 才进行不同系之间的切换.

二乘三取二冗余结构原理与四重化冗余结构相似, 冗余系统也由两个系组成, 只不过每个系就是一个三取二冗余结构.

四重化冗余结构和二乘三取二冗余结构能够提高系统的可靠性和安全性, 但是大量冗余结构也会使得冗余机制实现起来比较复杂, 增加了系统运行的成本.

对于三取二冗余结构也可以扩展成为 *n*-模冗余的结构, 如图 9 所示. 该结构运行结果的检测和正确运行结果的判定方式与三取二冗余结构相似, 可以采取少数服从多数原则.

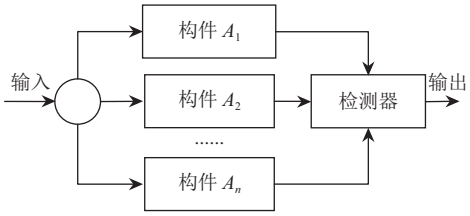

图 9 构件 *n* 模冗余结构

同时, 对于三取二冗余结构和 *n* 模冗余结构都可以考虑构件之间的并串联关系, 但是具体分析方式与上面的原理相似, 故在此就不在重复分析.

在具有冗余机制的可信软件系统中, 可以采用奇偶校验、错误检测、完整性检测和 HASH 函数等方式来对冗余构件运行结果进行检测, 同时根据检测的结果来确定构件在运行过程中是否存在失效问题. 当系统中的一个构件产生故障或者失效时, 可以利用恢复块策略、检查点技术等方法来对其进行替换或者恢复, 从而保证系统的可信性.

5 检验与分析

如果一个基于构件的软件系统中核心构件有两个, 分别为构件 *A* 和构件 *B*, 它们在运行过程中产生故障的概率分别为 p_A=0.3 和 p_B=0.2, 则当对其进行单个构件的双模冗余结构、两个串联构件的双模冗余方式 (方式一、方式二) 以及三取二冗余结构时, 其冗余系统对应的概率如表 1 所示.

从表 1 中可以看出, 在不同构件发生故障的情况下, 单个构件的双模冗余结构和两个串联构件的双模冗余方式虽然不能确定系统中哪个构件产生了故障, 但是能够提高故障概率检测的效果, 而三取二冗余结构能够很好地提高系统的可靠性, 降低系统发生故障的概率.

表 1 不同冗余模式下的概率对应值

冗余方式	概率	
	p_A=0.3	P_B=0.2
单构件双模冗余	0.42	0.32
串联构件的双模(1)	0.6864	0.6864
串联构件的双模(2)	0.6288	0.6668
三取二冗余结构	0.216	0.104

6 结论

可信软件作为计算机软件研究领域最具价值和最具挑战性的核心课题之一, 引起了国内外政府组织、科学界和工业界的高度重视. 我们构建的软件系统不够可信的原因就是故障的存在, 故障的存在说明软件系统内部有缺陷的部件. 为了降低软件故障的发生率, 同时能够检测出系统是否产生故障和确定哪些部件产生故障可以采用系统的容错的方法, 而对系统故障进行容错要依靠冗余. 为此, 本文在基于构件的可信软件结构设计中加入冗余机制, 也就是对可能出现故障的关键和核心构件进行冗余处理, 使得当冗余中的其中一些构件发生失效时, 其他的冗余部件可以继续提供服务, 从而保证软件系统运行的可信性.

参考文献

1 Laprie JC. Dependability: Basic Concepts and Terminology. Vienna: Springe-Verlag, 1991.

2 NSTC. Research challenges in high confidence systems. Proceedings of the Committee on Computing, Information, and Communications Workshop. 1997.

3 陈火旺, 王戟, 董威. 高可信软件工程技术. 电子学报, 2003, 31(A12): 1933–1938.

4 Saha GK. Software based fault tolerance: A survey. Ubiquity, 2006, 7(25): 1–15.

5 Reis GA, Chang J, Vachharajani N, *et al*. Software-controlled fault tolerance. ACM Transactions on Architecture and Code Optimization, 2005, 2(4): 366–396. [doi: 10.1145/1113841]

6 Reis GA, Chang J, Vachharajani N,*et al*. SWIFT: Software implemented fault tolerance. Proceedings of the International Symposium on Code Generation and Optimization. Washington DC, USA. 2005. 243–254.

7 Xie ZP, Sun HY, Saluja K. A survey of software fault tolerance techniques. http://www.pld.ttu.ee/IAF0030/Paper_4.pdf. [2011-05-22].

8 Sathre J, Zambreno J. Automated software attack recovery using rollback and huddle. Design Automation for Embedded Systems, 2008, 12(3): 243–260. [doi: 10.1007/s10617-008-9020-4]

9 何加浪, 张琨, 孟锦, 等. 可进化模块冗余软件混合容错模型. 南京理工大学学报, 2012, 36(2): 272–277, 284.

10 张立众. 一种双总线双控制器软件冗余系统的设计. 陕西理工学院学报 (自然科学版), 2014, 30(3): 41–46.

11 黄涛, 陈祥献, 黄海. 基于三取二冗余结构的安全计算机系统. 计算机工程, 2011, 37(18): 254–257. [doi: 10.3969/j.issn.1000-3428.2011.18.085]

12 杨芙清. 软件工程技术发展思索. 软件学报, 2005, 16(1): 1–7.

13 Atkinson C, Bunse C, Gross HG, *et al*. Component-based Software Development for Embedded Systems. Berlin Heidelberg: Springer-Verlag, 2005.