软件可靠性
专家系统(SRES)开发

徐 仁 佐 著

清华大学出版社

(京)新登字 158 号

内 容 简 介

本书全面系统地讨论了软件可靠性工程中,软件可靠性模型应用的不一致性,以及由此产生的不良影响。为克服这不良影响,作者开发出软件可靠性专家系统——SRES。书中对 SRES 开发的各个方面的问题:软件可靠性模型的选择、比较判优、它们的参数估计技术、专家系统开发工具 CLIPS 的选用及性能、推理程序的运行过程及结论、软件可靠性专家系统 SRES 的使用及操作等,都作了详尽的描述。

本书适用于从事软件可靠性工程的研究与应用人员,也适用于从事计算机软件工程、人工智能技术及相近专业大专院校师生。

图书在版编目(CIP)数据

软件可靠性专家系统(SRES)开发/徐仁佐著. —北京:清华大学出版社,1996.10
ISBN 7-302-01844-8

Ⅰ.软… Ⅱ.徐… Ⅲ.软件可靠性-专家系统-软件开发 Ⅳ.TP311.5

中国版本图书馆 CIP 数据核字(96)第 20337 号

出版者:清华大学出版社(北京清华大学校内,邮编 100084)
印刷者:北京市清华园胶印厂
发行者:新华书店总店北京科技发行所
开 本:787×1092 1/16 印张:8.75 字数:200 千字
版 次:1996 年 11 月第 1 版 1996 年 11 月第 1 次印刷
书 号:ISBN 7-302-01844-8/TP·829
印 数:0001—3000
定 价:12.00 元

谨以此书纪念

含辛茹苦哺育我们长大成人，

却又过早地离开了我们的母亲

DEVELOPMENT OF SOFTWARE RELIABILITY EXPERT SYSTEM——SRES

XU REN—ZUO

SUMMARY

Development of Software Reliability Expert System——SRES introduces the inconsistency in applications of software reliability models in the field of Software Reliability Engineering and the harmful effects caused by this character. To overcome these harmful effects, author developed the Software Reliability Expert System——SRES under supporting of the National Nature and Science of China. In this book, author describes various respects in detail, they include: selection of software reliability models, compariting among them, judging what is good and what is bad, the parameter estimation techniques of these models, expert system development tool——CLIPS, the execution process of reasoning program in SRES, the way of using and operating of SRES, and so on.

This book was written to give practical help to the researchers and users who are doing jobs in the field of Softwae Reliability Engineering. In addition, this book is suitable for reading for teachers and students and graduated ones in the field of Computer Software Engineering and Artificial Intelligence. Especially, this book is indispensable one to who are doing jobs of software development and software quality management.

前　言

还在作者于 1989 年 6 月以访问学者身份，访问瑞典 Linköping 大学期间，作者就与 Bo Bergman 教授对软件可靠性工程技术中软件可靠性模型应用的不一致性(The Inconsistency of Applications of Software Reliability Models)进行深入的讨论。当时我们就认为，该不一致性是严重制约该技术推广应用的因素。回国后，作者即着手研究克服不一致性的问题。

用户既然受不一致性的困扰，对于如何选用软件可靠性模型感觉到棘手，针对这一点，我们可应用人工智能的成熟技术，开发一个专家系统，以帮助用户来较顺利地选用模型，从而在一定程度上缓解由于不一致性的消极影响。这就是我们的出发点。

感谢中国国家自然科学基金委员会的资助，使我们的这一想法终于有条件予以实现。本书就是我们在开发软件可靠性专家系统 SRES 过程中的经验总结，希望对读者会有所帮助，以促进我国软件可靠性工程技术的发展与推广应用。

要开发软件可靠性专家系统，有几个方面的问题要解决好：第一，面对如此众多的软件可靠性模型，怎样进行选择？经过研究，我们认为应选那些具有典型代表性、成熟度高、可应用性好的模型。但有一点要声明的，由于我们的水平所限，不一定选上的软件可靠性模型就一定好，没有选上的就一定不好。正是基于此，我们在系统中留有接口，以便将来经过实践检验，可以适时地增减软件可靠性模型。第二，选用什么样的专家系统开发工具？经过比较对照，我们选定荣获美国 NASA 航天科技大奖的 CLIPS(C Language Implementation Production System)。这是由于它的优异性能、高效推理、便于系统集成等方面的原因，使我们选定了它。第三，对于软件可靠性模型，特别是异类模型，如何选用工程上直观、简便的标准，对它们进行比较与判优？为此我们采用拟合率值(The Value of Fitting Rate)作为比较的标准。第四，为了工程上使用的方便，用户学习掌握的快速，我们决定采用“界面友好、直观可视、操作简便、易于掌握”的十六字方针，在屏幕上显示出直观的拟合图形以及分析的数据结果。

在软件可靠性专家系统 SRES 开发过程的始终，我们坚持一个原则：“一切让数据说明问题”。这样就对使用的软件故障数据的质量要求很高，但我们相信，只要严格软件工程的管理，认真做好数据收集工作，这个目标是可以达到的。“世界上怕就怕认真二字”。

在本书中，我们将就这些问题逐一展开讨论。为便于读者进行研究，参考文献放在每节之后。

在本系统的开发过程中，我们得到许多单位的支持与合作，在我们的科研工作中，得到武汉大学数学系胡泽民教授、武汉大学物理系刘莲君副教授以及武汉大学数学系刘妍岩同志的大力支持与协作，我的研究生张健在 SRES 的实现中做了大量创造性的工作，在此一并致谢。在本书的撰写及定稿的录入、照排过程中，得到武汉大学计算机软件工程国家重点实验室刘和安、何平二位同志的大力帮助，在此向他们表示感谢。

由于作者的水平所限，书中肯定存在缺点与错误，敬请读者批评指正。

徐仁佐

1996 年 4 月于珞珈山

目　录

0 总　论

软件可靠性工程，是对软件的质量（特别是软件的可靠性）进行控制与管理的实用性学科，同时，它又是一门年青的、正在形成与发展中的新兴学科。它要研究的问题很多，主要集中于：软件工程中对软件开发过程的管理与监控、软件开发过程中的质量（特别是软件的安全性与可靠性）保证技术、软件质量量化指标及测度建立与应用、软件质量定量化模型的建立与应用、软件开发进度的监控技术与手段、软件开发的进度与成本之间互相制约又互相促进的关系的研究、软件开发过程中人的因素的研究，等等。

软件可靠性定量估测模型的建立与应用的研究，至今已有二十多年了。时至今日，作为这一领域的积极成果之一，就是建立了一批可以用于实际工程的软件可靠性模型。虽然建模中使用了许多领域的知识，但是作为其中起主要作用的仍然是统计方法。目前许多可供实际使用的软件可靠性模型，其中最引人注目的共同特点之一，就是每个模型的理论基础都建立在各个不同的统计假设之上。这些假设界定了具体模型关于软件故障行为的概率方式，同时也限定了每个模型对软件行为故障的不同描述方式，和软件开发人员与查错人员使用的测试与排错的不同方法。这些假设是每个软件可靠性模型的理论基础，也同时界定了每个模型的不同适应范围。因此，也同时决定了它们被应用的广度与深度。

首先，软件可靠性模型必须应用于大型的软件开发项目。而且这些开发项目，多半是国防上的、工业生产上的大型计算机应用项目。这里，“大型”的涵义是：数量大，这反映在开发出来的源代码行数，不是那些几百条、几千条、甚至上万条软件源代码的小程序所能望其项背的。并且，这样的软件逻辑结构复杂多变，随着输入信息的变化，其中的内部状态、中间状态组合呈现出来的是天文数字，其极端的情况是不可枚举的。其次，这些软件项目，多半是用于对尖端武器的控制，工业生产上的关键性设备的监控与管理，其监控对象多数具有对安全性、可靠性要求极高的特性。例如：对一个核电站的监控软件，宇宙飞船的生命保障系统中所使用的电子计算机的软件系统，等等。这些设备与装置，多数价格昂贵，一旦发生故障，往往损失惨重，甚至以牺牲生命作为代价。

现在，世界上各个国家都拥有大量这样的装备，因此，对它们的安全性与可靠性的要求，也就自然而然地提出来了。以前，由于对软件质量的认识不足，片面地认为，只有硬件设备才有可靠性问题，软件是不存在可靠性问题的。也就是说，许多人片面地将软件的可靠性潜意识地、武断地论断为100%！直到由于软件故障所产生的损失给了人们以惨重的教训以后，许多人才猛然醒悟了：“啊！原来软件也有可靠性问题。”

现在的问题在于：既然软件可靠性也是这么重要，目前又有许多可供选择的软件可靠性模型可用来评估软件可靠性，那么好吧，我们就“随便”选几个模型来，看看我们开发的软件的可靠性指标达到什么水平了。这里反映出来的又是一个认识上的问题。一方面，确实有着不少可供任意选用的软件可靠性模型；另一方面，软件可靠性又是如此重要。于是，上面提到的想法就自然而然地产生了。但是，由这一认识所暴露出来的更深层的问题在哪

里呢？这一深层次的问题就在于：对于现有软件可靠性模型的选择中的随意性。这种随意性的根源就在于：既然这软件可靠性模型的建模基础就是统计理论，只要“随便”选用其中的任一个，运用正确的统计方法，总可以得出一些结果来。但是，如果进一步问一下：这个结果的可信度是多少？它们是否准确反映出了被评估软件系统的正确状况？另外，如果再任选别的模型，也可以得出另一些结果，那么，我们究竟应相信哪一个模型的结果才是真正反映了软件系统的准确状况呢？

软件可靠性工程发展到今天，束缚、限制软件可靠性模型应用的症结就在于此。我们将其归结为一个词：不一致性。正是因为存在着软件可靠性评估中的不一致性，才导致了软件可靠性模型选择上的随意性。

软件可靠性评估中的不一致性主要包含两个方面的问题。第一，由于每个软件可靠性模型(Software reliability model——SRM)，都具有特定的理论假设(比如，对软件发生故障的随机行为的描述等)，现有模型已将目前数理统计中常用的随机分布都用上了，并且对于软件开发(特别是软件测试阶段)所收集的数据，运用正确的统计方法加以处理，总可以得到一些结果。选用不同的模型会产生不同的结果，由此产生的问题就是软件可靠性评估的不一致性。第二，在软件进入系统测试阶段，随着软件测试的深入进展，收集到的故障数据也在不断地增多。在测试某时刻 T_1，进行软件可靠性评估，可能选用了模型 M_1；但经过某 Δt，到 T_2 时刻($T_2>T_1$)，又进行一次评估，可能会选同一个模型 M_1，也可能会选用另一个模型 M_2。这样就产生了另一种意义下的不一致性的问题。

目前，已发表的软件可靠性模型不下百种之多，但是，对于应用其中的某些模型进行过软件项目的软件可靠性分析的软件工程师们、系统设计师们来说，一直困扰他们的一个问题就是：“既然许多模型都可以应用于软件工程实践，到底哪个模型的结果更可信？”其实，这就是软件可靠性模型应用中的不一致性的一个情况。另一方面，在软件系统测试的过程中，收集到一定的数据之后，即可进行软件可靠性分析。此时，选哪个模型才能得到“最佳”结果？随着软件系统测试的深入进行，收集到的软件故障数据也会不断增加。然后在某一适当时刻，又可进行软件可靠性分析，这时，是选与上次分析时所用的同一模型，还是重新选用另外的模型呢？这是软件可靠性模型应用中的不一致性的另一表现形式。这在我们介绍软件可靠性模型的书中，已有一些讨论。

通过以上的分析，我们可以将软件可靠性模型应用中的不一致性，归结为软件可靠性模型的选用问题。因此，我们给出软件可靠性模型应用中的不一致性定义如下：

定义：在进行软件可靠性分析时，由于选用不同的软件可靠性模型，所得出的分析结果不同，致使用户无所适从，这种现象，就称为软件可靠性模型应用中的不一致性。

在进行不一致性研究的过程中，我们选用了两个软件故障数据集合，分别记为 DS1 和 DS2，DS1 为完全数据，DS2 为不完全数据，DS1 即为 NTDS 数据，DS2 为我们在开发某软件项目时收集的一个软件故障数据集合 D. inc。关于数据，我们在下一章讨论。

为了进一步研究不一致性，我们对于 DS1，分别选用了 Goel-Okumoto 模型(记为 GO)，Yamada-Osaki 模型(记为 YO)，Goel 的三参数 NHPP 模型(记为 Goel(3))，以及 Littlewood-Verrall 的贝叶斯模型(记为 LDM，LVLM，LVQM)。对于 DS2，分别选用了 Weibull 模型(记为 WM)，GO，YO，Goel(3)，以及应用 EM 算法对参数值进行调整的

NHP3AD 模型。

通过对以上两类数据，以及所选用的模型，我们可以进一步来分析软件可靠性模型应用中的不一致性的严重程度。表 0.1 为应用相应软件可靠性模型对 DS1 进行分析的结果。

表 0.1　应用软件可靠性模型对软件故障数据 DS1 的分析结果

模型名	MTBF	R(MTBF)	$R(\Delta t=1)$	$R(\Delta t=84.9)$	F_r
GO			0.9964	0.9701	0.1707
YO			0.9991	0.9930	0.1309
Goel(3)			0.9679	0.7585	0.1758
LDM	24.9792	0.3679	0.9608	0.7119	0.1850
LVLM	66.9090	0.2500	0.9708	0.7875	0.1264
LVQM	81.2562	0.2500	0.9758	0.8197	0.1423

表 0.2 为应用相应软件可靠性模型对 DS2 进行分析的结果：

表 0.2　应用软件可靠性模型对软件故障数据 DS2 的分析结果

模型名	MTBF	R(MTBF)	$R(\Delta t=1)$	$R(\Delta t=3.5)$	F_r
WM	3.4761	0.9409	0.9862	0.9404	0.1160
GO			0.8248	0.5188	0.2211
YO			0.9185	0.7574	0.2301
Goel(3)			0.9506	0.8447	0.1749
NHP3AD			0.7941	0.4549	0.2251

表 0.1 和表 0.2 中的 F_r，我们在下文中再予以讨论(表 0.1 和表 0.2 中的结果，都是用我们开发的 SRES 计算出来的)。

从下面的两个表中可以看出：随着选用的模型不同，对同一个软件故障数据集合的分析结果，存在着显著的离散性，其反映出来的不一致性十分明显。DS1 中的累积测试时间长为 849 天，我们分别取 $\Delta t=1$ 和 $\Delta t=84.9$，计算了软件系统的当前可靠度 $R(\Delta t)$；DS2 中的累积测试时间长为 106 天，分别取 $\Delta t=1$ 和 $\Delta t=3.5$，计算了软件系统的当前可靠度 $R(\Delta t)$。在表 0.3 和表 0.4 中，我们分别计算出表 0.1 和表 0.2 中 $R(\Delta t)$的平均相对误差。

从下面的表 0.3 和表 0.4 中可以看出，当 Δt 增大时，无论是使用完全数据或不完全数据，其平均相对误差也会增大。而且，从表 0.3 和表 0.4 的相应的对比中，我们可以看出，进行软件可靠性分析时，使用完全数据比使用不完全数据，所产生的平均相对误差，普遍较小。这也从另一个角度再次说明，我们关于不完全数据中信息的丢失，对软件可靠性分析结果带来影响的观点是正确的。

表 0.3　表 0.1$R(\Delta t)$平均相对误差

模型名	平均相对误差	
	$\Delta t=1$	$\Delta t=84.9$
GO	0.0183	0.2333
YO	0.0211	0.3648
Goel(3)	0.0108	0.0425
LDM	0.0181	0.1526
LVLM	0.0079	0.0626
LVQM	0.0028	0.0243

表 0.4　表 0.2$R(\Delta t)$平均相对误差

模型名	平均相对误差	
	$\Delta t=1$	$\Delta t=3.5$
WM	0.1021	0.3373
GO	0.0782	0.2622
YO	0.0265	0.0770
Goel(3)	0.0624	0.2012
NHP3AD	0.1125	0.3531

表 0.1—表 0.4 明确地揭示了软件可靠性模型应用于软件可靠性分析中，所产生的不一致性。对于不一致性的产生，我们可以解释如下：每一个软件可靠性模型都有着作为它的理论基础的假设，而根本的假设就在于软件运行中发生的故障都服从于某一规律，对这一规律的数学描述就是统计分析规律。指数模型假设软件故障的发生服从指数分布，NHPP 类模型都假设软件故障的发生服从非齐次泊松分布，Weibull 模型则假设软件的故障行为可以用 Weibull 分布来描述，如此等等。这就好像我们观赏一件艺术作品时，不同的观赏者，总是从不同的角度去观察它，并因此得出不同的结论一样。软件系统也可以视为一种艺术作品，为它进行可靠性分析也如同观赏艺术作品一样，不同的模型，其理论假设不同，就好像是从不同的角度去观赏它，得出的结论自然是不一致的。关键的问题在于：我们应根据什么样的原则去选择适当的软件可靠性模型？在选用不同的软件可靠性模型进行分析后，对于所得到的分析结果，应确认哪一个才是比较正确地反映了软件可靠性水平的真实状况？

下面，我们从理论上进一步讨论一下不一致性产生的原因。

故障率，也有人称它为风险函数(hazard function)，也是一个直接源于硬件可靠性的术语。

一般地说，我们用 $\lambda(t)$ 来表示风险函数，$\lambda(t)$ 就是程序正确地运行到时刻 t 时，单位时间内程序发生故障的概率(实际上，$\lambda(t)$ 应该是概率密度，真正的概率应该是 $\lambda(t)\cdot\Delta t$)。

如果以 T 表示从 0 开始运行一程序到程序发生故障为止所经过的时间，则对于不同的运行，T 的值显然是不同的。因而可以断定，T 是一个连续的随机变量。于是有：

$$\lambda(t)\Delta t = P_r\{t < T \leqq t + \Delta t \mid T > t\},$$

因此，我们又可以写：

$$\begin{aligned} R(t+\Delta t) &= P_r\{T > (t+\Delta t)\} \\ &= P_r\{(T > t) \wedge \text{在区间}[t, t+\Delta t]\text{内程序不发生故障}\} \\ &= R(t)[1 - \lambda(t)\Delta t] \end{aligned}$$

对上面的等式关于 t 求微分，我们得到：

$$dR = -\lambda(t)R(t)dt$$

从而可以导出下面的微分方程：

$$\frac{dR}{R} = -\lambda(t)dt$$

解之，得：

$$R(t) = \exp\left[-\int_0^t \lambda(x)dx\right]$$

上式描述了风险函数 $\lambda(t)$ 与可靠度函数 $R(t)$ 的关系。

风险函数 $\lambda(t)$ 在软件可靠性的研究中，受到广泛的关注。因为从上式我们可以看出，一旦知道了 $\lambda(t)$ 的表达式，则 $R(t)$ 即可以直接计算出来。但是在实际的研究工作中，关于 $\lambda(t)$ 与 t 的关系十分复杂，人们无法得出关于 $\lambda(t)$ 对 t 的准确表达式。在现有的软件可靠性模型中，关于 $\lambda(t)$，人们作了各种各样的假设，从而从理论上导致了软件可靠性模型应用中的不一致性。

不一致性产生的危害是十分显而易见的，为克服不一致性的危害，必须克服 SRM 选择上的随意性。换言之，每个用户都极力想选用“最适用”的模型。这样一来，对于“最适用”就必须要有公认的判定准则。

我们开发出的软件可靠性专家系统（Software Reliability Expert System——SRES），正是为了帮助用户在不完全了解许多软件可靠性模型的情况下，根据系统指示，正确提供出待评估软件系统的软件故障数据，就可以正确选用 SRM 以评估软件系统可靠性。因此，我们这里强调提供出来的软件故障数据必须是真实准确地反映出软件系统的故障历史和行为。因为，我们开发的 SRES，有一个确定不变的原则：“一切让数据来说明问题。”SRES 系统所作的推理，判断，全都以对用户提供的软件故障数据所作的预分析为依据，如果在数据中含有虚假成份，那么一切的一切都将付之东流了。

就我们开发的 SRES 的理论基础而言，本 SRES 就是根据对用户提供软件故障数据所作的预分析结果，在事先定义的一组判优准则的约束下的一个优化问题。其优化过程就是指用选定的经验模型对用户提供出的软件故障数据进行拟合的优化，并据此选定一个“最适”模型，用于预计软件系统在将来某一具体时刻的故障行为。

在开发本 SRES 的过程中，我们有许多问题要研究，有不少困难要克服。首先：SRES 所含的经验模型库中，应从现有的众多 SRM 中选出哪些模型作为经验模型？选择原则应如何确定？其次，选择的这些经验模型，有的存在奇异性（Singularity），有的参数估计方程过于复杂，给方程组的求解带来许多困难，在计算方法上我们怎样采取措施，正确地求解这些方程？对于众多的经验模型，判优准则应如何确定？最后，专家系统的知识、事实应如何表示？如何组织？采用什么样的推理机制？这些问题，又可以归结为怎样正确选用好的专家系统开发环境和工具？

最后，在实现这些设计原则的时候，怎样将这些都有机地组合成一个协调工作的 SRES？对于 SRES 的输入应怎样组织？SRES 应该有哪些输入输出信息？它们又应该怎样组织？

本书将就上面所涉及到的问题，进行一一的阐述，力图正确地回答它们。

本书还将就如何正确使用本 SRES，以及本 SRES 整个的工作过程加以详尽的叙述，以使读者能对本 SRES 有一个彻底的、完整的、正确的了解。

SRES 以软件测试技术为保证手段，以测试时收集的故障数据为依据，对数据进行预分析和预处理，并针对估测过程中的不一致性问题，动态地选用适合的模型，对软件系统的可靠性目标及有关的质量指标进行预测，从而一方面保证软件项目的可靠性，另一方面对软件项目开发过程进行管理，以保证软件的开发质量。

SRES 在 DEC_{pc} LP_x 450d2 上开发，操作系统为 MSDOS 6.2，开发语言为 Borland C++3.1，专家系统开发工具及运行环境为美国 NASA 开发的"C 语言实现产生式系统"(C Language Implementation Production System)，简称 CLIPS(5.1 版)。本项目开发之专家系统，全称为"软件可靠性专家系统"(Software Reliability Expert System)，以下简称 SRES。SRES 可在任一厂家生产的 PC486 或 386(要求带协处理器 80386)以及符合要求的兼容机上运行。

本书将就 SRES 所使用的软件故障数据类型、经验模型、CLIPS，等内容作系统的介绍。

SRES 设计原则：

① 菜单驱动方式

用户界面简单明了、统一，用户操作简单，在对 SRES 不作深入了解的情况下，即可很快掌握 SRES 的使用与操作。

② SRES 的判优推理使用正向链接推理方式(CLIPS 的限制)

用户在人—机对话方式下，只要简单地用"Yes"或"No"回答系统的一系列提问，即可得出最终的分析结果。

③ 图形显示。

SRES 在进行数据预分析时，都有图形将模型的分析结果直观地显示出来。

④ 统一规定输入、输出的文件格式

用户只要按规定的格式将软件故障数据文件准备好，就可以开始使用 SRES。输出文件已由系统准备好，在使用完毕以后，退到 MSDOS 状态下，即可打印输出文件。

综合性参考文献

[1] 徐仁佐，谢旻，郑人杰．软件可靠性模型及应用．清华大学出版社，广西科学技术出版社，1994

[2] Mellor P. SOFTWARE RELIABILITY MODELLING: THE STATE OF THE ART, Information and Software Technology, 29(2). 1987

[3] Musa J D, Iannino A and Okumoto K. SOFTWARE RELIABILITY: measurement, prediction, application. McGraw-Hill, New York, 1987

[4] Xie M. SOFTWARE RELIABILITY MODELLING. World Scientific Publishing Co. Pre. Ltd., Singapore, 1991

[5] Giarratano J C and Riley G. EXPERT SYSTEM: PRINCIPLES AND PROGRAMMING. Bosten, PWS-KENT Publishing Co., 1989

[6] Gonzalez A J and Dankel D D. THE ENGINEERING OF KNOWLEDGE-BASED SYSTEMS—THEORY AND PRACTICE, Prentice-Hall International, Inc., New Jersey, 1993

[7] 吴鹤龄. 专家系统工具 CLIPS 及其应用. 北京理工大学出版社,1991

[8] 吴鹤龄,马建峰,李仕文. 图形专家系统工具 GEST. 北京理工大学出版社,1993

[9] Farr W H. A SURVEY OF SOFTWARE RELIABILITY MODELING AND ESTIMATION: NSWC-TR-82-171, NAVAL SURFACE WEAPONS CENTER. September, 1983.

[10] American National Standard. ANSI/AIAA R-013-1992, Recommended Practice for Software Reliability. American Institute of Aeronautics and Astronautics, and American National Standards Institute, 1993

[11] Poore J H, Mills H D and Mutchler D. Planning and Certifying Software System Reliability. IEEE SOFTWARE, January 1993: 88—99

1 SRTS 使用的数据类型

SRTS 将常用的软件可靠性模型所要求的数据分为完全数据、不完全数据和特殊数据三大类。而且，经验模型库中的软件可靠性模型也依这三类数据分为三组：完全数据模型、不完全数据模型和特殊数据模型。对于某些模型，在估计参数时对数据有些特殊要求，我们将在介绍经验模型时，一一加以特别的说明。

完全数据的形式定义为：数据集合 $\{y(i)|y(0)=0, i=1, 2, \cdots, n\}$ 称为完全数据集合，如果：$\forall i(i \in \{1, 2, \cdots, n\} \rightarrow y(i)-y(i-1)=1)$。其中 $y(i)$ 为直到时间 $t(i)$ 时的累积故障数，t_i 为累积时间。

不完全数据的形式定义为：数据集合 $\{y(i)|y(0)=0, i=1, 2, \cdots, n\}$ 称为不完全数据集合，如果：$\exists i(i \in \{1, 2, \cdots, p\} \rightarrow y(i)-y(i-1)>1)$，其中 $y(i)$，$t(i)$ 的意义与完全数据相同。

完全数据的例子见表 1.1 和表 1.2 所示。不完全数据见表 1.3 和表 1.4 所示。

表 1.1 Musa 数据(完全数据)

序号	区间长度	累积天数	序号	区间长度	累积天数
1	115	1	21	15	26
2	0	1	22	390	26
3	83	3	23	1863	27
4	178	3	24	1337	30
5	194	3	25	4508	36
6	136	3	26	834	38
7	1077	3	27	3400	40
8	15	3	28	6	40
9	15	3	29	4561	42
10	92	3	30	3186	44
11	50	3	31	10571	47
12	71	3	32	563	47
13	606	6	33	2770	47
14	1189	8	34	652	48
15	40	8	35	5593	50
16	788	18	36	11696	54
17	222	18	37	6724	54
18	72	18	38	2546	55
19	615	18	39	10175	56
20	589	26			