doi: 10.7690/bgzdh.2016.03.011

基于失效模式的嵌入式软件架构可靠性分析

许丽星，谢　敏，刘　志
（中国工程物理研究院电子工程研究所，四川 绵阳 621900）

摘要：为了能在软件开发早期优化软件架构，提高软件的可靠性，提出一种基于失效模式的软件架构可靠性分析方法——SABRA 分析法。将成熟的软件可靠性工程技术与软件失效模式分析技术相结合，分析故障模式之间的关联关系，根据故障模式的特征识别软件架构中的敏感元素和不可靠元素，对相应的不可靠部分进行容错、避错等设计，并通过建立事件树找到体系结构的薄弱环节。结果表明：该方法简单、有效，能提高整个体系结构的可靠性。

关键词：可靠性分析；失效模式；FMEA
中图分类号：TP309　　**文献标志码**：A

Failure Mode Based Embedded Software Architecture Reliability Analysis

Xu Lixing, Xie Min, Liu Zhi
(*Institute of Electronic Engineering*, *China Academy of Engineering Physics*, *Mianyang* 621900, *China*)

Abstract: In order to optimize software structure and increase software reliability in software early development, based on invalidation mode, propose a software architecture reliability analysis approach, which called SABRA. Combine mature reliability engineering techniques with software invalidation mode analysis technology, analyze correlation among failure modes. According to sensitive element and unreliable element in feature recognition frame of failure mode, carry out fault tolerance and failure avoidance design for corresponding unreliable part, and find out system weak points by establishing event tree. The results show the method is easy and effective to improve system reliability.

Keywords: reliability analysis; failure mode; FMEA

0　引言

随着嵌入式系统功能的日趋复杂以及嵌入式软件在整个系统中实现功能的比例越来越大，软件可靠性成为提高整个嵌入式系统可靠性的瓶颈[1]。软件之所以不可靠，一是与软件中存在的缺陷相关，二是与软件的使用相关。软件缺陷是导致软件失效，影响软件可靠性的根源；因此，为了提高嵌入式软件的可靠性，需要在软件设计过程中对可能发生的失效进行分析，采取必要的措施避免将引起失效的缺陷引入软件[2]。

软件产品中的故障风险可通过故障模式和影响分析（failure modes and effects analysis，FMEA）来估计，而潜在的故障可通过软件故障树分析（fault tree analysis，FTA）识别[3]。软件 FMEA 是分析软件系统中每一个模块、组件所有可能产生的故障模式及其对软件系统造成所有可能影响的一种归纳方法。软件 FTA 是用于表明软件中哪些模块的故障、外部事件或者他们的组合导致软件发生故障的逻辑图。

软件 FMEA 与软件 FTA 2 种技术在单独应用时各有不足：软件 FMEA 是一种自底向上的单因素失效分析方法，无法完善的表达失效原因之间的各种逻辑关系。此外，其分析结果以表格方式列出，不如故障树的图形化表达方式直观。软件 FTA 是一种自顶向下依照树状结构倒推故障原因的方法，选取顶事件时，可能会遗漏潜在的顶层失效事件，有时候这种影响是关键的。另外，软件 FTA 在分析故障原因时也会有所遗漏，这会影响底事件的重要度排序，从而影响实施改进措施时对轻重缓急的判断。树型结构在描述分析方面也不如 FMEA 详尽。

因此，笔者提出一种软件架构可靠性分析方法（software architecture-based reliability analysis，SABRA），将 FMEA 与 FTA 2 种技术综合起来，从分析对象的失效模式出发，根据失效的因果逻辑关系建立事件树（event tree，ET），进而分析软件系统架构的薄弱环节，优化系统架构。

1　顶层分析

FMEA 架构范围始于系统功能级别最顶端，但可在整个系统层次中通过各种方式递归地应用到组件级别。为了理解和预计系统架构设计的质量需求，Bachman[4]定义了 4 个重要的要求：1) 有详细的可靠性需求的需求规格说明；2) 为达到可靠性需求的架构决策；3) 将架构决策与可靠性需求联系起来；4) 将决策实现为设计。

GB/T—11457 标准对失效的定义是清楚和全面

收稿日期：2015-12-02；修回日期：2015-12-30
基金项目：中国工程物理研究院质量与可靠性共性技术研究课题（S2014ZK）
作者简介：许丽星（1981—），女，山东人，硕士，助理研究员，从事软件工程、软件研发研究。

的。失效(failure)是由软件的错误和故障引起的。软件失效泛指程序在运行中丧失了全部或部分功能，出现偏离预期的正常状态的事件。错误(error)是指开发人员在开发过程中出现的失误、疏忽和错误。故障(fault)是指软件在运行过程中出现的一种不希望或不可接受的内部状态，通常是由于软件缺陷在运行时引起并产生的错误状态。

SABRA 方法的活动图如图 1 所示，分为 3 个基本过程：定义、分析和调整。

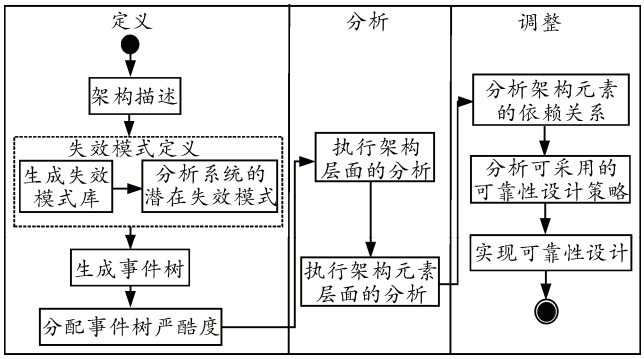


图 1　SABRA 方法活动图

1) 定义：定义系统架构、失效模式、事件树及其失效严酷度等级。

2) 分析：以定义阶段作为输入，开展架构层面的分析和架构元素层面的分析。

3) 调整：根据前面的分析结果和架构元素间的依赖关系，分析可采用的可靠性设计策略，实现可靠性设计，调整系统架构，增强系统可靠性。

2　系统架构和失效模式定义

2.1　系统定义

系统定义的目的是确定软件 FMEA 分析的对象，描述系统的功能，确定分析对象。SABRA 方法中确定分析对象为功能模块。以某飞控系统软件为例，系统主要功能图如图 2 所示。该飞控软件运行于飞控计算机内，完成飞机自动飞行控制系统的系统管理和控制律计算等功能。

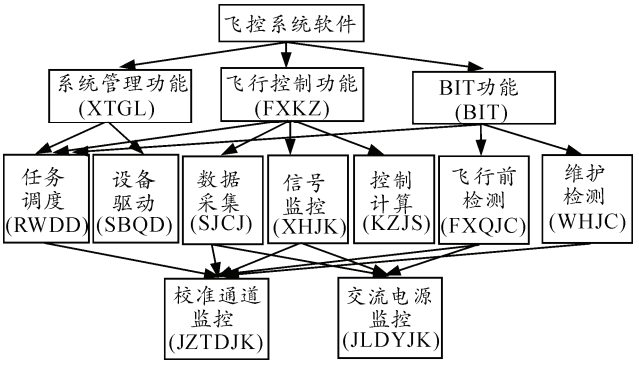


图 2　某飞控软件功能层次

2.2　确定分析对象的失效模式

根据给定的软件运行条件，可能引起软件潜在失效的内部或外部原因就是软件失效模式。软件失效模式模板如表 1 所示。

表 1　失效模式描述示例

名称	含义
FID	失效模式 ID
AEID	发生失效的体系结构元素的字母缩写
故障(fault)	失效原因，描述原因和其特性
错误(error)	描述引起错误体系结构元素的状态或条件
失效(failure)	失效影响

笔者基于 FMEA 方法，并做出部分修改的软件失效模式模板如表 1 所示。FMEA 中定义了失效模式的 5 个属性：失效 ID、相关组件(或模块)、失效原因、失效模式和失效影响。软件失效模式(software failure mode)是指软件失效发生的不同方式。软件失效影响(software failure effect)是指软件失效模式对软件系统的运行、功能和状态等造成的后果。表 1 所示的模板中，故障、错误和失效分别代表了失效原因、失效模式和失效影响。

确定被分析对象的失效模式包含 2 步[5]：1) 确定相关失效的领域模型；2) 从领域模型中得到被分析对象的失效模式。

2.2.1　定义失效模式库

只有将被分析对象的所有可能失效模式尽可能全面地分析出来，才能采取相应措施改进设计，防止失效现象的发生；因此，失效模式的分析是否全面合理决定了软件 FMEA 的分析效果，是整个分析过程中最为关键的一步。

表 2　飞控系统失效模式库

名称	属性	描述示例
故障	来源	内因、外因、其他元素
	特征	硬件、软件
	瞬时/永久性	永久、瞬时
错误	类型	值错误、死锁、错误的执行路径、时间太早/太晚、数据越界、未进行条件测试
	可检测性	可检测、不可检测
	可修复性	可修复、不可修复
失效	类型	时间、行为、错误解析、错误的呈现
	目标	系统失效、其他组件失效

在进行失效模式分析时，如果单凭分析人员的经验，难免容易由于人为因素造成遗漏，这将影响分析的效果。在工程实践中，人们常常将以往的分析经验和积累失效案例加以总结，归纳出软件的失效模式。分析人员根据被分析软件的具体特点选择适用的失效模式，可以在很大程度上避免由于分析人员经验不足造成的遗漏。随着人们对软件失效研究的进一步深入，会丰富、完善现有的失效模式，

从而更加有效地指导分析工作。

针对飞控软件，其失效模式库如表2。下面要做的工作就是针对每一个被分析对象确定其潜在的失效模式(例如：相应时间超时或者输出错误值等)。

2.2.2 分析系统的潜在失效模式

从失效模式库中分析特定系统潜在的失效模式，并为架构中的每个元素定义相关的描述。表3给出了飞控系统软件的失效模式，限于篇幅，这里只描述9个失效模式。架构元素的类型与使用的架构角度相关[6]。从开发的角度，架构元素就是功能模块；从组件和连接件的角度来说，架构元素就是组件和连接件。文中所说的架构元素定位在模块。

表3 飞控系统软件失效模式及其特性

FID	AEID	故障	错误	失效
F1	XHJK	描述：读取超时标志位的等待时间过短，即读取发生在硬件置位前 来源：JZTDJK(F4)或 FXQJC (F6) 特征：软件 永久/瞬时：永久	超时判断错误 类型：值错误 可检测性：可检测 可恢复性：可恢复	描述：得出错误的表决结果 类型：行为 目标：系统失效
F2	XHJK	描述：写入数据位置错误 来源：FXQJC (F5) 特征：软件 永久/瞬时：永久	采集通道标志位置错误 类型：错误的路径 可检测性：可检测 可恢复性：不可恢复	描述：导致飞控系统全部功能失效 类型：行为 目标：JLDYJK(F3)
F3	JLDYJK	描述：测量信号存在干扰 来源：XHJK(F2) 特征：软件 永久/瞬时：瞬时	采集值偶然误差过大 类型：值错误 可检测性：可检测 可恢复性：可恢复	描述：导致飞控系统全部功能失效 类型：行为 目标：系统失效
F4	JZTDJK	描述：软件计数错误 来源：内部 特征：软件 永久/瞬时：永久	采集通道电压值错误 类型：值错误 可检测性：可检测 可恢复性：可恢复	描述：监控通道状态误判 类型：行为 目标：XHJK(F1)
F5	FXQJC	描述：协议错误 来源：外部 特征：软件 永久/瞬时：永久	不能与相连的设备进行通信 类型：太早/太晚 可检测性：可检测 可恢复性：可恢复	描述：无法发出监控的状态信息 类型：时间 目标：XHJK(F2)
F6	FXQJC	描述：软件错误 来源：外部 特征：软件 永久/瞬时：永久	信号含义解析错误 类型：值错误 可检测性：不可检测 可恢复性：不可恢复	描述：发出无效指令 类型：错误解析 目标：XHJK(F1)
F7	WHJC	描述：测量信号存在干扰 来源：外部 特征：软件 永久/瞬时：瞬时	采集值偶然误差过大 类型：值错误 可检测性：可检测 可恢复性：可恢复	描述：无法发出有效数据 类型：错误解析 目标：BIT(F8)
F8	BIT	描述：输出数据类型为 float 来源：WHJC (F7)与 RWDD(F9) 特征：软件 永久/瞬时：瞬时	输出数据精度误差 类型：值错误 可检测性：不可检测 可恢复性：可恢复	描述：经过迭代计算的结果错误 类型：行为 目标：系统失效
F9	RWDD	描述：软件错误 来源：内部 特征：软件 永久/瞬时：永久	输出参数不完全 类型：值错误 可检测性：可检测 可恢复性：不可恢复	描述：状态参数不精确 类型：错误值 目标：BIT(F8)

表3所示的失效模式表格增加了对故障、错误和失效特征的描述。特征就是这些描述的关键词，将表格的每一行联系起来。比如，在故障模式 F2 中，故障的来源为 F5，这就是说故障 F5 的发生会导致故障 F2 的发生。故障的产生也可是多因素的，故障模式 F1 的发生就是 F4 和 F6 共同作用的结果。

2.3 生成事件树

从对失效模式的深入研究发现，失效模式之间是有关联的，事故的出现不是无缘无故的，是许多原因事件相继发生造成的。其中某一事件的发生会引起另一些事件的出现；而一些事件的发生是另一些事件首先发生为前提条件。事件树分析法是一种时序逻辑的事故分析方法，它发起于某一初因事件，按照事故的发展演变时序，分阶段一步一步地推进。每一事件可能的后续事件只能取完全对立的2种状态之一(正常或故障、成功或失败、安全或危险等)，逐步向后果方向演变，直到达到系统故障或事故发生为止[7-9]。

一个事件树表示故障和失效的因果关系，事件树的根表示一个失效，叶子表示故障。因为一个失效可以是多个故障触发，事件树的节点就通过这些逻辑关系关联起来。通常，事件树是一系列故障树组成的图 $G(V, E)$，V 是图的顶点，E 为有向边 (u,v)，

$u,v \in V$。$V=F \cup A$，F 是失效模式的集合，每个失效模式都与一个模块相关；F_u 是失效模式集合的根节点事件，$F_u \in F$；A 表示逻辑关系门，显然 $A=A_{\text{AND}} \cup A_{\text{OR}}$，对 $\forall g \in A$，出度$(g)=1 \cap$ 入度$(g) \geqslant 1$。

对于飞行控制系统，从表 3 所示的失效模式可以得到如图 3 所示的事件树。事件树由 3 棵故障树组成，其中椭圆虚线框中所示的元素为事件树根节点故障集。

2.4　定义事件树的严酷度等级

软件失效影响严酷度(severity)指软件失效模式所产生后果的严重程度。

软件失效影响及其严酷度的确定可以为失效模式改进措施的制定提供依据。对于产生后果越严重的失效模式，越应该采取有效措施加以改进，以避免危险事件的发生。

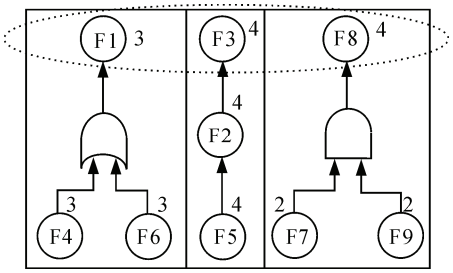


图 3　从表 3 的失效模式获得的事件树

对军事装备系统和有安全风险的系统来说，通常按照人身伤害程度对失效进行分类，如表 4 所示。

表 4　武器装备常用的严酷度类别及定义

严酷度	严酷度值	描述
I 类(灾难的)	4	引起人员死亡或产品(如飞机、坦克、导弹及船舶等)毁坏、重大环境损害
II 类(致命的)	3	引起人员的严重伤害或重大经济损失或导致任务失败、产品严重损坏及严重环境损害
III 类(中等的)	2	引起人员的中等程度伤害或中等程度的经济损失或导致任务延误或降级、产品中等程度的损坏及环境损害
IV 类(轻度的)	1	不足以导致人员伤害或轻度的经济损失或产品轻度的损坏及环境损害，但它会导致非计划性维护或修理

通常，对 FTA 根节点的概率取决于叶子节点的失效概率[8]。在计算叶子节点的失效概率前，先根据事件树根节点对系统整体造成的严重后果分配严酷度值。图 3 中失效模式旁边的数字就表示该失效模式的严酷度值。

叶子节点的严酷度值由根节点的严酷度值决定，S_u 表示根节点的严酷度值，$\forall f \in F$ 且 $f \notin F_u$，计算公式如下式：

$$s(f)=\sum_{\substack{\forall v s.t \\ (f,v)\in E}} s(v)\times P(v \mid f)。\tag{1}$$

其中 $P(v \mid f)$ 表示当 f 发生时 v 发生的概率。用这个值乘以 v 的严酷度去计算 f 的严酷度。根据概率学理论，$P(v \mid f)=P(v \cap f)/P(f)$。如果 v 是或门，那么当 f 发生时 v 的输出一定会发生，这就是说 $P(v \cap f)=P(f)$，那么 $P(v \mid f)=1$。如果 v 是与门，对所有与 v 相连的 x 来说，$P(v \cap f)=\prod P(x)$，要计算 $P(v \mid f)$ 就要知道所有的 $P(x)$，这里的 x 不包括 f。

对飞控系统软件来说，F8 被分配的严酷度值为 4，F7 和 F9 之间是与门，那么：

$$s(F_7)=s(F_8)\times P(F_8 \mid F_7)=4\times P(F_9)；$$

$$s(F_9)=s(F_8)\times P(F_8 \mid F_9)=4\times P(F_7)。$$

由此可以看出，要获得每个失效模式的严酷度值需要知道每一个失效模式发生的概率。理论上有 3 种方法去获得这个概率：

1) 使用固定值：设每种失效具有相同的权重，那么失效模式发生的概率就由与门或者或门决定。

2) 假设分析：基于一定的假设，考虑每种失效模式的发生概率。根据假设条件的不同，每种失效模式的发生概率有所不同。

3) 测量真实的失效概率：根据历史数据或者测试获得失效模式发生的真实概率。

如果不知道失效模式发生的概率，或者该失效模式对于最终的输出没有相关影响，可以考虑使用固定概率。对于没有任何历史信息的系统，假设分析非常有用。第 3 种方式是获得严酷度值最精确的方式，但是历史数据通常会丢失或者很难获得。

文中采用固定概率 0.5，严酷度分配[10]如下：

$$s(f)=\sum_{\substack{(u,v)\in E \\ v\in F \wedge v\in A_{\text{OR}}}} s(v)+\sum_{\substack{(u,v)\in E \\ v\in A_{\text{AND}}}} s(v)\times p^{\text{入度}(v)-1}。\tag{2}$$

3　软件结构分析

3.1　架构层面的分析

分析过程的第一步是架构层面的分析，根据可靠性确定架构的关键模块。这里笔者考虑 2 种模块：不可靠模块和敏感模块。不可靠模块是成为失效场景中大部分来源的模块；敏感模块是与大部分的失效场景都有关的模块，包括由内部故障引起的失效、其他因素引起的失效。

采用文献[3]推荐的失效比例 PF(percentage of

failures)方法来确定架构的关键模块。对每一个模块 c 来说，PF 的计算如下式：

$$\mathrm{PF}_c = \frac{\text{与}c\text{相关的失效数}}{\text{失效总数}} \times 100\%。 \tag{3}$$

PF 将所有的失效等同对待。飞控系统各模块的失效比例图如图 4 所示。显然，模块 XHJK 和模块 FXKZ 所占的比例远远高出其他模块。

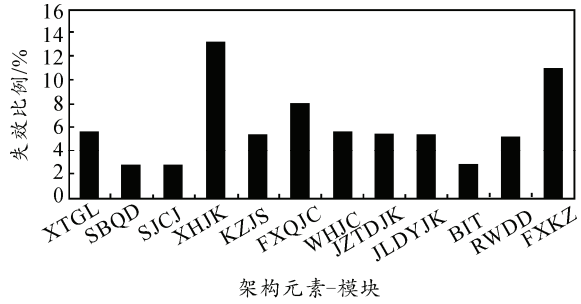


图 4 架构中模块的失效模式比例

然而，模块本身是不同的，考虑失效的严酷度等级，定义一种加权的失效比例 WPF，如下式：

$$\mathrm{WPF}_c = \frac{\sum\limits_{\substack{\forall u \in F \\ \mathrm{AEID}(u)=c}} s(u)}{\sum\limits_{\forall u \in F} s(u)} \times 100\%。 \tag{4}$$

搜集每个模块相关的失效模式并将其严酷度值相加，各模块的 WPF 如图 5 所示。

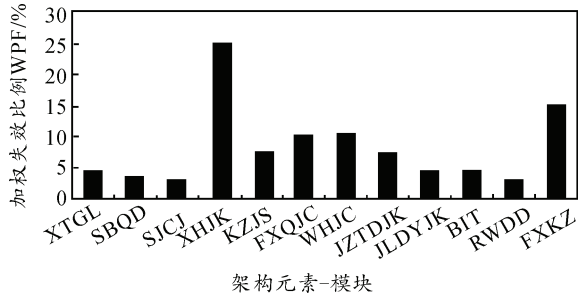


图 5 架构中模块的加权失效模式比例

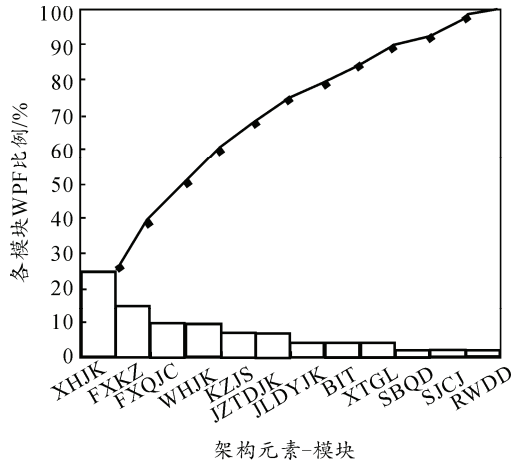


图 6 按照 WPF 分组的帕累托

虽然图 4 和图 5 中各模块的比例不同，但模块 XHJK 和模块 FXKZ 的比例仍然是最高的。按照 WPF 的结果对架构模块采用降序排列，使用帕累托图(Pareto)表示(如图 6 所示)，从图中可以更直观地看出：模块 XHJK 和 FXKZ 占总模块数的 16.7%，WPF 值却占到 40%，即这 2 个模块是进行可靠性分析的重点。

3.2 结构元素层面的分析

从系统架构层面的分析可以量化架构中模块失效模式对系统的影响，如从前面对飞控系统的分析，可以得到模块 XHJK 和 FXKZ 对整个系统有很大的影响；但是这 2 个模块会怎样影响整个系统的可靠性，就需要对这 2 个模块的失效模式做进一步分析。关注每种失效模式的特征正是结构元素层面分析的工作。

将 2 个模块对应的每种失效模式按照特征进行分类，如图 7 和图 8 所示。从图中可以很清晰地看出：导致模块 XHJK 失效的原因大多都是其他模块引起的，导致模块 FXKZ 失效的原因是其自身和其他模块。这些信息对于后面调整系统架构非常有用。

4 架构调整

根据前面的分析进行结构调整需要经过 3 个步骤：1) 定义与失效相关的模块；2) 识别架构策略；3) 应用架构策略。

4.1 分析结构模块间的依赖关系

软件架构策略通常被看作是支持软件质量属性的特征[4]。SABRA 方法中，应用这个概念来得到支持可靠性的结构设计决策。

通常，要对一个模块使用恢复策略也要考虑与其相关的模块，因为对单个模块的处理通常会影响到与其相关的模块。为此，首先要分析模块间的依赖关系，这也定义了设计决策应用的边界。表 5 为飞控系统软件模块的依赖关系。

表 5 飞控系统软件模块依赖关系

AEID	具有依赖关系的模块
XTGL	RWDD,SBQD,FXKZ,XHJK,JZTDJK
SBQD	XTGL,FXQJC
SJCJ	FXKZ
XHJK	FXKZ,JZTDJK,JLDYJK,XTGL,BIT,FXQJC,WHJC
KZJS	FXKZ,RWDD
FXQJC	BIT
WHJC	BIT
JZTDJK	XHJK,SJCJ,FXKZ,RWDD
JLDYJK	XHJK,RWDD
BIT	FXQJC,WHJC
RWDD	XTGL
FXKZ	SJCJ,XHJK,KZJS,JZTDJK,JLDYJK

4.2　识别架构策略

识别了敏感模块、模块间依赖关系和潜在的失效模式，笔者继续识别关于可靠性的结构策略。

软件缺陷是导致软件失效、影响软件可靠性的根源。从定性意义上讲，软件产品中的缺陷数越多，缺陷被触发导致失效的可能性越大，可靠性越低。因此，为了保证软件可靠性，可以有 2 种设计途径：

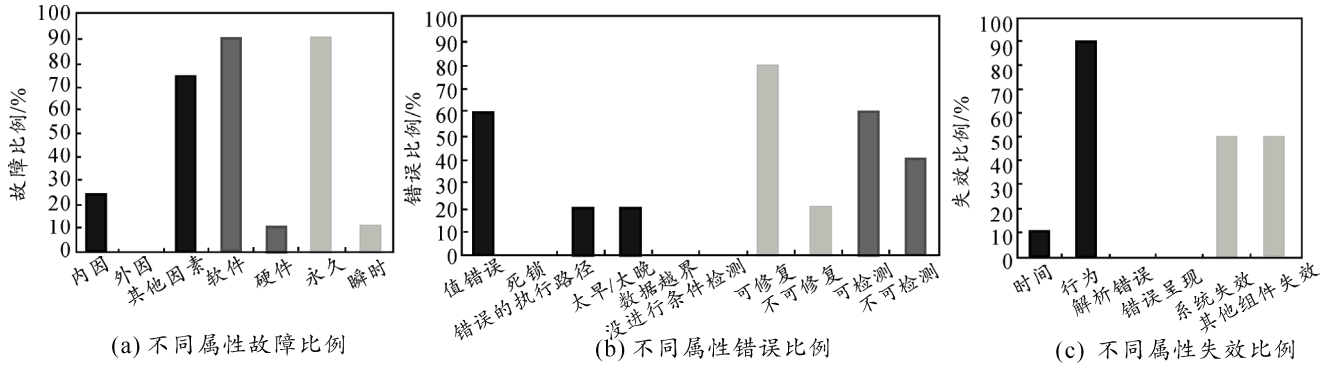
1) 避免引入缺陷；

2) 万一引入缺陷，避免因缺陷导致失效。

第 1 种途径体现了软件可靠性设计的避错设计思想，第 2 种途径是软件可靠性设计中的容错设计。

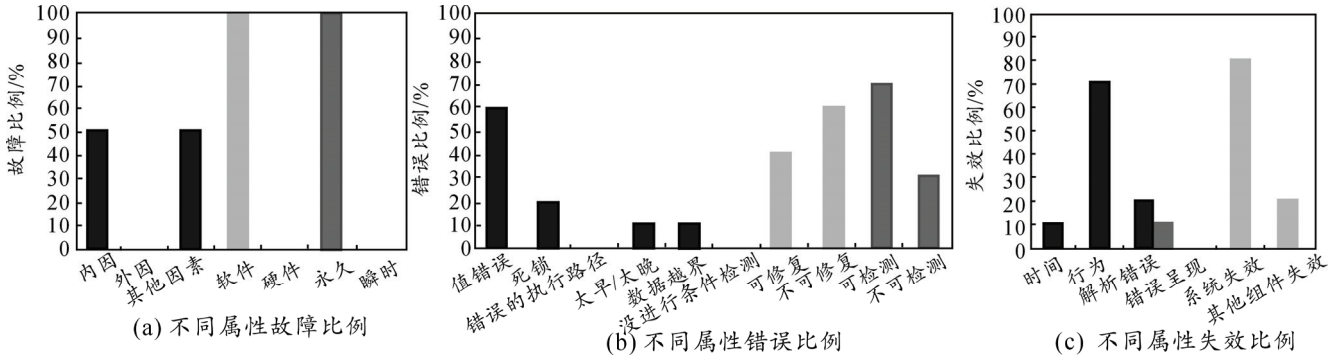
从 SABRA 的分析中可以获得系统潜在的失效模式，可视作是缺陷预测。这是缺陷预防、容错和缺陷消除的基础。笔者重点关注容错，即在系统存在故障的情况下，发现并纠正故障使计算机系统不受到影响继续正确运行，或将故障降到可接受范畴的技术。

识别了敏感模块后，笔者分析与这些敏感模块相关联的故障、错误特征，如图 7 和图 8 所示。故障分为内在故障和外在故障 2 大类。



(a) 不同属性故障比例　(b) 不同属性错误比例　(c) 不同属性失效比例

图 7　模块 XHJK 故障、错误和失效特性图



(a) 不同属性故障比例　(b) 不同属性错误比例　(c) 不同属性失效比例

图 8　模块 FXKZ 故障、错误和失效特性

1) 内在故障，指由于软件的设计(需求分析及设计)、生产(编程)过程中考虑不周等人为错误而导致软件本身的故障，这些故障在系统中使软件发生失效，这一类错误具有永久性、重复性和不可恢复的特征。

2) 外在故障，指由于外界因素导致系统存在故障，多为硬件错误等环境因素影响所致，具有瞬态、偶发性和可恢复的特征。

根据以上定义可知：如果是内在故障，说明敏感模块本身就是不可靠的，这就要求敏感模块本身要进行容错设计；如果是外部故障，说明模块失效是由自身之外的因素导致，就需要对这些外因进行容错，这些不可靠模块就可以通过结构模块依赖关系表和事件树找到；如果故障是瞬时的，虽然可能不会引起失效，仍然需要采取容错措施。

容错分为故障检测、故障诊断和故障处理。表 6 列举了部分容错措施适用的情况。

表 6　容错措施

容错类型	容错措施
故障检测	看门狗(错误类型=死锁) N 版本编程(错误类型=值错误；错误来源=内部故障；永久性错误) 在线监控(错误类型=值错误或错误的执行路径)
故障恢复	N 版本编程技术(错误类型=值错误；错误来源=内部故障；永久性错误) 重启(错误类型=死锁，瞬时错误)

4.3　应用结构策略

结构调整的最后一步就是应用结构策略。前面的分析得出了可以应用的策略，考虑费用和系统其他的约束(资源约束等)后，可以清晰地确定特定系统的设计策略。表 7 为飞控系统软件模块 XHJK 和 FXKZ 的容错改进措施。

(下转第 53 页)