基于系统结构和运行环境的系统生存性模型

耿 技，宋 旭，陈 伟，秦志光

(电子科技大学计算机科学与工程学院 成都 611731)

【摘要】软件系统安全问题日益严峻，软件生存性研究成为软件安全性研究的延伸和拓展，但是目前的生存性模型很多都不能提供一个可以指导实践的方法。该文研究系统结构和运行环境对系统生存性的影响，在服务的生存性模型中，考虑不同的环境对于服务生存性的影响，在进一步构架系统的生存性模型时，引入系统架构和服务之间的关系，以服务生存性为基础定义系统生存性。以该形式化的生存性模型为基础，结合系统运行环境给出了一个系统在提供的部分或全部服务失效时的恢复方案。

关 键 词 运行环境; 服务生存性; 系统恢复; 系统结构; 系统生存性
中图分类号 TP309 文献标志码 A doi:10.3969/j.issn.1001-0548.2014.01.017

Survivability Model Based on System Structure and Runtime Environment

GENG Ji, SONG Xu, CHEN Wei, and QIN Zhi-guang

(School of Computer Science and Engineering, University of Electronic Science and Technology of China Chengdu 611731)

Abstract The security problems of software become more and more serious，and software survivability is an extension of software security researches. However, the survivability models can rarely bring out a practical method by now. In this paper, we consider the effect of different system structures and runtime environments on system survivability. Firstly, we construct a service survivability model by considering the effect of environment; secondly, we consider the relationship between the system structure and services and construct a model based on service survivability to reflect the system survivability. Finally, based on this model, a recovery approach is illustrated when parts of or all the system's services are failed.

Key words runtime environment; service survivability; system recovery; system structure; system survivability

现代社会，计算机网络和信息系统已经广泛运用于工业、政府、国防、能源、金融、电信、医疗等各个部门，这些系统的架构越来越复杂，涉及的领域及问题也越来越敏感，其安全问题越来越受到人们的关注。目前，安全问题的研究已经由入侵阻止、入侵检测发展到了可生存性研究的层面[1]。产生这种趋势的原因在于，传统的软件安全技术已经无法保证现代复杂系统对于可靠性的要求，而现代软件系统结构的复杂性也已经使得即使是最优秀的系统分析、设计人员也无法保证其可靠性的实现[2]。

对于系统的可生存性，现在还没有一个统一的定义存在，一般而言，系统的可生存性可以描述为：遭遇攻击、失效或者事故时，系统可以及时恢复并完成其任务的能力[3-8]。

在系统设计和实现的过程中，由于系统生存性概念的提出，大大降低了系统设计和实现的难度。对于一个大型、复杂的系统，可生存性较可靠性而言提供了两点优势：1) 系统的体系结构中，可生存性由于允许一些错误的发生使得其对于硬件冗余的需要更低，从而使得系统实现过程中成本降低；2) 对于系统服务的确保工作可以减小到仅仅对于关键服务的确保，这使系统分析和设计人员可以更好地处理系统的复杂性[2]。

目前现有的生存性策略，有些侧重于处理不利环境中的系统应该如何选择某种特定的服务形式和预防策略来降低不利环境对系统的危害，从而保证系统的稳定运行；而另外一些则侧重于不利环境造成一定程度上的系统服务失效的前提下，应该如何

收稿日期：2010－04－23；修回日期：2012－09－20
基金项目：国家自然科学基金(60973118)；中央高校基本科研业务费专项资金(ZYGX2011J072)
作者简介：耿技(1963－)，男，博士生，教授，主要从事系统软件、软件确保、信息安全方面的研究.

对系统进行评估并且采取何种策略逐步恢复系统。无论何种生存性策略，都定义了系统关键服务的概念，并且要求在无论何种环境下，系统的关键服务都应该得到保证。

文献[9]通过对系统生存性进行的数学分析，提出了一个形式化的系统生存性定义和评估模型，以及基于该模型制定系统所服务恢复策略的方法。但是该模型没有考虑系统所处具体操作环境以及系统架构对生存性的影响，且在服务和服务操作集合的定义上也没有考虑到复用的存在。文献[10]从服务和服务操作集合复用的角度对文献[9]提出的系统生存性定义和评估模型进行了改进，但是依然没有考虑到环境因素对于生存性的影响。

文献[2]指出同一系统在不同环境下所获得的生存性是可能不相同的，并进一步分析了环境可能对生存性产生的影响。

另外文献[11]提出了一个多层模型框架用以提高分布式系统中的代理节点的生存性；文献[12]提出并验证了一个针对开源软件生存性的多维测量方法；文献[13]就SOA架构提出了一个针对服务组件的生存性的验证逻辑。

本文基于文献[9]提出的系统生存性形式化定义和评估模型，研究系统结构和所处的运行环境对系统生存性的影响，提出基于系统结构和运行环境的系统生存性模型，以及依托该模型的系统恢复方案。

1 生存性模型

本文旨在建立一个可针对不同环境和不同系统架构的系统生存性模型。建立该模型首先考查系统服务的生存性，并在服务的基础上，根据软件组件之间的关系建立软件的生存性模型。

1.1 服务生存性

系统的生存性意味着系统可以在恶劣环境下对服务进行降级，以保证系统至少可以持续地提供用户能接受的最低限度的服务。因此，在考查系统生存性时，首先要考查系统中涉及的服务的生存性。首先，本文介绍了文献[9]中的生存性模型，并在此基础上进一步考查环境对系统生存性的影响。

在文献[9]中，提出了一个对系统生存性的形式化定义，其将系统生存性定义为一个六元组$(\boldsymbol{S},\mathbf{SO}^{(i)},\varphi,\Lambda,F,\Delta)$，这里考虑到环境因素将会对软件的生存性产生影响，将其扩展为$(\boldsymbol{S},\mathbf{SO}^{(i)},\varphi,\Lambda,F,\Delta,\mathbf{ET},\boldsymbol{E})$，其中，$\boldsymbol{S}=\{s_1,s_2,s_3,\cdots,s_n\}$为系统所能提供的$n$种不同服务所构成的集合。$\mathbf{SO}^{(i)}=\{\mathrm{so}_1^{(i)},\mathrm{so}_2^{(i)},\mathrm{so}_3^{(i)},\cdots,\mathrm{so}_m^{(i)}\}$：服务$s_i$所对应的$m$个操作所构成的集合，并基于该定义有如下集合[9]：系统服务可能实现形式$\mathbf{SO}_{\mathrm{a}}^{(i)}$、关键操作集合$\mathbf{SO}_{\mathrm{c}}^{(i)}$、可接受服务形式$\mathbf{SO}_{\mathrm{ai}}^{(i)}$、不可接受服务形式集合$\mathbf{SO}_{\mathrm{ac}}^{(i)}$。其中每个集合均为集合$\mathbf{SO}^{(i)}$的幂集合的子集。

针对服务s_i，用服务的每个操作来定义如下映射关系以鉴别每个操作的相对重要程度：

$$\varphi^{(i)}:\mathbf{SO}^{(i)}\to[0,+\infty]$$

$$\varphi^{(i)}(\mathrm{so}_j^{(i)})=\omega_j^{(i)}\mid\begin{cases}+\infty & \mathrm{so}_j^{(i)}\in\mathbf{SO}_{\mathrm{c}}^{(i)}\\ \omega_j^{(i)}\in[0,+\infty) & \text{其他}\end{cases}\tag{1}$$

此时，根据文献[9]，对于某个服务$\boldsymbol{s}_i$的实现形式为$\mathbf{SO}_{\mathrm{a}}^{(i)}$，则这个服务在当前环境下以当前形式实现的生存性为：

$$\varsigma^{(i)}=\begin{cases}0 & \mathbf{SO}_{\mathrm{a}}^{(i)}\in\mathbf{SO}_{\mathrm{ai}}^{(i)}\\ \dfrac{\sum_{j=1}^{|\mathbf{SO}_{\mathrm{a}}^{(i)}|}\omega_j^{(i)}\Big|_{\mathrm{so}_j^{(i)}\in\mathbf{SO}_{\mathrm{a}}^{(i)}-\mathbf{SO}_{\mathrm{c}}^{(i)}}}{\sum_{j=1}^{|\mathbf{SO}^{(i)}|}\omega_j^{(i)}\Big|_{\mathrm{so}_j^{(i)}\in\mathbf{SO}^{(i)}-\mathbf{SO}_{\mathrm{c}}^{(i)}}} & \text{其他}\end{cases}\tag{2}$$

本文对Λ、F、Δ的定义如下：Λ表示系统服务进行恢复的操作集合；F表示在系统生存性说明中定义的一系列可能引起系统失效的事件；Δ表示系统服务失效后的恢复策略，即对于Δ定义了一个如式(3)所示的映射关系，其映射中$\{\lambda_i\mid 1\leqslant i\leqslant p\}$各个服务操作间是或的关系。

$$\Delta_{f_k}^{(i)}:\mathbf{SO}^{(i)}\times F\to$$

$$\Lambda(\mathrm{so}_j^{(i)},f_k)\mapsto\Lambda_k^{i,j}=$$

$$\begin{cases}\{\lambda_i\mid 1\leqslant i\leqslant p\} & f_k\text{对}\mathrm{so}_j^{(i)}\text{有影响}\\ \varnothing & f_k\text{对}\mathrm{so}_j^{(i)}\text{无影响}\end{cases}\tag{3}$$

由于系统在不同的环境下所需要提供的服务是不同的，这也造成了在不同的环境下，服务的各个操作其相对的重要程度会发生变化，即映射$\varphi^{(i)}$是受环境影响的，从而系统的生存性评估也是不同的。这里，环境和系统遭受的攻击、失效与灾难是无关的，即环境与系统受到的不利因素无关。

为了对环境进行精确的定义，首先定义环境类型集合$\mathbf{ET}=\{\mathrm{et}_1,\mathrm{et}_2,\mathrm{et}_3,\cdots,\mathrm{et}_k\}$，其中对于某种特性的环境类型$\forall\mathrm{et}_i\in\mathbf{ET}$，有$\mathrm{et}_i=\{\mathrm{et}_i^1,\mathrm{et}_i^2,\mathrm{et}_i^3,\cdots,\mathrm{et}_i^l\}$为环境类型$\mathrm{et}_i$的一种取值。这样环境集合可以定义为集合$\prod_{i=1}^{j}\mathrm{et}_i$(笛卡尔积)的一个子集，即在每种环境类型中选取一个具体值组成对一个特定环境的描述。

如**ET**={时间，天气}，时间={星期一，星期二，星期三}，天气={晴天，阴天}，那么(星期一，晴天)就是一个具体的环境。由此可以精确无歧义的对环境进行定义，并由此构成环境集合 $\boldsymbol{E}\in\prod_{i=1}^{k}\mathrm{et}_i$，由于某些具体环境在现实中是不可能的，因此不要求 $\boldsymbol{E}=\prod_{i=1}^{k}\mathrm{et}_i$。

$\boldsymbol{E}=\{e_1,e_2,e_3,\cdots,e_k\}$ 表示系统可能处于的k种环境，则服务 $\boldsymbol{s}_i$ 在环境 $e_l\in\boldsymbol{E}$ 下提供形式为 $\mathbf{SO}_{\mathrm{a}}^{(i)}$ 的服务时的生存性记为 $\varsigma_{e_l}^{(i)}$。在一段相当长的考察周期内，环境 e_l 出现的概率为 P_{e_l}，则以期望生存性定义服务的实现形式为 $\mathbf{SO}_{\mathrm{a}}^{(i)}$ 时服务 $\boldsymbol{s}_i$ 在不同环境下的生存性：

$$\varsigma^{(i)}=\boldsymbol{E}(\varsigma_{e_l}^{(i)})=\sum_{i=1}^{k}P_{e_l}\varsigma_{e_l}^{(i)} \tag{4}$$

特殊地，如果 $\mathbf{SO}_{\mathrm{c}}^{(i)}$ 对于任何环境下为固定的，即 $\exists e_l\in\boldsymbol{E}$，$\varphi_{e_l}^{(i)}(\mathrm{so}_j^{(i)})=\omega_{j|e_l}^{(i)}=+\infty$ 则对于任何环境 $\mathrm{so}_j^{(i)}\in\mathbf{SO}_{\mathrm{c}}^{(i)}$，那么 $\forall e_l\in\boldsymbol{E}$，$\varphi_{e_l}^{(i)}(\mathrm{so}_j^{(i)})=\omega_{j|e_l}^{(i)}=+\infty$。

由此对于服务的实现形式为 $\mathbf{SO}_{\mathrm{a}}^{(i)}$，如果 $\exists e_l\in\boldsymbol{E}$，$\varsigma_{e_l}^{(i)}=0$，即在环境 e_l 下，$\mathbf{SO}_{\mathrm{a}}^{(i)}\in\mathbf{SO}_{\mathrm{ai}}^{(i)}$，那么存在操作 $\mathrm{so}_j^{(i)}\in\mathbf{SO}_{\mathrm{c}}^{(i)}$ 且 $\mathrm{so}_j^{(i)}\notin\mathrm{SO}_{\mathrm{a}}^{(i)}$。由于所有环境下 $\mathbf{SO}_{\mathrm{c}}^{(i)}$ 相同，则 $\forall e_l\in\boldsymbol{E}$，$\mathbf{SO}_{\mathrm{a}}^{(i)}\in\mathbf{SO}_{\mathrm{ai}}^{(i)}$，故而 $\forall e_l\in\boldsymbol{E}$，$\varsigma_{e_l}^{(i)}=0$，那么在这种情况下，$\varsigma^{(i)}=0$。事实上，基于这个假设，在所有环境下 $\mathbf{SO}_{\mathrm{c}}^{(i)}$、$\mathbf{SO}_{\mathrm{ai}}^{(i)}$、$\mathbf{SO}_{\mathrm{ac}}^{(i)}$ 是相同的。则对于其他情况，即 $\mathbf{SO}_{\mathrm{a}}^{(i)}\in\mathbf{SO}_{\mathrm{ac}}^{(i)}$。首先考察特定环境 $e_l\in\boldsymbol{E}$ 下服务 $\boldsymbol{s}_i$ 的生存性。由式(2)有：

$$\varsigma_{e_l}^{(i)}=\frac{\sum_{j=1}^{|\mathbf{SO}_a^{(i)}|}\omega_{j|e_l}^{(i)}\Big|_{\mathrm{so}_j^{(i)}\in\mathbf{SO}_{\mathrm{a}}^{(i)}-\mathbf{SO}_{\mathrm{c}}^{(i)}}}{\sum_{j=1}^{|\mathbf{SO}^{(i)}|}\omega_{j|e_l}^{(i)}\Big|_{\mathrm{so}_j^{(i)}\in\mathbf{SO}^{(i)}-\mathbf{SO}_{\mathrm{c}}^{(i)}}}=\sum_{k=1}^{|\mathbf{SO}_a^{(i)}|}\frac{\omega_{k|e_l}^{(i)}}{\sum_{j=1}^{|\mathbf{SO}^{(i)}|}\omega_{j|e_l}^{(i)}\Big|_{\mathrm{so}_j^{(i)}\in\mathbf{SO}^{(i)}-\mathbf{SO}_{\mathrm{c}}^{(i)}}} \tag{5}$$

由式(5)可知，在计算服务的生存性时，所关心的并不是该服务的每个操作所对应的 $\omega_{k|e_l}^{(i)}$ 的具体数值，而是其相对情况。因此将环境 e_l 下的映射关系 $\varphi_{e_l}^{(i)}$ 修改为 $\tau_{e_l}^{(i)}$ 并不影响生存性的计算结果。$\tau_{e_l}^{(i)}$ 的定义如下：

$$\tau_{e_l}^{(i)}:\mathbf{SO}^{(i)}\to[0,1]\cup\{+\infty\}$$

$$\tau_{e_l}^{(i)}(\mathrm{so}_k^{(i)})=\varsigma_{j|e_l}^{(i)}\Big|\begin{cases}+\infty & \mathrm{so}_j^{(i)}\in\mathbf{SO}_{\mathrm{c}}^{(i)}\\ \dfrac{\omega_{k|e_l}^{(i)}}{\sum_{j=1}^{|\mathbf{SO}^{(i)}|}\omega_{j|e_l}^{(i)}\Big|_{\mathrm{so}_j^{(i)}\in\mathbf{SO}^{(i)}-\mathbf{SO}_{\mathrm{c}}^{(i)}}} & \text{其他}\end{cases} \tag{6}$$

基于这种映射关系，则生存性为：

$$\varsigma_{e_l}^{(i)}=\sum_{k=1}^{|\mathbf{SO}_{\mathrm{a}}^{(i)}|}\frac{\omega_{k|e_l}^{(i)}}{\sum_{j=1}^{|\mathbf{SO}^{(i)}|}\omega_{j|e_l}^{(i)}\Big|_{\mathrm{so}_j^{(i)}\in\mathbf{SO}^{(i)}-\mathbf{SO}_{\mathrm{c}}^{(i)}}}=\sum_{k=1}^{|\mathbf{SO}_{\mathrm{a}}^{(i)}|}\varsigma_{k|e_l}^{(i)}\Big|_{\mathrm{so}_k^{(i)}\in\mathbf{SO}_{\mathrm{a}}^{(i)}-\mathbf{SO}_{\mathrm{c}}^{(i)}} \tag{7}$$

显然，上述两式是等价的。此时，根据式(4)，服务 $\boldsymbol{s}_i$ 的实现形式 $\mathbf{SO}_{\mathrm{a}}^{(i)}$ 在不同环境下的生存性为：

$$\varsigma^{(i)}=\sum_{l=1}^{k}P_{e_l}\times\varsigma_{e_l}^{(i)}=\sum_{l=1}^{k}P_{e_l}\times\sum_{j=1}^{|\mathbf{SO}_{\mathrm{a}}^{(i)}|}\varsigma_{j|e_l}^{(i)}\Big|_{\mathrm{so}_j^{(i)}\in\mathbf{SO}_{\mathrm{a}}^{(i)}-\mathbf{SO}_{\mathrm{c}}^{(i)}}=\sum_{j=1}^{|\mathbf{SO}_{\mathrm{a}}^{(i)}|}\sum_{l=1}^{k}P_{e_l}\times\varsigma_{j|e_l}^{(i)}\Big|_{\mathrm{so}_j^{(i)}\in\mathbf{SO}_{\mathrm{a}}^{(i)}-\mathbf{SO}_{\mathrm{c}}^{(i)}} \tag{8}$$

进一步，将前述操作和数值的映射关系进行修改。考虑到环境的影响，进一步将映射关系修改为：

$$\sigma^{(i)}:\mathbf{SO}^{(i)}\to[0,1]\cup\{+\infty\}$$

$$\sigma^{(i)}(\mathrm{so}_j^{(i)})=\eta_j^{(i)}\Big|\begin{cases}+\infty & \mathrm{so}_j^{(i)}\in\mathbf{SO}_{\mathrm{c}}^{(i)}\\ \boldsymbol{E}(\zeta_{j|e_l}^{(i)})=\sum_{l=1}^{k}P_{e_l}\times\varsigma_{j|e_l}^{(i)} & \text{其他}\end{cases} \tag{9}$$

那么生存性为：

$$\varsigma^{(i)}=\sum_{j=1}^{|\mathbf{SO}_a^{(i)}|}\eta_j^{(i)}\Big|_{\mathrm{so}_j^{(i)}\in\mathbf{SO}_{\mathrm{a}}^{(i)}-\mathbf{SO}_{\mathrm{c}}^{(i)}} \tag{10}$$

1.2　系统生存性

软件可以正常运行所需要的各个服务之间往往有层次关系，即一个服务的实现可能需要其他几个服务的实现来保证，这样服务按层次组织在一起构成整个系统，每个服务为上层服务提供服务并接受下层服务所提供的服务，这种服务可能是数据的调用也可能是用户操作的流程或其他形式。

在层次服务模型中，这种模型的建立是非常自然的，如图1所示，网上支付服务的实现可能由多种具体的支付服务的实现来实现，而这些服务的实现进一步又可能由其他服务来实现。

对于层次结构的软件其层次模型可由如图2所示树状表述，该树有唯一的根节点，即假设所有的服务最终都将引致一个服务(即根节点所代表的是服务)的实现，这个假设貌似没有依据，但是可以假设所有服务最终成功完成后都将进行服务成功的声明，那么根节点这个服务仅仅代表系统服务运行正确，这样系统的生存性就归结为这个服务的生存性。

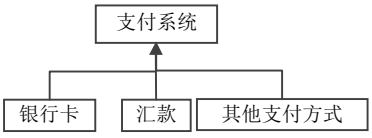

图1 服务层次模型举例

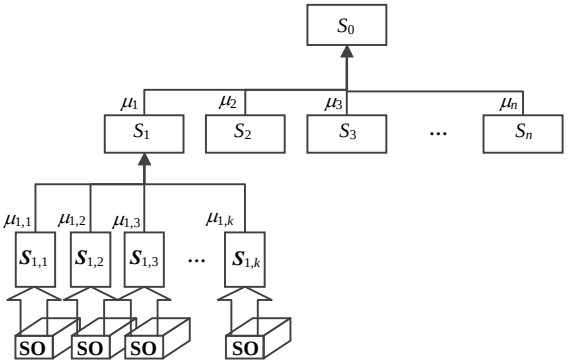

图2 层次服务模型

对于根据服务层次形成的软件系统模型图，首先根据前述模型计算叶子节点的生存性(如图2中 $S_{1,1}$)，对于中间层的服务则根据其子服务的生存性来计算其生存性，然后据此由下而上地逐层计算系统的生存性。

在由子服务计算父服务的生存性过程中，首先根据子服务的相对重要性程度赋以权重λ，则父服务的生存性由子服务的生存性加权平均得出。如图2所示，对于服务 $\boldsymbol{S}_1$ 由子服务 $\boldsymbol{S}_{1,1},\boldsymbol{S}_{1,2},\boldsymbol{S}_{1,3},\cdots,\boldsymbol{S}_{1,k}$ 组成，对应的权重分别为 $\mu_{1,1},\mu_{1,2},\mu_{1,3},\cdots,\mu_{1,k}$，那么父服务 $\boldsymbol{S}_1$ 的生存性为：

$$\varsigma^{(1)}=\sum_{i=1}^{k}\mu_{1,i}\times\varsigma^{(1,i)}\qquad \mu_{1,i}\in[0,1]\text{ 且}\sum_{i=1}^{k}\mu_{1,i}=1 \tag{11}$$

这样就可以逐层计算从而得出服务 $\boldsymbol{S}_0$ 的生存性，即系统的生存性。

事实上，图2所示模型的系统生存性为：

$$\varsigma=\mu_1\varsigma^{(1)}+\mu_2\varsigma^{(2)}+\cdots+\mu_n\varsigma^{(n)}=$$
$$\mu_1(\mu_{1,1}\varsigma^{(1,1)}+\mu_{1,2}\varsigma^{(1,2)}+\cdots+\mu_{1,k}\varsigma^{(1,k)})+\cdots \tag{12}$$

由此可以知，在层次服务模型树中，叶子服务 $\boldsymbol{S}_{i,j,k,\cdots,l}$ 的权重为 $\mu_i\mu_j\mu_k\cdots\mu_l$，根据该权重可以直接由叶子服务推算出系统的生存性。

1.3 系统生存性的恢复策略

以上建立了系统生存性的数学模型，下面将该生存性模型运用于实践，指导系统的恢复过程。

当系统受到攻击或者发生系统错误等其他不利因素影响时，系统的关键服务可能会产生失效或者仅仅是非关键服务产生失效，而这将带来两种截然不同的后果：关键服务失效会使得系统无法为客户提供服务；而非关键服务失效则仍可以为用户提供有限度的服务。因此在恢复过程中，首先应该考虑关键服务的恢复，再进行非关键服务的恢复。

1.3.1 系统关键服务的恢复

当系统受到不利环境影响时，系统的关键服务也有可能受到影响，从而使得系统无法提供任何服务或不能提供用户所需的最低程度的服务，为了保证系统具有生存性，可以向用户提供一定程度的服务，在遇到不利环境时，系统应首先对系统的关键服务进行恢复。假设 $f_k\in F$ 发生，造成系统失效，则恢复过程如下：

1) 构造待恢复关键服务集合 $R_k^{\mathrm{c}}=\{\mathrm{so}_j^{(i)}\mid\forall i,\forall j,\ \mathrm{so}_j^{(i)}\in\mathbf{SO}_{\mathrm{c}}^{(i)}\text{且}\Delta_{f_k}^{(i)}(\mathrm{so}_j^{(i)},f_k)\neq\varnothing\}$，系统恢复操作集合 $\mathrm{RO}=\varnothing$。

2) 构造系统恢复操作集合 Γ_k^{c}，对于 $(\lambda_i,n_i)\in\Gamma_k^{\mathrm{c}}$，有 $\lambda_i\in\bigcup_{\mathrm{so}_j^{(i)}\in R_k^{\mathrm{c}}}\Delta_{f_k}^{(i)}(\mathrm{so}_j^{(i)},f_k)$，$n_i$ 是恢复操作 λ_i 所影响的关键服务的计数。

3) 选取具有最大的 n_i 的二元组 $(\lambda_i,n_i)\in\Gamma_k^{\mathrm{c}}$，执行恢复操作 λ_i。如果多个恢复操作具有相同的影响服务计数且最大，则计算这些恢复操作执行后系统在该环境下的生存性，并选取生存性最大的那个操作。(事实上，这里由于系统关键服务还未恢复生存性，依然为0，这里计算生存性时假设关键服务均已恢复，这是考虑到恢复操作 λ_i 可能影响到非关键服务的操作，从而使得关键服务恢复完毕后非关键服务可以尽可能多的恢复。)

4) $\mathrm{RO}\cup=\{\lambda_i\}$，$\Gamma_k^{\mathrm{c}}-=\{(\lambda_i,n_i)\}$，对于所有的 $\mathrm{so}_j^{(i)}\in R_k^{\mathrm{c}}$，如果 $\lambda_i\in\Delta_{f_k}^{(i)}(\mathrm{so}_j^{(i)},f_k)$，$R_k^{\mathrm{c}}-=\{\mathrm{so}_j^{(i)}\}$，对于 $\forall\lambda_j\in\Delta_{f_k}^{(i)}(\mathrm{so}_j^{(i)},f_k),j\neq i$，$\Gamma$ 中的元素 (λ_j,n_j) 更新为 (λ_j,n_j-1)。如果 $n_j-1=0$，则 $\Gamma_k^{\mathrm{c}}-=\{(\lambda_j,n_j)\}$。

5) 如果 $R_k^{\mathrm{c}}=\varnothing$，则终止本步骤；否则转到步骤3)。

1.3.2 系统非关键服务的恢复

在完成第一步修复措施后，系统可以为用户提供低限度的可接受的服务，但是到目前为止系统还

未能提供完善的服务，部分功能还未恢复。为了能够尽可能地向用户提供完善的服务，需要进一步对系统的非关键服务进行恢复。

由于系统的生存性是受系统运行所处环境影响的，不同环境下，系统各个服务的重要程度以及用户偏好是有可能不同的，因此在进行非关键服务恢复时，需要考虑到系统恢复时所处的环境和系统在其他环境的表现。

假设系统在环境A下受到不利因素的影响，因此系统需要进行恢复操作，下面考虑环境A的两种极端的可能：1) 环境A可能延续的时间非常长，使得系统在进行恢复操作的过程中环境不会发生改变，这样系统恢复过程中对于生存性指数的考察仅仅需要考虑环境A下的生存性指数；2) 环境A延续的时间非常短，其后续环境变化完全随机发生，在该情况下，对于系统的恢复应该要求在各个环境下尽可能地达到较好的效果，一般来讲要求生存性指数在各个环境下的期望最高。

在现实中，面临的情况多是上述两种可能的一个折中，因此在非关键服务的恢复过程中，要综合考虑系统所处环境和系统在各个环境下的期望。一种简单的办法是为软件在各个环境下的期望生存性与当前环境下的生存性分配权重，考虑该权重应该与当前环境持续时间有关系，故设在恢复过程中参数为$\tau_A(t)\varsigma_A^{(i)}+(1-\tau_A(t))\varsigma^{(i)}$，其中$\varsigma_A^{(i)}$为服务$\boldsymbol{S}_i$在环境$A$下的生存性，$\varsigma^{(i)}$为服务$\boldsymbol{S}_i$在各种环境下的期望生存性，$\tau_A(t)\in[0,1]$为环境$A$的权重，其具体数值和系统处于环境$A$下的持续时间有关。这样，对于上述两种特殊情况，简单的令$\tau_A(t)=0$或1即可。

基于以上假设，对于系统非关键服务的恢复可以依照以下步骤进行：

1) 构造待恢复的服务集合$R_k^i=\{\mathrm{so}_j^{(i)}\mid\forall i,\forall j,\ \mathrm{so}_j^{(i)}\notin\mathbf{SO}_{\mathrm{c}}^{(i)}$且$\forall\lambda_i\in\mathrm{RO},\lambda_i\notin\Delta_{f_k}^{(i)}(\mathrm{so}_j^{(i)},f_k)\}$，以及恢复操作集合$\Gamma_k^i=\bigcup_{\mathrm{so}^{(i)}\in R_k^i}\Delta_{f_k}^{(i)}(\mathrm{so}_j^{(i)},f_k)$。

2) 对所有$\lambda_i\in\Gamma_k^i$，计算采用恢复操作λ_i后的生存性指数$\tau_A(t)\varsigma_A^{(i)}+(1-\tau_A(t))\varsigma^{(i)}$，其中$A$为系统当前处的环境。选取具有最高生存性指数的恢复操作λ_i。

3) 对于$\forall\mathrm{so}^{(i)}\in R_k^i$，$\lambda_i\in\Delta_{f_k}^{(i)}(\mathrm{so}_j^{(i)},f_k)$，$R_k^i--=\{\mathrm{so}^{(i)}\}$，$\mathrm{RO}\cup=\{\lambda_i\}$，$\Gamma_k^i--=\{\lambda_i\}$。

4) 如果$\Gamma_k^i=\varnothing$或者$R_k^i=\varnothing$则终止本步骤；否则转到步骤2)。

基于以上两步恢复，系统可以在遭遇到不利因素影响时，保证可以满足用户基本需求的前提下，尽可能地为用户提供更加全面、可靠的服务。

2　案例分析

假设某系统的系统结构A如图3所示。

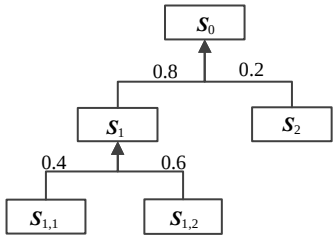

图3　系统结构示意

其中环境$\boldsymbol{E}=\{e_1,e_2,e_3\}$在系统运行过程中，每种环境出现的概率$P$={0.2,0.6,0.2}；系统在运行过程中可能受到的不利影响有$F=\{f_1,f_2,f_3\}$系统修复集合为$\Lambda=\{\lambda_1,\lambda_2,\cdots,\lambda_9\}$，下面对$f_1$发生时进行分析。系统服务$\boldsymbol{S}_{1,1}$、$\boldsymbol{S}_{1,2}$、$\boldsymbol{S}_2$所包含的服务操作、权重以及$f_1$发生是的修复策略，如表1～表3所示。

表1　服务$\boldsymbol{S}_{1,1}$的权重映射和恢复策略

$\boldsymbol{S}_{1,1}$	权重映射$\varphi_{e_i}^{1,1}$			修复策略$\Delta_{f_1}^{(i)}$
	e_1	e_2	e_3	
$\mathrm{so}_1^{(1,1)}$	+∞	+∞	+∞	λ_2
$\mathrm{so}_2^{(1,1)}$	1	2	5	λ_2,λ_3
$\mathrm{so}_3^{(1,1)}$	3	1	2	λ_2,λ_3
$\mathrm{so}_4^{(1,1)}$	4	4	4	$\lambda_2,\lambda_3,\lambda_6$
$\mathrm{so}_5^{(1,1)}$	2	5	1	λ_6
$\mathrm{so}_6^{(1,1)}$	+∞	+∞	+∞	—
$\mathrm{so}_7^{(1,1)}$	5	3	3	λ_3

表2　服务$\boldsymbol{S}_{1,2}$的权重映射和恢复策略

$\boldsymbol{S}_{1,2}$	权重映射$\varphi_{e_i}^{1,2}$			修复策略$\Delta_{f_1}^{(i)}$
	e_1	e_2	e_3	
$\mathrm{so}_1^{(1,2)}$	+∞	+∞	+∞	λ_2,λ_3
$\mathrm{so}_2^{(1,2)}$	1	3	2	λ_2,λ_3
$\mathrm{so}_3^{(1,2)}$	2	1	1	λ_2,λ_3
$\mathrm{so}_4^{(1,2)}$	+∞	+∞	+∞	—
$\mathrm{so}_5^{(1,2)}$	3	2	3	λ_3

表3　服务$\boldsymbol{S}_2$的权重映射和恢复策略

$\boldsymbol{S}_2$	权重映射$\varphi_{e_i}^{2}$			修复策略$\Delta_{f_1}^{(i)}$
	e_1	e_2	e_3	
$\mathrm{so}_1^{(2)}$	+∞	+∞	+∞	λ_2,λ_3
$\mathrm{so}_2^{(2)}$	+∞	+∞	+∞	λ_2,λ_3
$\mathrm{so}_3^{(2)}$	4	2	3	λ_2,λ_3
$\mathrm{so}_4^{(2)}$	1	3	4	λ_2,λ_3
$\mathrm{so}_5^{(2)}$	+∞	+∞	+∞	—

$so_6^{(2)}$	3	4	2	—
$so_7^{(2)}$	2	1	1	λ_3

以下对环境 e_1 下 f_1 发生后，系统的修复过程进行分析。

1) 系统关键服务的恢复 $R_k^c=\{so_1^{(1,1)},so_1^{(1,2)},so_1^{(2)},so_2^{(2)}\}$，$\Gamma_k^c=\{(\lambda_2,4),(\lambda_3,3)\}$，故 $RO=\{\lambda_2\}$，即第一步采取恢复操作 λ_2。此后 $R_k^c=\varnothing$，系统关键服务恢复完毕。此时，在环境 e_1 下系统的生存性为 $\varsigma=161/375\approx0.429$

2) 系统其他服务的恢复，假设 $\tau_{e_1}(t)=(4-t)/4$，进行完每一步恢复操作后 t 加1，所有恢复操作均耗时1个时间单位，初始 t =1。此时有：$R_k^i=\{so_5^{(1,1)},so_7^{(1,1)},so_5^{(1,2)},so_7^{(2)}\}$，$\Gamma_k^i=\{\lambda_3,\lambda_6\}$。

如果采用恢复操作 λ_3，则有系统的生存性指数为：$\tau_{e_1}(1)\varsigma_{e_1}^{(3)}+(1-\tau_{e_1}(1))\varsigma^{(3)}\approx0.949$。

如果采用恢复操作 λ_6，则有系统的生存性指数为0.638，据此，恢复操作选择 λ_3，即 $RO=\{\lambda_2,\lambda_3\}$。此时，$R_k^i=\{so_5^{(1,1)}\}$，$\Gamma_k^i=\{\lambda_6\}$，故下一步恢复操作选取 λ_6，并完成系统恢复，故恢复操作集合为 $RO=\{\lambda_2,\lambda_3,\lambda_6\}$。

3 结 束 语

本文以现有的系统生存性模型为基础，逐一考查了系统运行环境和软件架构对于生存性的影响，考察了不同环境下的生存性定义以及环境因素如何在环境得到体现，另外，对于系统的层次结构在生存性中如何得以体现也进行了分析，并依此给出了系统生存性的形式化定义。进一步，对于本文中提出的系统生存性模型，给出了一个应用的案例，并在案例中进行分析，给出了系统的恢复过程。

参 考 文 献

[1] 张永, 方滨兴, 包秀国. 网络可生存性研究概述[J]. 计算机工程与应用, 2005, 41(7): 119-121.
ZHANG Yong, FANG Bing-xing, BAO Xiu-guo. The research summarization of network survivability[J]. Computer Engineering and Applications, 2005, 41(7): 119-121.

[2] KNIGHT J, STRUNK A E. Achieving critical system survivability through software architectures[J]. Architecting Dependable Systems II, 2004, 3069: 51-78.

[3] HILTUNEN M A, SCHLICHTING R D, UGARTE C A, et al. Survivability through customization and adaptability: the cactus approach[C]//DARPA Information Survivability Conference and Exposition. Hilton Head: IEEE, 2000.

[4] KNIGHT J C, STRUNK E A, SULLIVAN K J. Towards a rigorous definition of information system survivability[C]// DARPA Information Survivability Conference and Exposition. Washington: IEEE, 2003.

[5] ELLISON R J, FISHER D A, LINGER R C, et al. Survivability: Protecting your critical systems[J]. Internet Computing, 1999, 3(6): 55-63.

[6] CALDERA J. Survivability requirements for the US health care industry[D]. Pittsburgh: Carnegie Mellon University, 2000.

[7] ELLISON R J, FISHER D A, LINGER R C, et al. An approach to survivability systems[R]. CERT Coordination Center, Software Engineering Institute. Carnegie Mellon University, 1999.

[8] BYON I. Survivability of the U.S. electric power industry[D]. Pittsburgh: Carnegie Mellon University, 2000.

[9] AYARA A, NAJJAR F. A formal specification model of survivability for pervasive systems[C]//International Symposium on Parallel and Distributed Processing with Applications. Sydney: IEEE, 2008.

[10] XIA Q, WANG Z. Survivability recovery of information system based on component availability[C]//IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER). Kunming: IEEE, 2011.

[11] SHARON S, DENNIS E. Survivability through run-time software evolution[C]//IEEE International Symposium on Network Computing and Applications. Cambridge: IEEE, 2009: 108-113.

[12] UZMA R, MARIETTA J T. Defining and evaluating a measure of open source project survivability[J]. Software Engineering, 2012, 38(1): 163-174.

[13] ZUO Yan-jun, LANDE S. A logical framework of proof-carrying Survivability[C]//IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). Changsha: IEEE, 2011: 472-481.

编 辑 张 俊