ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.12, December 2007, pp.3031−3047
DOI: 10.1360/jos183031

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于可生存性规范的软件构件系统恢复的建模与推理*

卢 暾+

(复旦大学 计算机与信息技术系,上海 200433)

Modeling and Reasoning of the Software Component Based System Recovery Based on Survivability Specification

LU Tun

(School of Computing and Information Technology, Fudan University, Shanghai 200433, China)

+ Corresponding author: Phn: +86-21-55664465, Fax: +86-21-65643502, E-mail: lutun@fudan.edu.cn

Lu T. Modeling and reasoning of the software component based system recovery based on survivability specification. *Journal of Software*, 2007,18(12):3031−3047. http://www.jos.org.cn/1000-9825/18/3031.htm

Abstract: The component-based system will provide a predefined survivability specification which consists of corresponding degraded services in the presence of various kinds of malicious attacks, system failures or accidents. The main contributions of this paper are (1) presenting the method to represent service core based on component families and installation orders, which can precisely capture the system services perceived by users; (2) proposing the reasoning rules of system recovery based on component compatibility and installation execution, which are used to judge the success property (the newly started service works well) and safety property (formerly started services are not damaged); and (3) presenting the algorithms to simplify installation execution based on the concept of projection, which supports the reasoning analysis of system recovery of big scale. By the analysis process based on survivability specification, the corresponding reasoning rules can be systemically applied in practice. A component-based system named MVoD (mobile video-on-demand) is illustrated to demonstrate the practicability and efficiency of the formal model and the analysis method.

Key words: component-based software engineering; component-based system; component reuse; survivability specification; service core; system recovery

摘 要: 从基于构件的系统在面临各种不同的恶意攻击、系统故障和意外事故时,根据预先定义好的生存性规范为用户提供不同等级的系统服务出发,研究了构件系统恢复中的一些关键问题.其主要贡献是:(1) 提出了基于构件家族和构件安装顺序的服务核表示方法,该方法能够准确表达系统在不同条件下为用户提供能够感知的服务;(2) 建立了基于构件兼容性和构件安装执行的构件系统恢复性质的推理规则,用于判断系统恢复的成功性(新启动的服务能够正常工作)和安全性(原有的服务不会受到损害);(3) 提出了基于投影的构件安装执行的简化算法,为大规模构件系统恢复的性质推理分析提供了支持.利用基于可生存性规范的构件系统恢复的分析流程,可以在实际中系统地应用相应的推理规则.最后通过一个基于构件的移动视频点播 MVoD 系统来展示所提出的模型与方法的实用性和有效性.

* Supported by the National Grand Fundamental Research 973 Program of China under Grant No.2005CB321905 (国家重点基础研究发展规划(973)); the Open Subject Fund of Jiangsu Provincial Key Laboratory of Network and Information Security of China under Grant No.BM2003201 (江苏省网络与信息安全重点实验室开放课题基金)

Received 2007-06-11; Accepted 2007-10-16

关键词: 基于构件的软件工程;基于构件的系统;构件重用;可生存性规范;服务核;系统恢复
中图法分类号: TP311 文献标识码: A

基于构件的软件工程(component-based software engineering,简称 CBSE)已逐渐成为软件开发的主流范例.基于构件的软件复用和开发被认为是提高软件开发效率和质量的有效途径,并在分布式系统中得到了广泛应用[1],而这其中就包括各种各样的关键应用,如嵌入式的处理控制系统(远程通信系统、配电系统、空中交通控制系统)、交易处理系统(银行系统、股票交易系统、航班预留系统)等.我们的日常生活、社会生产等活动都与这些关键应用密切相关,如果这类系统发生故障或是受到攻击,将会导致灾难性的后果,甚至影响社会的安全与稳定.

随着 Internet 的出现与发展,现在的基于构件的系统(component-based system,简称 CBS)所面临的环境从静态、封闭逐步走向动态、开放.从技术的角度来看,以软件构件等技术支持的软件实体(如现在流行的 Web 服务等)以开放、自主的方式存在于 Internet 的各个节点之上,任何一个软件实体可在开放的环境下通过某种方式加以发布,并以各种协同方式与其他软件实体进行跨网络的互连、互通、协作和联盟[2,3].软件构件的这些新特点,对基于构件的系统的结构与可信性属性(如可生存性等)带来了巨大的影响.如何使基于构件的系统在出现故障和安全攻击的情况下得以生存,是一个需要解决的关键问题[4].

现在,国内外对系统的可生存性研究主要集中在以下 4 个方面:1) 网络通信系统的可生存性研究.这方面的研究起步较早,也是目前可生存性研究较多的领域.它主要通过分析网络链路的连通特性,对通信设备的性能从硬件上来研究系统的可生存性[5−8];2) 网络信息系统可生存性的定义研究.主要是对可生存性的网络信息系统所应该具有的某些能力进行描述性的定义.如通过描述系统能够抵挡某些类型的安全攻击[9],能够对某些错误进行容错处理[10],或具有入侵检测的能力[11]等来定义系统的可生存性;3) 信息系统的可生存性分析研究.这方面的研究主要有两个方向:一是对系统的结构和代码进行分析,找出可能存在的可生存性薄弱环节,然后采用一定的方法(如随机改变软件系统的特征)来提高系统的可生存性[12];二是在系统开发的过程中试图通过利用专家经验所定义的规范,把可生存性融入到系统设计的生命周期中去,以提高软件的可生存性[13];4) 构建可生存的信息系统的方法研究.主要是利用入侵检测、故障隔离、冗余和自适应等技术来提高和增强某类特定系统的可生存性[14,15].

从对研究现状的分析来看:一方面,关于什么是系统可生存性,即系统可生存性的定义,至今都没有一个统一的认识.现在,大部分的研究都是在不同的上下文环境中对生存性应具备的一些属性进行描述;另一方面,现有对可生存性的研究大多数还是采用以安全为中心的分析方法(如可靠性、可信性和可获得性等),而对于基于具有自主性、开放性和强动态性的软件构件系统的可生存性研究十分匮乏[16].

本文使用的“可生存性”定义是现在较为通用的定义:系统在受到恶意攻击、出现系统失效和故障等事故的情况下,仍然能够继续实时地为用户提供基本服务(essential service)的能力[17].它强调的核心是系统作为一个整体在某些条件下提供用户能够接受和感知的服务的能力.Knight 等人注意到这样的事实,即用户在大部分的时间内期望正常的系统功能;如果不能获得正常的功能,用户就可能会在不同的条件下接受可以替换的服务功能[18,19].当在系统状况逐渐好转的情况下(如替换故障构件等),构件系统就从一个只能提供较少服务功能的系统状态恢复到一个能够提供相对较多服务功能的系统状态.那么,如何表示系统从一个系统状态恢复到另一个系统状态、以怎样的顺序安装构件来启动一个相应的服务以及如何衡量与推理系统状态间转移的效果和性质等,就是本文需要解决的问题.

目前,关于面向可生存性的构件系统恢复的建模与分析方面的研究比较少.我们针对上述问题提出了一种基于可生存性规范的构件系统恢复的形式化模型.该模型利用构件家族中不同构件间的兼容关系以及构件安装策略来建模服务启动的方式,并给出基于该模型的构件系统恢复的性质分析方法来推理新启动的服务能否正常运行(成功性)和已启动的服务不受影响(安全性).最后,通过一个实例来说明该模型和分析方法的可用性和有效性.

本文第 1 节给出研究动机与问题的描述.第 2 节提出基于可生存性规范的系统恢复形式化模型.第 3 节提出系统恢复性质的推理规则,并给出基于该形式化模型的系统恢复方法与流程.第 4 节通过一个实例说明该模型在实际应用中的有效性.第 5 节对比相关工作,指出本文主要工作的特点.第 6 节给出结论与展望.

1 研究的动机与问题的描述

1.1 研究的实例

本节给出一个简单的基于构件的系统——移动视频点播系统(mobile video-on-demand,简称 MVoD),以阐述我们研究的动机与方法.一个 MVoD 系统包括以下 11 个功能性构件:MVoD 是该系统的统一接入点,是用户使用系统的门户;名字查询(NS)构件负责查找本地转发器(LF)的名字;服务定位(SL)构件负责定位相对于当前用户来说最近的本地转发器;本地转发器负责把数据转发给自己所覆盖范围内的移动设备;授权、认证和计费(3A)构件提供识别用户、权限管理和服务计费等功能;样片预览(SP)提供一些免费的视频预览功能;视频点播(VoD)构件利用同步(SC)构件,对视频编解码(VC)的视频流和音频编解码(AC)的音频流进行同步,提供视频点播的功能;流量控制(FC)构件在网络拥塞或服务器过载的情况下,动态调整数据流的发送速度;内容数据库(CDB)则存放着各种视频资料.在大多数实际的情况下,提供某种系统功能的构件会有多个,这些具有相似功能的构件就构成了一个构件家族.MVoD 系统中的构件按照一定的结构进行连接,并通过它们交互为用户提供所需的服务.这些构件间的连接不但反映了这些构件间的功能依赖关系,而且决定了系统中各个构件的安装顺序.例如,LF 构件依赖 3A 构件(称为“被依赖者”)的功能来实现系统的授权、认证和计费服务,所以,3A 构件必须先于 LF 构件安装.图 1 给出了 MVoD 系统中所有这些构件功能间的依赖关系.

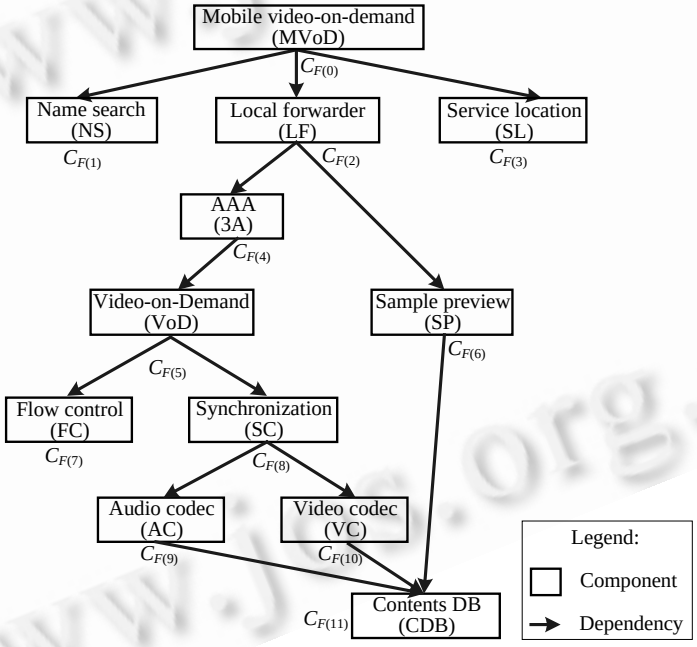

Fig.1 Component dependency graph of components in MVoD

图 1 MVoD 系统中的构件依赖关系图

从用户与系统交互的观点来看,我们关心的是 MVoD 系统可以提供给哪些用户能够感知的服务.而不考虑那些不能以某种方式传递给用户的系统内部服务.例如,LF,SP,CDB 提供的样片预览服务,必须与 MVoD,NS,SL 一起才能提供用户可以感知和使用的服务.基于用户对系统功能的感知(perception)和系统的应用逻辑(application logic),MVoD 系统可以提供如下 11 项服务,如图 2 所示:服务查询与定位服务(s_1);样片预览服务(s_2);

视频点播用户的授权、认证和计费服务(s_3);音频服务(s_4);视频服务(s_5);音频和视频服务(s_6);带流量控制的视频点播服务(s_7);带样片预览的音频服务(s_8);带样片预览的视频服务(s_9);带样片预览的视频点播服务(s_{10});完整的移动视频点播服务(s_{11}).

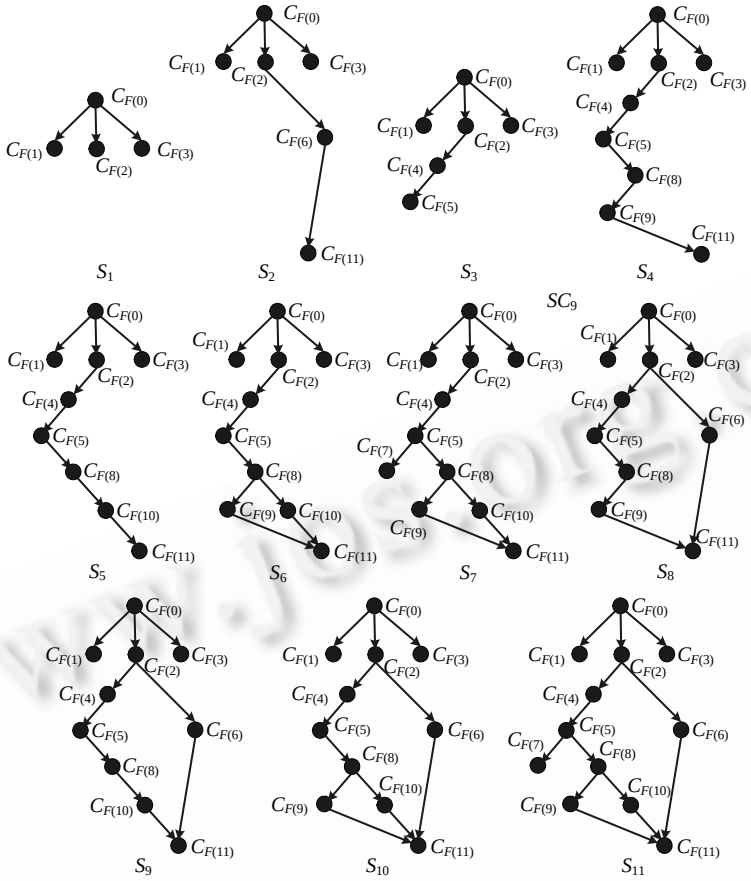

Fig.2 Perceived services by users of MVoD system

图 2 MVoD 系统提供的用户能够感知的服务

在受到外界的恶意攻击,或者某些构件发生故障,或者客观环境出现事故等情况下,MVoD 系统就只能提供部分用户能够感知的服务.例如,如果授权、认证和计费(3A)构件受到攻击(或者出现故障,或者发生地震、水灾等灾难性事故),那么,系统这个时候只能提供服务查询与定位服务(s_1)和样片预览服务(s_2)这两个受限的服务.可见,在运行的过程中,系统每个时刻能够为用户提供的功能是通过一个服务的集合(如$\{s_1,s_2\}$)来体现的(称为“服务核”).实际上,服务核表示系统在某个时刻所能提供的某个级别的系统功能.所有服务核的集合则构成了该系统的生存性规范,表示系统所有可能的系统状态.而系统恢复则是描述在某些服务启动(通过支持它的构件按照功能依赖关系进行安装)的过程中,系统从一个系统状态转移到另一个系统状态,实现一个服务核恢复到另一个服务核.其中有如下 5 个关键问题需要解决:

(1) 如何在一个服务核中启动一个服务及其相应的构件?

(2) 如何表示系统从一个服务核恢复到另一个服务核?

(3) 构件安装顺序对系统恢复具有什么样的影响?

(4) 如何衡量系统恢复过程的效果?

(5) 如何推理系统恢复过程中服务核间转移的性质?

1.2 问题的描述

在问题进行描述之前,我们先对本文中频繁使用的两个概念“构件”和“服务”进行定义.本文所指的“构件”是“软件构件(software component)”.它采用使用较为广泛的Szperski对软件构件的定义,即软件构件是一个仅带特定契约接口和显式语境依赖的结构单元,它可以独立部署,且易于第三方组合[20].同时,采用 Avizienis 等人对“服务(service)”的定义,即一个系统所能传递的服务(delivered service)是该系统的用户所能感知的(perceived)实际行为(actual behavior)[21].

在实际的基于构件的系统中,一个服务通常是由一组来自不同构件家族的构件,按照一定的逻辑连接和依赖关系(dependency)并通过构件间的交互与协作来支持(support)的.构件间的逻辑连接与依赖关系,决定了在支持某个服务启动(start)时,这些构件的安装(install)顺序.

一个构件家族由一组具有相似功能的构件所组成,我们用 $C_{F(i)}$来表示具有家族号 i 的构件家族.对于构件家族 $C_{F(i)}$中的任何一个构件 c_i^p,p 是版本号.对于一个构件集合 C_k,用 π_{C_k} 表示该集合中构件根据家族号集中进行的某种排列,用$\{\pi_{C_k}\}$表示 π_{C_k} 的构件集合(明显地有,$C_k=\{\pi_{C_k}\}$).C_k 中构件的所有排列记为 $\Pi(\pi_{C_k})$.

服务 s_k 由构件集合的$\{\pi_{C_k}\}$所支持,则记为 $supp(s_k,\{\pi_{C_k}\})$.同时,用顺序连接符∘来连接两个实体(entity),表示一个实体先于(precede)另一个实体.例如,$c_i^p\circ c_j^q$ 意味着构件 c_i^p 在 c_j^q 之前安装.实体可以是构件,也可以是后面讨论的恢复步或者安装表达式等.

2 构件系统恢复的形式化建模

2.1 可生存性规范

定义 1(服务核). 一个服务核(service core)是一个二元组 $SC=(S^{sys},C^{sys})$,这里:

- $S^{sys}=\{s_1,\ldots,s_{k-1}\}$是构件系统在某个时刻所能提供的服务的集合;
- C^{sys} 是一个构件集合,且满足对$\forall s_i\in S^{sys}$,存在$\{\pi_{C_i}\}\subseteq C^{sys}$ 使得 $supp(s_i,\{\pi_{C_i}\})$.

服务核是通过对构件系统所能提供的用户能够感知的服务进行抽取而得到的.一个构件系统的可生存性规范(survivability specification)是由一组预先定义好的服务核所组成,记为 $SPEC=\{SC^1,\ldots,SC^n\}$.由于构件系统具有层次化的特性,即一个构件可能又是一个构件系统.所以,根据不同的粒度,系统可以拥有多个可生存性的规范.同一个可生存性规范中不同服务核的服务集合一定是不同的,而不同的可生存性规范中服务核的服务集合可以是相同的.

当一组构件从异常情况中(攻击、故障或事故)恢复,并在系统中按照一定的顺序进行安装时,它们支持的服务就启动了.系统则从当前的服务核向着另一个服务核进行系统恢复.

定义 2(服务核恢复步). 对于 $supp(s_k,\{\pi_{C_k}\})$,从当前服务核 $SC=(S^{sys},C^{sys})$出发的服务核恢复步记为 $SC^R=(S^{R-sys},C^{R-sys})$,这里:

- 服务 s_k 在 $S^{sys}(s_k\notin S^{sys})$中启动;
- $\{\pi_{C_k}\}$中所有的构件按照它们在 π_{C_k} 中的顺序在 C^{sys} 中进行安装.

服务 s_k 的启动相对比较简单,只需把 s_k 当成一个新的服务插入到 S^{sys} 中即可,即 $S^{R-sys}=S^{sys}\cup\{s_k\}(s_k\notin S^{sys})$.$\{\pi_{C_k}\}$中构件的安装就比较复杂,牵涉到构件的安装策略问题.因为当一个构件要在 C^{sys} 中安装时,C^{sys} 中可能已经存在了来自相同构件家族的构件.从构件重用的角度来看,如何安装这个构件(直接安装、替换已有的构件安装,或是什么也不做等等)就是所谓的安装策略.

对于一个服务核恢复步,它具有两种恢复效果:一是成功恢复,即在该服务核恢复步进行之后,新启动的服务能够提供用户所需的正常功能;二是安全恢复,即在该服务核恢复步进行之后,以前为用户提供正常功能的服务仍然能够继续提供正常的功能.现实中的构件系统恢复对成功性和安全性的需求是不同的,即这两者的重要性是有区别的.对于一个系统来说,最理想的恢复情况是系统恢复既能保证成功性又能保证安全性.当不能兼顾时,安全性则显得更为重要.因为安全性至少能保证原来可以正常提供的服务仍然能够正常提供,而成功性可能

会为了提供恢复的服务,而导致原来能够正常提供的服务受损.

根据一个服务核的恢复步,下面给出成功和安全的服务核恢复步的定义.

定义 3(成功的服务核恢复步). 给定服务核恢复步 $SC^R=(S^{R-sys},C^{R-sys})$,如果存在 $\{\pi_{C_k'}\}\subseteq C^{R-sys}$ 使得 $supp(s_k,\{\pi_{C_k'}\})$,那么,SC^R 就称为一个成功的服务核恢复步.

定义 4(安全的服务核恢复步). 给定服务核恢复步 $SC^R=(S^{R-sys},C^{R-sys})$,如果对$\forall s_i\in S^{sys}$ 都存在 $\{\pi_{C_i}\}\subseteq C^{R-sys}$ 使得 $supp(s_k,\{\pi_{C_k'}\})$,那么,SC^R 就称为一个安全的服务核恢复步.

我们记 $R=SC^{R_1}\circ...\circ SC^{R_m}$ 为服务核 SC 和 SC'间的一个服务核恢复步序列,其中每一个 SC^{R_j} ($SC^{R_j}\in R$, $1\le j\le m$)都是一个服务核恢复步.如果服务核 SC 经过服务核恢复步序列 R 到达服务核 SC',且 $S^{R_m-sys}=(S^{R-sys})'$,那么该过程称为服务核恢复,记为 $SC\xrightarrow{R}SC'$.对于一个服务核恢复步序列来说,只要最后一个服务核恢复步的服务集合与某个服务核的服务集合相同了,该服务核恢复步序列就是一个服务核恢复,而不需要相应的支撑构件也完全相同.因为不是所有的服务核恢复步都能到达可生存性规范所定义的服务核,所以不是所有的系统恢复过程都是服务核恢复.

定义 5(成功的服务核恢复). 如果每一个 $SC^{R_j}(SC^{R_j}\in R,1\le j\le m)$ 都是成功的服务核恢复步,那么 $SC\xrightarrow{R}SC'$ 称为一个成功的服务核恢复.

定义 6(安全的服务核恢复). 如果每一个 $SC^{R_j}(SC^{R_j}\in R,1\le j\le m)$ 都是安全的服务核恢复步,那么 $SC\xrightarrow{R}SC'$ 称为一个安全的服务核恢复.

2.2 构件的安装与安装的执行

2.2.1 构件安装策略

构件安装策略决定一个构件如何在构件集合中进行安装.在定义构件安装策略之前,首先引入这样两个操作:插入操作 $Insert(c_i^p,C_k)$,把构件 c_i^p 作为一个新元素插入到构件集合 C_k 中;替换操作 $Repalce(c_i^p,C_k)$,用构件 c_i^p 替换构件集合 C_k 中所有具有构件家族号 i 的构件.

定义 7(安装策略). 给定 $supp(s_k,\{\pi_{C_k}\})$ 和 $SC=(S^{sys},C^{sys})$,安装策略 IP_i^p 表示一个构件 $c_i^p\in\{\pi_{C_k}\}$ 如何安装到 S^{sys} 中.定义如下 4 种安装策略:

- 直接安装策略 DIP(direct installation policy),即 $IP_i^p=DIP$.执行操作 $Insert(c_i^p,C^{sys})$.
- 完全安装策略 TIP(total installation policy),即 $IP_i^p=TIP$.如果 $C^{sys}\cap C_{F(i)}=\varnothing$,则执行操作 $Insert(c_i^p,C^{sys})$;如果 $C^{sys}\cap C_{F(i)}\neq\varnothing$,则执行操作 $Replace(c_i^p,C^{sys}\cap C_{F(i)})$.
- 新版本安装策略 INP(installation of new-version policy),即 $IP_i^p=INP$.如果 $C^{sys}\cap C_{F(i)}=\varnothing$,则执行操作 $Insert(c_i^p,C^{sys})$;如果 $\{c_i^q\mid c_i^q\in C^{sys}\cap C_{F(i)},q\le p\}\neq\varnothing$,则执行操作 $Repalce(c_i^p,\{c_i^q\mid c_i^q\in C^{sys}\cap C_{F(i)},q\le p\})$;否则,什么操作也不执行.
- 空安装策略 NIP(none installation policy),即 $IP_i^p=NIP$.如果 $C^{sys}\cap C_{F(i)}=\varnothing$,则执行操作 $Insert(c_i^p,C^{sys})$;否则,什么操作都不执行.

很明显,构件安装策略 IP_i^p 只会影响构件集合$\{\pi_{C_k}\}$中具有构件家族号 i 的构件.

2.2.2 安装的执行

引入构件的兼容性是为了体现同一构件家族中具有不同版本号的各个构件之间的关系.

定义 8(向后兼容). 给定 $supp(s_k,\{\pi_{C_k}\})$,并记$\{\pi'_{C_k}\}$为构件集合$\{\pi_{C_k}\}$执行完操作 $Replace(c_i^p,\{c_i^r\mid c_i^r\in\{\pi_{C_k}\},p\ge r\})$之后的构件集合.如果 $supp(s_k,\{\pi'_{C_k}\})$ 成立,则称构件 c_i^p 具有向后兼容性(backward compatibility,简称 BC).

定义 9(向前兼容). 给定 $supp(s_k,\{\pi_{C_k}\})$,并记$\{\pi'_{C_k}\}$为构件集合$\{\pi_{C_k}\}$执行完操作 $Replace(c_i^p,\{c_i^r\mid c_i^r\in\{\pi_{C_k}\},p<r\})\}$之后的构件集合.如果 $supp(s_k,\{\pi'_{C_k}\})$ 成立,则称构件 c_i^p 具有向前兼容性(forward compatibility,简称 FC).

通俗地说,构件 c_i^p 具有向后兼容性意味着构件 c_i^p 能够实现构件家族 i 中版本号不比 p 大的那些构件 c_i^r ($p\geq r$)的功能;构件 c_i^p 具有向前兼容性意味着构件 c_i^p 能够实现构件家族 i 中版本号比 p 大的那些构件 c_i^r ($p<r$)的功能.

本文中涉及到的构件间兼容关系只有 3 种,即向前兼容(FC)、向后兼容(BC)和既向前兼容又向后兼容($BC\wedge FC$).对于构件家族中可能出现的与其他构件既不向前兼容也不向后兼容的构件,可以把该构件从构件家族中分离出来,形成一个新的构件家族.

定义 10(安装表达式). 构件 c_i^p 的安装表达式是一个三元组 $(c_i^p, Comp_i^p, IP_i^p)$,表示具有兼容性 $Comp_i^p$ 的构件 c_i^p 按照安装策略 IP_i^p 进行安装.

按照安装策略 IP_i^p 进行安装之后,如果构件 c_i^p 插入到了 S^{sys} 中,则称安装表达式 $(c_i^p, Comp_i^p, IP_i^p)$ 具有插入效果;否则称为具有未插入效果.插入效果包括直接插入和替换插入.

为了讨论方便,安装表达式 $(c_i^p, \_, \_)$ 表示构件 c_i^p 的兼容性和安装策略可以是任意值,其他情况类推.

定义 11(安装执行). 给定构件排列 π_{C_k},那么,π_{C_k} 的安装执行是 π_{C_k} 中每个构件的安装表达式的相应排列,记为 $IE(\pi_{C_k})$.

与前面的记号相统一,我们用 $\{IE(\pi_{C_k})\}$ 来表示安装执行 $IE(\pi_{C_k})$ 中的所有安装表达式的集合.如果安装执行 $IE(\pi_{C_k})$ 中有 n 个安装表达式,那么安装执行 $IE(\pi_{C_k})$ 的长度就为 n.

2.3 构件的安装与安装的执行

对于构件系统的部署来说,部署的规模越大,相应的安装执行就会越复杂.例如,Dassault 系统的主要产品 CATIA 就由好几百个构件构成[22],如果把这些构件的不同版本都考虑进去,则其相应的安装执行会更复杂.所以,如何简化构件系统的部署规模是本文模型能否在实际中应用的重要影响因素.下面通过引入安装执行投影的概念来简化复杂的安装执行.

安装执行 $IE(\pi_{C_k})$ 在构件家族 $C_{F(i)}$ 上的投影是一个安装表达式的排列,记为 $Proj_{F(i)}^{IE(\pi_{C_k})}$.它由 $IE(\pi_{C_k})$ 中具有构件家族号 i 的安装表达式,按照它们在 $IE(\pi_{C_k})$ 中的相应位置排列而成.

定义 12(等价执行效果). 如果 S^{sys} 中的构件按照安装执行 $IE(\pi_{C_k})$ 和 $IE(\pi'_{C_k})$ 分别执行后,所得到的构件集合相同,则称 $IE(\pi_{C_k})$ 和 $IE(\pi'_{C_k})$ 在 S^{sys} 上有等价的执行效果,记为 $Effect_{S^{sys}}(IE(\pi_{C_k})) = Effect_{S^{sys}}(IE(\pi'_{C_k}))$.

用 N_{C_k} 记录 S_k 中不同服务家族的个数,用 $P(\pi_{C_k})$ 表示这 N_{C_k} 个服务家族号按照它们在 π_{C_k} 中出现的顺序排列.关于安装执行的效果,我们有如下性质:

性质 1. 如果记 $P(\pi_{C_k}) = f_1 f_2 \ldots f_{N_{C_k}}$,则 $Effect_{S^{sys}}(IE(\pi_{C_k})) = Effect_{S^{sys}}(IE(Proj_{F(f_1)}^{IE(\pi_{C_k})} \ldots Proj_{F(f_{N_{C_k}})}^{IE(\pi_{C_k})}))$.

因为安装表达式 $(c_{f_i}^p, Comp_{f_i}^p, IP_{f_i}^p) \in IE(\pi_{C_k})(f_1 \leq f_i \leq f_{N_{C_k}})$ 只会影响构件集合 $\{\pi_{C_k}\}$ 中具有构件家族号 f_i 的构件,所以性质 1 显然成立.

性质 2. 给定 $\forall \pi_{C_k}, \pi'_{C_k} \in \Pi(C_k)$ 且 $\pi_{C_k} \neq \pi'_{C_k}$,如果 $P(\pi_{C_k}) = P(\pi'_{C_k})$,那么,

$$Effect_{S^{sys}}(IE(\pi_{C_k})) = Effect_{S^{sys}}(IE(\pi'_{C_k})).$$

证明:令 $P(\pi_{C_k}) = P(\pi'_{C_k}) = f_1 f_2 \ldots f_{N_{C_k}}$.因为 $\forall \pi_{C_k}, \pi'_{C_k} \in \Pi(C_k)$ 且 $P(\pi_{C_k}) = P(\pi'_{C_k})$,根据性质 1,有

$$\begin{aligned} Effect_{S^{sys}}(IE(\pi_{C_k})) &= Effect_{S^{sys}}(IE(Proj_{F(f_1)}^{IE(\pi_{C_k})} \ldots Proj_{F(f_{N_{C_k}})}^{IE(\pi_{C_k})})) \\ &= Effect_{S^{sys}}(IE(Proj_{F(f_1)}^{IE(\pi'_{C_k})} \ldots Proj_{F(f_{N_{C_k}})}^{IE(\pi'_{C_k})})) \\ &= Effect_{S^{sys}}(IE(\pi'_{C_k})), \end{aligned}$$

所以性质 2 得证. □

性质 3. 如果 $C^{sys} \cap C_{F(i)} \neq \varnothing$,那么 $Effect_{S^{sys}}(Proj_{F(i)}^{IE(\pi_{C_k})}) = Effect_{S^{sys}}(Proj_{F(i)}^{IE(\pi_{C_k})} \setminus (\_, \_, NIP))$.这里,$Proj_{F(i)}^{IE(\pi_{C_k})} \setminus (_, _, NIP)$表示去掉所有形如$(_, _, NIP)$的安装表达式之后的安装执行投影 $Proj_{F(i)}^{IE(\pi_{C_k})}$.

因为“空安装策略 *NIP*”不执行任何动作,不会对安装投影的效果造成任何影响,所以性质 3 显然成立.

性质 4. 对于 $Proj_{F(i)}^{IE(\pi_{C_k})}=(c_i^{t_1},Comp_i^{t_1},IP_i^{t_1})\circ\ldots\circ(c_i^{t_h},Comp_i^{t_h},IP_i^{t_h})\circ\ldots\circ(c_i^{t_a},Comp_i^{t_a},IP_i^{t_a})$,如果 $(s_i^{t_h},Comp_i^{t_h},IP_i^{t_h})$ 是 $Proj_{F(i)}^{IE(\pi_{C_k})}$ 中最后一个形如(_,_,*TIP*)的安装表达式,那么,

$$Effect_{S^{sys}}(Proj_{F(i)}^{IE(\pi_{C_k})})=Effect_{S^{sys}}((c_i^{t_h},Comp_i^{t_h},IP_i^{t_h})\circ\ldots\circ(c_i^{t_a},Comp_i^{t_a},IP_i^{t_a})).$$

因为 $Proj_{F(i)}^{IE(\pi_{C_k})}$ 中最后一个形如(_,_,*TIP*)的安装表达式 $(s_i^{t_h},Comp_i^{t_h},IP_i^{t_h})(IP_i^{t_h}=TIP)$ 将消除前面所有形如 $(s_i^{t_l},Comp_i^{t_l},IP_i^{t_l})$ $(1\le l\le h-1,h\ge 2)$的安装表达式对安装投影的影响,所以,这些安装表达式可以被去除,性质 4 显然成立.

基于以上关于安装执行效果的性质,我们给出算法 1 来简化安装执行 $IE(\pi_{C_k})$.该算法的主要思想是:首先获取构件家族号的排列 $P(\pi_{C_k})$ (算法 2),然后把安装执行 $IE(\pi_{C_k})$ 在 $P(\pi_{C_k})$ 中的每一个服务家族上进行投影(算法 3),接着利用上述性质对每一个安装执行投影进行简化(算法 4),最后把简化后的安装执行 $IE(\pi_{C_k})$ 中的安装表达式按照它们在 $IE(\pi_{C_k})$ 中的相应顺序进行排列,得到最终简化的安装执行.

算法 1. 安装执行简化的主过程.

Procedure *SIMPLIFY_IE*($IE(\pi_{C_k})$)

输入:安装执行 $IE(\pi_{C_k})$;

输出:简化的安装执行 $IE(\pi_{C_k})$*_Simplified* .

$IE(\pi_{C_k})$*_Simplified* ← *NULL*

$P(\pi_{C_k})\leftarrow$ *PERMUTATION_FAMILY_NUMBER_ IE*($IE(\pi_{C_k})$)

for 每一个家族号 $f_i\in P(\pi_{C_k})$ do

{ $Proj_{F(f_i)}^{IE(\pi_{C_k})}\leftarrow$ *IE_PROJECTION*($IE(\pi_{C_k})$, f_i)

$Proj_{F(f_i)}^{IE(\pi_{C_k})}$*_Simplified* ← *SIMPLIFY_PROJECTION*($Proj_{F(f_i)}^{IE(\pi_{C_k})}$)

$IE(\pi_{C_k})$*_Simplified* ← $IE(\pi_{C_k})$*_Simplified* $\circ$ $Proj_{F(f_i)}^{IE(\pi_{C_k})}$*_Simplified* }

end for

把 $IE(\pi_{C_k})$*_Simplified* 中的安装表达式按照它们在 $IE(\pi_{C_k})$ 中的相应顺序进行排列

return ($IE(\pi_{C_k})$*_Simplified*)

end procedure

算法 2. 获取安装执行的组件家族号排列的过程.

Procedure *PERMUTATION_FAMILY_NUMBER_ IE*($IE(\pi_{C_k})$)

输入: $IE(\pi_{C_k})=(s_{l_1}^{q_1},C_{l_1}^{q_1},IS_{l_1}^{q_1})\circ\ldots\circ(s_{l_j}^{q_j},C_{l_j}^{q_j},IS_{l_j}^{q_j})\circ\ldots\circ(s_{l_n}^{q_n},C_{l_n}^{q_n},IS_{l_n}^{q_n})$;

输出: $P(\pi_{C_k})$.

$P(\pi_{C_k})\leftarrow$ *NULL*

for *i*=1 to *n* do

if $l_i\notin P(\pi_{C_k})$

then do $P(\pi_{C_k})\leftarrow P(\pi_{C_k})\circ l_i$

end if

end for

return ($P(\pi_{C_k})$)

end procedure

算法 3. 安装执行在服务家族上投影的过程.

Procedure *IE_PROJECTION*($IE(\pi_{C_k})$,*i*)

输入:安装执行 $IE(\pi_{C_k})=(s_{l_1}^{q_1},C_{l_1}^{q_1},IS_{l_1}^{q_1})\circ...\circ(s_{l_j}^{q_j},C_{l_j}^{q_j},IS_{l_j}^{q_j})\circ...\circ(s_{l_n}^{q_n},C_{l_n}^{q_n},IS_{l_n}^{q_n})$,组件家族号 i;

输出:安装执行在组件家族 i 上的投影 $Proj_{F(i)}^{IE(\pi_{C_k})}$.

$Proj_{F(i)}^{IE(\pi_{C_k})} \leftarrow NULL$

for t=1 to n do

if $l_t=i$

then $Proj_{F(i)}^{IE(\pi_{C_k})} \leftarrow Proj_{F(i)}^{IE(\pi_{C_k})}\circ(s_{l_t}^{q_t},C_{l_t}^{q_t},IS_{l_t}^{q_t})$

end if

end for

return ($Proj_{F(i)}^{IE(\pi_{C_k})}$)

end procedure

算法 4. 简化安装执行在服务家族上投影的过程.

Procedure $SIMPLIFY\_PROJECTION(Proj_{F(i)}^{IE(\pi_{C_k})})$

输入: $Proj_{F(i)}^{IE(\pi_{C_k})}=(c_i^{t_1},Comp_i^{t_1},IP_i^{t_1})\circ...\circ(c_i^{t_h},Comp_i^{t_h},IP_i^{t_h})\circ...\circ(c_i^{t_a},Comp_i^{t_a},IP_i^{t_a})$

输出: $Proj_{F(i)}^{IE(\pi_{C_k})}\_Simplified$

$Proj_{F(i)}^{IE(\pi_{C_k})}\_Simplified \leftarrow Proj_{F(i)}^{IE(\pi_{C_k})}$,$L$=1

for j=1 to a do

if $(c_i^{t_j},Comp_i^{t_j},IP_i^{t_j})$ 具有 $(\_,\_,NIP)$ 的形式且 $C^{sys}\bigcap C_{F(i)}\neq\varnothing$

then do 消除 $Proj_{F(i)}^{IE(\pi_{C_k})}\_Simplified$ 中的 $(c_i^{t_j},Comp_i^{t_j},IP_i^{t_j})$

else if $(c_i^{t_j},Comp_i^{t_j},IP_i^{t_j})$ 具有 $(\_,\_,TIP)$ 形式 且 $L<j$

then do

{ 消除 $Proj_{F(i)}^{IE(\pi_{C_k})}\_Simplified$ 中的 $(c_i^{t_L},Comp_i^{t_L},IP_i^{t_L})$ $L\leftarrow j$ }

end do

end if

end if

end for

return ($Proj_{F(i)}^{IE(\pi_{C_k})}\_Simplified$)

end procedure

显然,算法 1 的时间复杂性为 $O(n+m\times(n+a))=O(n^2)$,其中,n 为安装执行 $IE(\pi_{C_k})$ 的长度.

3 基于形式化模型的构件系统恢复推理

本节通过对简化的安装执行 $IE(\pi_{C_k})\_Simplified$ 介绍一些规则来分析和推理构件系统恢复的性质.

规则 1. 如果 $\{IE(\pi_{C_k})_Simplified\}$中的任意安装表达式 $(c_i^p,Comp_i^p,IP_i^p)$ 都具有 $(c_i^p,\_,DIP)$ 或 $(c_i^p,\_,TIP)$ 的形式,那么 $SC^R=(S^{R-sys},C^{R-sys})$是一个成功的服务核恢复步.

规则 1 可以直接由“直接安装策略 *DIP*”和“完全安装策略 *TIP*”的定义得出,证明从略.

规则 2. 如果安装执行 $\{IE(\pi_{C_k})_Simplified\}$满足以下两个条件:

- $\{IE(\pi_{C_k})\_Simplified\}$ 中的每个具有“未插入效果”的安装表达式都具有 $(c_i^p,\_,NIP)$ 形式,且 $C^{R-sys}\cap C_{F(i)}$ 中的任意构件都具有 $BC\wedge FC$ 的兼容性;
- $\{IE(\pi_{C_k})\_Simplified\}$ 中的每个具有“未插入效果”的安装表达式都具有 $(c_i^p,\_,INP)$ 形式,$C^{R-sys}\cap C_{F(i)}$ 中的任意构件都具有 BC 的兼容性.

那么,$SC^R=(S^{R-sys},C^{R-sys})$是一个成功的服务核恢复步.

证明:对于$\{IE(\pi_{C_k})_Simplified\}$中的任意一个安装表达式$(c_i^p,Comp_i^p,IP_i^p)$,它或者是具有"插入效果"或者是具有"未插入效果".下面分别讨论这两种情况:

i) 安装表达式$(c_i^p,Comp_i^p,IP_i^p)$具有"插入效果"

在这种情况下,需要用来支持新启动的服务构件 s_k 的构件 c_i^p 已经被插入到 C^{R-sys} 中了,即需要用来支持新启动的服务构件 s_k 的构件 c_i^p 已经包括在 C^{R-sys} 中了.

ii) 安装表达式$(c_i^p,Comp_i^p,IP_i^p)$具有"未插入效果"

在这种情况下,需要分别讨论规则 2 中给出的两个条件:(a) 在第 1 个条件中,构件 c_i^p 因为"空安装策略 *NIP*"而没有被插入到 C^{R-sys} 中.但是,$(C^{R-sys}\cap C_{F(i)})$中的任意构件 c_i^q 都具有 $BC\wedge FC$ 的兼容性,这可以保证 c_i^q 能够完成 c_i^p 的功能.所以,需要支持 s_k 的构件 c_i^q 已经包括在 C^{R-sys} 中了;(b) 在第 2 个条件中,构件 c_i^p 因为"新版本安装策略 *INP*"而没有被插入到 s_k 中,即存在 $c_i^q\in(C^{R-sys}\cap C_{F(i)})$ $(p<q)$.但是,$(C^{R-sys}\cap C_{F(i)})$中的任意构件都具有 *BC* 的兼容性,这确保了 c_i^q 能够完成 c_i^p 的功能.所以需要支持 s_k 的构件 c_i^q 已经包括在 C^{R-sys} 中了.

因此,不论是在情况 i)还是情况 ii)下,关于构件 c_i^p 需要支持 s_k 的构件都包括在 C^{R-sys} 中了.由于$(c_i^p,Comp_i^p,IP_i^p)$在安装执行$\{IE(\pi_{C_k})_Simplified\}$中的任意性,所以,所有需要支持 s_k 的构件都包括在 C^{R-sys} 中,则必然存在$\{\pi_{C_k'}\}\subseteq C^{R-sys}$,使得 $supp(s_k,\{\pi_{C_k'}\})$.根据定义 4 可得,$SC^R=(S^{R-sys},C^{R-sys})$是一个成功的服务核恢复步,即规则 2 成立. □

规则 3. 如果$\{IE(\pi_{C_k})_Simplified\}$中的任意安装表达式$(c_i^p,Comp_i^p,IP_i^p)$都具有$(c_i^p,_,DIP)$或$(c_i^p,_,NIP)$的形式,那么,$SC^R=(S^{R-sys},C^{R-sys})$是一个安全的服务核恢复步.

规则 3 可以直接由"直接安装策略 *DIP*"和"空安装策略 *NIP*"的定义得出,证明从略.

规则 4. 如果$\{IE(\pi_{C_k})_Simplified\}$中的每个具有"插入效果"任意安装表达式$(c_i^p,Comp_i^p,IP_i^p)$都具有$(c_i^p,BC\wedge FC,TIP)$或$(c_i^p,BC,INP)$形式,服务核恢复步 $SC^R=(S^{R-sys},C^{R-sys})$是一个安全的服务核恢复步.

证明:对于安装执行$IE(\pi_{C_k})$中的任意一个安装表达式$(c_i^p,Comp_i^p,IP_i^p)$,它或者是具有"插入效果"或者是具有"未插入效果".下面分别讨论这两种情况:

i) 安装表达式$(c_i^p,Comp_i^p,IP_i^p)$具有"插入效果"

在这种情况下,需要分别讨论规则 4 中给出的两种安装表达式的形式:(a) 安装表达式$(c_i^p,Comp_i^p,IP_i^p)$具有$(c_i^p,BC\wedge FC,TIP)$形式.构件 c_i^p 因为"完全安装策略 *TIP*"替换了$(C^{R-sys}\cap C_{F(i)})$中的所有构件而插入到 C^{R-sys} 中.但是,c_i^p 具有 $BC\wedge FC$ 的兼容性,这可以保证 c_i^p 能够完成$(C^{R-sys}\cap C_{F(i)})$中任意一个构件的功能.所以,需要支持 $s_l(s_l\in S^{R-sys},1\le l\le k-1)$的构件已经包括在 C^{R-sys} 中了;(b) 安装表达式$(c_i^p,Comp_i^p,IP_i^p)$具有(c_i^p,BC,INP)形式.构件 c_i^p 因为"新版本安装策略 *INP*"替换了$(C^{R-sys}\cap C_{F(i)})$中版本号比 *p* 小的所有构件而插入到 C^{R-sys} 中.但是,c_i^p 具有 *BC* 的兼容性,这可以保证 c_i^p 能够完成$(C^{R-sys}\cap C_{F(i)})$中任意一个版本号比 *p* 小的构件的功能.所以,需要支持 s_l 的构件也已经包括在 C^{R-sys} 中了.

ii) 安装表达式$(c_i^p,Comp_i^p,IP_i^p)$具有"未插入效果"

在这种情况下,原来支持已有服务 s_l 的服务构件没有发生任何变化,即需要用来支持 s_l 的所有构件已经包括在 C^{R-sys} 中了.

因此,不论是在情况 i)还是情况 ii)下,关于构件 c_i^p 需要支持 s_l 的构件都包括在 C^{R-sys} 中了.由于$(c_i^p,Comp_i^p,IP_i^p)$在安装执行 $IE(\pi_{C_k})\_Simplified$ 中的任意性,所以,对 $s_l(s_l\in S^{R-sys},1\le l\le k-1)$来说,所有需要支持 s_l 的构件都包括在 C^{R-sys} 中,则必然存在$\{\pi_{C_l'}\}\subseteq C^{R-sys}$,使得 $supp(s_l,\{\pi_{C_l'}\})$.由定义 5 可得,$SC^R=(S^{R-sys},C^{R-sys})$是一个安全的服务核恢复步,即规则 4 成立. □

规则 5. 如果 $\forall(c_i^p, Comp_i^p, IP_i^p) \in \{IE(\pi_{C_k})\}$ 都具有 $(c_i^p, \_, DIP)$ 形式,那么,$SC^R=(S^{R-sys}, C^{R-sys})$是一个成功且安全的服务核恢复步.

规则 5 可以直接由规则 3 和规则 4 推理得出,证明从略.

规则 6. 给定服务核恢复 $SC \xrightarrow{R} SC'$,对 R 中的每一个服务核恢复步 SC^{R_j} $(SC^{R_j} \in R, 1 \le j \le m)$,

- 如果 SC^{R_j} 满足规则 1 或规则 2,那么, $SC \xrightarrow{R} SC'$ 是一个成功的服务核恢复;
- 如果 SC^{R_j} 满足规则 3 或规则 4,那么, $SC \xrightarrow{R} SC'$ 是一个安全的服务核恢复;
- 如果 SC^{R_j} 满足规则 5,那么, $SC \xrightarrow{R} SC'$ 是一个成功且安全的服务核恢复.

规则 6 实际上是上述 5 个规则的推论,证明从略.

有了对系统恢复性质的推理规则,下面给出基于可生存性规范的系统恢复分析流程,如图 3 所示.它具体包括以下 4 个步骤:

第 1 步:确定构件系统所能提供的用户能够感知的所有服务.这可以通过利用 MSC(message sequence charts)[23]或 LSC(live sequence charts)[24]对构件系统的描述,来抽取所有的服务.

第 2 步:根据可能出现的不同攻击、故障或者事故,确定构件系统的服务核.本文采用一种比较简单的服务核确定方法,即对第 1 步确定的每一个服务找出它的所有子服务来构成相应的服务核.

第 3 步:根据服务核建立构件系统的可生存性规范.可生存性规范定义了构件系统在不同的情况下所处的系统状态.

第 4 步:根据可生存性规范和推理规则对给定的系统恢复的性质(成功性和安全性)进行推理.

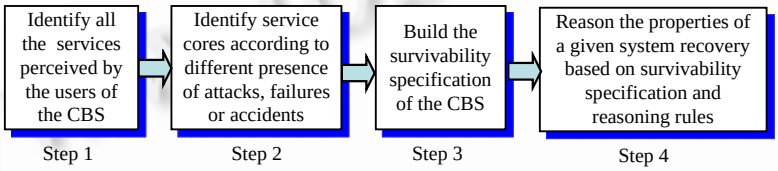

Fig.3 Analysis process of component-based system recovery based on survivability specification

图 3 基于可生存性规范的构件系统恢复分析流程

4 模型的应用

本节将展示基于可生存性规范的形式化模型在移动视频点播系统 MVoD 的系统恢复中是如何应用的.该系统所涉及的 11 个构件家族以及每个构件家族所包含的不同版本的构件见表 1.

Table 1 Component families and components with different versions

表 1 构件家族及其不同版本的构件

Component family	Component with different versions	Component family	Component with different versions
$C_{F(0)}$	$\{c_0^1, c_0^2\}$	$C_{F(6)}$	$\{c_6^1, c_6^2, c_6^3\}$
$C_{F(1)}$	$\{c_1^1, c_1^2, c_1^3\}$	$C_{F(7)}$	$\{c_7^1, c_7^2\}$
$C_{F(2)}$	$\{c_2^1, c_2^2, c_2^3, c_2^4\}$	$C_{F(8)}$	$\{c_8^1, c_8^2, c_8^3, c_8^4\}$
$C_{F(3)}$	$\{c_3^1, c_3^2, c_3^3\}$	$C_{F(9)}$	$\{c_9^1, c_9^2\}$
$C_{F(4)}$	$\{c_4^1, c_4^2\}$	$C_{F(10)}$	$\{c_{10}^1, c_{10}^2, c_{10}^3, c_{10}^4, c_{10}^5\}$
$C_{F(5)}$	$\{c_5^1, c_5^2, c_5^3, c_5^4, c_5^5\}$	$C_{F(11)}$	$\{c_{10}^1, c_{10}^2\}$

下面根据第 3 节给出的“构件系统恢复分析流程”对 MVoD 系统中的服务核恢复步序列 $R = SC^{R_1} \circ SC^{R_2} \circ SC^{R_3}$ 进行系统恢复性质的推理(如图 4 所示).

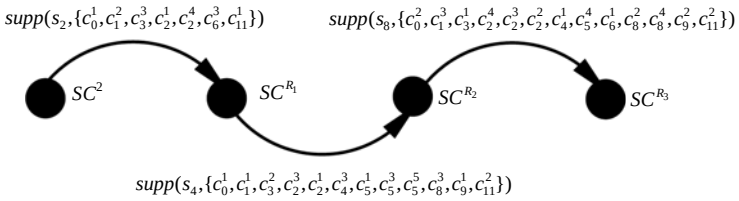

Fig.4 Service core recovery step sequence R

图 4 服务核恢复步序列 R

首先确定用户能够感知的所有服务,即 $s_1,s_2,s_3,s_4,s_5,s_6,s_7,s_8,s_9,s_{10},s_{11}$,如图 2 所示.接着,根据不同类型的攻击、故障和事故,把 MVoD 系统所能提供的这 11 个服务组合成 11 个服务核(见表 2).然后,根据服务核建立系统的可生存性规范 $SPEC=\{SC^1,SC^2,SC^3,SC^4,SC^5,SC^6,SC^7,SC^8,SC^9,SC^{10},SC^{11}\}$.

Table 2 Service cores of the MVoD system

表 2 MVoD 系统的服务核

Service core (SC)	Service set (S^{sys})	Supporting component set (C^{sys})
SC^1	$\{s_1\}$	$\{c_0^1,c_1^2,c_3^1,c_3^3,c_2^1\}$
SC^2	$\{s_1,s_2\}$	$\{c_0^1,c_1^1,c_1^2,c_3^2,c_2^3,c_6^3,c_{11}^2\}$
SC^3	$\{s_1,s_3\}$	$\{c_0^1,c_1^1,c_3^2,c_2^2,c_4^2,c_5^2,c_5^4\}$
SC^4	$\{s_1,s_3,s_4\}$	$\{c_0^2,c_1^2,c_3^1,c_2^4,c_5^1,c_8^1,c_8^2,c_9^1\}$
SC^5	$\{s_1,s_3,s_5\}$	$\{c_0^1,c_1^3,c_3^1,c_3^3,c_2^1,c_4^1,c_5^1,c_5^3,c_8^3,c_8^1,c_{10}^4,c_{10}^3,c_{11}^1\}$
SC^6	$\{s_1,s_3,s_4,s_6\}$	$\{c_0^2,c_1^2,c_3^2,c_2^2,c_2^3,c_4^3,c_5^1,c_5^5,c_8^2,c_9^1,c_{10}^2,c_{11}^2\}$
SC^7	$\{s_1,s_3,s_4,s_5,s_6,s_7\}$	$\{c_0^1,c_0^2,c_1^2,c_3^1,c_3^2,c_2^2,c_4^2,c_5^5,c_7^1,c_8^2,c_8^4,c_{10}^2,c_{10}^4,c_9^2,c_{11}^1\}$
SC^8	$\{s_1,s_2,s_3,s_4,s_8\}$	$\{c_0^2,c_1^3,c_3^3,c_2^3,c_2^3,c_4^3,c_5^3,c_6^2,c_6^1,c_8^4,c_8^1,c_9^1,c_{11}^1\}$
SC^9	$\{s_1,s_2,s_3,s_5\}$	$\{c_0^2,c_1^2,c_3^1,c_2^3,c_4^3,c_5^3,c_5^1,c_6^2,c_8^4,c_8^1,c_{10}^1,c_{10}^5,c_{11}^2\}$
SC^{10}	$\{s_1,s_2,s_3,s_4,s_5,s_6,s_8,s_9\}$	$\{c_0^2,c_1^3,c_3^2,c_2^3,c_4^1,c_5^1,c_5^5,c_6^2,c_7^2,c_8^2,c_8^4,c_{10}^5,c_{10}^3,c_{11}^2\}$
SC^{11}	$\{s_1,s_2,s_3,s_4,s_5,s_6,s_7,s_8,s_9,s_{10},s_{11}\}$	$\{c_0^2,c_1^2,c_3^1,c_2^3,c_2^1,c_4^3,c_4^1,c_4^2,c_5^3,c_5^1,c_5^5,c_6^3,c_8^1,c_7^1,c_7^2,c_{10}^1,c_{10}^5,c_{10}^3,c_{11}^1\}$

系统恢复的最关键一步就是根据服务核恢复步序列,从起始服务核 $SC^3=(S^{3-sys},C^{3-sys})=(\{s_1,s_3\},\{c_0^1,c_1^1,c_3^2,c_2^2,c_4^2,c_5^2,c_5^4\})$一步步地推理系统恢复的整个过程和性质.首先考虑第 1 个服务核恢复步 SC^{R_1}.起始服务核 SC^3 中的每个构件的兼容性以及 $supp(s_2,\{c_0^1,c_1^2,c_3^3,c_2^1,c_2^4,c_6^3,c_{11}^1\})$ 中的安装表达式见表 3、表 4 中所列.所以,相应的安装执行是

$$IE(\pi_{C_2})=(c_0^1,BC\wedge FC,NIP)\circ(c_1^2,BC,DIP)\circ(c_3^3,BC,INP)\circ(c_2^1,FC,INP)\circ$$
$$(c_2^4,BC\wedge FC,TIP)\circ(c_6^3,FC,TIP)\circ(c_{11}^1,BC,INP).$$

根据安装执行化简算法,先求出构件家族号排列为 0∘1∘3∘2∘6∘11,接着把安装执行 $IE(\pi_{C_2})$ 依次在构件家族号上进行投影,根据安装执行的效果的性质对这些投影进行化简:因为 $C^{sys-2}\cap C_{F(0)}\neq\varnothing$且 $IP_0^1=NIP$,由性质 3 可知, $Proj_{F(0)}^{IE(\pi_{C_2})}=(c_0^1,BC\wedge FC,NIP)$ 可以去除;因为 $Proj_{F(2)}^{IE(\pi_{C_2})}=(c_2^1,FC,INP)\circ(c_2^4,BC\wedge FC,TIP)$ 中(c_2^4 ,$BC\wedge FC,TIP$)是 $Proj_{F(2)}^{IE(\pi_{C_2})}$ 中最后一个形如(_,_,TIP)的安装表达式,由性质 4 可知,(c_2^1,FC,INP) 可以去除;而其他安装式投影都不能化简了.所以,简化的安装执行为

$$IE(\pi_{C_2})\_Simplified=(c_1^2,BC,DIP)\circ(c_3^3,BC,INP)\circ(c_2^4,BC\wedge FC,TIP)\circ$$
$$(c_6^3,FC,TIP)\circ(c_{11}^1,BC,INP).$$

按照 $IE(\pi_{C_2})\_Simplified$ 中的安装表达式依次进行构件的安装,得到 $SC^{R_1}=(S^{R_1-sys},C^{R_1-sys})=(\{s_1,s_2,s_3\},$

$\{c_0^1,c_1^1,c_1^2,c_3^3,c_2^4,c_4^2,c_5^2,c_5^4,c_6^3,c_{11}^1\}$).因为 $IE(\pi_{C_2})\_Simplified$ 所涉及到的构件 $c_1^2,c_3^3,c_2^4,c_6^3,c_{11}^1$ 全部都插入到了 C^{R_1-sys} 中,由规则 2 可知, SC^{R_1} 是一个成功的服务核恢复步.

Table 3 Components and their compatibilities of SC^3

表 3 SC^3 中的构件及其兼容性

Component	Compatability
c_0^1	$BC\wedge FC$
c_1^1	$BC\wedge FC$
c_3^2	FC
c_2^2	$BC\wedge FC$
c_4^2	$BC\wedge FC$
c_5^2	FC
c_5^4	BC

Table 4 Installation expressions of $supp(s_2,\{c_0^1,c_1^2,c_3^3,c_2^1,c_2^4,c_6^3,c_{11}^1\})$

表 4 $supp(s_2,\{c_0^1,c_1^2,c_3^3,c_2^1,c_2^4,c_6^3,c_{11}^1\})$ 的安装表达式

Component (c_i^p)	Compatability ($comp_i^p$)	Installation policy (IP_i^p)
c_0^1	$BC\wedge FC$	NIP
c_1^2	BC	DIP
c_3^3	BC	INP
c_2^1	FC	INP
c_2^4	$BC\wedge FC$	TIP
c_6^3	FC	TIP
c_{11}^1	BC	INP

类似地,我们接着依次考察服务核恢复步 SC^{R_2} 和 SC^{R_3}. C^{R_1-sys} 中构件的兼容性和 $supp(s_4,\{c_0^1,c_1^1,c_3^2,c_2^3,c_2^1,c_4^3,c_5^1,c_5^3,c_5^5,c_8^3,c_9^1,c_{11}^2\})$ 中的安装表达式见表 5 和表 6; C^{R_2-sys} 中构件的兼容性和 $supp(s_8,\{c_0^2,c_1^3,c_3^1,c_2^4,c_2^3,c_2^2,c_4^1,c_5^4,c_6^1,c_8^2,c_8^4,c_9^2,c_{11}^2\})$ 中的安装表达式见表 7 和表 8.按照上述过程,利用相应的安装执行的效果性质,分别对这两个安装表达式进行化简,可以得到:

$$\begin{aligned}IE(\pi_{C_4})\_Simplified=&(c_0^2,FC,DIP)\circ(c_2^3,FC,INP)\circ(c_2^1,FC,DIP)\circ(c_4^3,BC\wedge FC,INP)\circ\\&(c_5^5,BC\wedge FC,TIP)\circ(c_8^3,BC,DIP)\circ(c_9^1,BC,NIP)\circ(c_{11}^2,BC,INP),\\IE(\pi_{C_8})\_Simplified=&(c_0^1,BC\wedge FC,TIP)\circ(c_1^3,BC,INP)\circ(c_2^3,BC,TIP)\circ(c_2^2,BC\wedge FC,INP)\circ\\&(c_4^1,BC,INP)\circ(c_6^1,BC,DIP)\circ(c_8^2,FC,INP)\circ(c_8^4,BC\wedge FC,INP)\circ\\&(c_9^2,BC\wedge FC,DIP)\circ(c_{11}^2,BC,TIP).\end{aligned}$$

按照 $IE(\pi_{C_4})\_Simplified$ 中的安装表达式依次进行构件的安装,得到 $SC^{R_2}=(S^{R_2-sys},C^{R_2-sys})=(\{s_1,s_2,s_3,s_4\},\{c_0^1,c_0^2,c_1^1,c_1^2,c_3^3,c_2^4,c_2^1,c_4^3,c_5^5,c_6^3,c_8^3,c_9^1,c_{11}^2\})$. $IE(\pi_{C_4})\_Simplified$ 中的所有构件(除了 c_2^3)都插入到了 C^{R_2-sys} 中.而没有插入的构件 c_2^3 因为具有 $BC\wedge FC$ 兼容性的构件 c_2^4 的存在,使得 c_2^3 的功能可以由 c_2^4 来提供.根据规则 2 可知, SC^{R_2} 是一个成功的服务核恢复步.相似地,我们也可以判断 $SC^{R_3}=(S^{R_3-sys},C^{R_3-sys})=(\{s_1,s_2,s_3,s_4,s_8\},\{c_0^1,c_1^3,c_3^3,c_2^3,c_4^3,c_5^5,c_6^3,c_6^1,c_8^4,c_9^1,c_9^2,c_{11}^2\})$也是一个成功的服务核恢复步.

Table 5 Components and their compatibilities of SC^{R_1}

表 5 SC^{R_1} 中的构件及其兼容性

Component	Compatability
c_0^1	$BC \wedge FC$
c_1^1	$BC \wedge FC$
c_1^2	BC
c_3^3	BC
c_2^4	$BC \wedge FC$
c_4^2	$BC \wedge FC$
c_5^2	FC
c_5^4	BC
c_6^3	FC
c_{11}^1	BC

Table 6 Installation expressions of $supp(s_4,\{c_0^1,c_1^1,c_3^2,c_2^3,c_2^1,c_4^3,c_5^1,c_5^3,c_5^5,c_8^3,c_9^1,c_{11}^2\})$

表 6 $supp(s_4,\{c_0^1,c_1^1,c_3^2,c_2^3,c_2^1,c_4^3,c_5^1,c_5^3,c_5^5,c_8^3,c_9^1,c_{11}^2\})$ 的安装表达式

Component (c_i^p)	Compatability ($comp_i^p$)	Installation policy (IP_i^p)
c_0^2	FC	DIP
c_1^1	$BC \wedge FC$	NIP
c_3^2	FC	NIP
c_2^3	BC	INP
c_2^1	FC	DIP
c_4^3	$BC \wedge FC$	INP
c_5^1	BC	DIP
c_5^3	$BC \wedge FC$	INP
c_5^5	$BC \wedge FC$	TIP
c_8^3	BC	DIP
c_9^1	BC	NIP
c_{11}^2	BC	INP

Table 7 Components and their compatibilities of SC^{R_2}

表 7 SC^{R_2} 中的构件及其兼容性

Component	Compatability
c_0^1	$BC \wedge FC$
c_0^2	FC
c_1^1	$BC \wedge FC$
c_1^2	BC
c_3^3	BC
c_2^4	$BC \wedge FC$
c_2^1	FC
c_4^3	$BC \wedge FC$
c_5^5	$BC \wedge FC$
c_6^3	FC
c_8^3	BC
c_9^1	BC
c_{11}^1	BC

Table 8 Installation expressions of $supp(s_8,\{c_0^2,c_1^3,c_3^1,c_2^4,c_2^3,c_2^2,c_4^1,c_5^4,c_6^1,c_8^2,c_8^4,c_9^2,c_{11}^2\})$

表 8 $supp(s_8,\{c_0^2,c_1^3,c_3^1,c_2^4,c_2^3,c_2^2,c_4^1,c_5^4,c_6^1,c_8^2,c_8^4,c_9^2,c_{11}^2\})$ 的安装表达式

Component (c_i^p)	Compatability ($comp_i^p$)	Installation policy (IP_i^p)
c_0^2	$BC \wedge FC$	TIP
c_1^3	BC	INP
c_3^1	FC	NIP
c_2^4	$BC \wedge FC$	NIP
c_2^3	BC	TIP
c_2^2	$BC \wedge FC$	INP
c_4^1	BC	INP
c_5^4	BC	NIP
c_6^1	$BC \wedge FC$	DIP
c_8^2	FC	INP
c_8^4	$BC \wedge FC$	INP
c_9^2	$BC \wedge FC$	DIP
c_{11}^2	BC	TIP

参考可生存性规范 $SPEC$,由于 $S^{R_3-sys}=\{s_1,s_2,s_3,s_4,s_8\}=S^{8-sys}$ 且 $R=SC^{R_1}\circ SC^{R_2}\circ SC^{R_3}$ 中的每一个服务核恢复步都是一个成功的服务核恢复步,根据规则 6 可以得出, $SC^3 \xrightarrow{R} SC^8$ 是一个成功的服务核恢复.

5 相关工作与比较

尽管系统的可生存性问题一直是该领域研究人员关注的热点,但是至今都没有一个精确和全面的可生存性的定义.Westmark[16]在 2004 年对 ACM,IEEE 和 SEI(Software Engineering Institute)的数据库进行了查询和统计,发现存在着各种各样的可生存性定义,并指出这种现象是导致可生存性研究与实践严重脱节的重要原因.Knight 等人在文献[18,19]中给出了一个对信息系统可生存性的严格定义(rigorous definition),通过规范的形式来定义系统的可生存性.该规范包括可接受的服务(acceptable service)、服务值(service value)、服务转移(service transition)、操作环境(operating environment)、服务概率(service probabilities)5 个方面.本文的工作就是在其相关工作的基础上研究了基于可生存性规范的构件系统恢复的形式化模型,并提出了相应的系统恢复性质的推理框架.

与已有的工作相比,我们注意到,构件系统是通过构件间的交互及其依赖关系来提供不同等级的、用户能够感知的服务的,这些依赖关系对构件系统可生存性有着重要的影响.当系统恢复需要启动一个服务时,构件间的依赖关系决定了支持该服务的构件的安装顺序.如果忽略了构件间的这种依赖关系,系统就不能提供用户可用的服务(available service)[25].例如,在移动视频点播系统 MVoD 的例子中,同步构件 $SC(C_{F(8)})$依赖于音频编解码构件 $AC(C_{F(9)})$和视频编解码构件 $VC(C_{F(10)})$之前启动,所以,同步构件必须在音频编解码构件和视频编解码构件之前启动.而音频编解码构件和音频编解码构件之间因为没有依赖关系,所以可以以任意顺序启动.这样,$c_8^4 \circ c_9^1 \circ c_{10}^2$ 或 $c_8^4 \circ c_{10}^2 \circ c_9^2$ 是遵循了依赖关系的构件安装序列,而 $c_9^1 \circ c_{10}^2 \circ c_8^4$ 或 $c_{10}^1 \circ c_9^2 \circ c_8^4$ 则不是.

服务核间转移的性质是衡量构件系统恢复的重要指标.运行时的动态构件安装是驱动构件系统从一个服务核转移到另一个服务核的重要因素.我们充分考虑到不同构件版本间的兼容性,通过定义构件的不同安装策略来描述系统状态变迁的成功性和安全性.在文献[26]中,Parrish 提出了一个关于构件系统部署的成功性(部署的应用能够工作)和安全性(没有已经存在的应用受损)的概念化的框架.本文的工作受其启发,把成功性和安全性的概念引入到构件系统恢复的过程中.而这方面的工作在可生存性研究领域中鲜有涉及.同时,我们建立了基于可生存性规范的、系统恢复性质的形式化推理框架.它能够确定什么样的系统恢复是成功的,什么样的系统恢复是安全的,这对于构建高可生存性的系统具有重要的实际意义.特别地,通过引入构件安装执行投影的概念来简化安装执行,会在一定程度上降低该模型在大规模的构件系统应用的复杂性,为该模型的实际应用提供重要的支持.

在大多数与可生存性相关的概念中,平滑退化(graceful degradation)和可生存性的内涵最为接近.平滑退化关注的是如何在现有系统配置的情况下为用户提供最大限度的服务功能,而可生存性则是在一定的系统状态下为用户提供预先定义好的、相应等级的服务.事实上,平滑退化提供了更细粒度上的对系统功能的控制.它强调的是系统如何平滑地从一个系统状态转移到另一个系统状态.在文献[27]中,Saridakis 提供了一种通过移除构件系统中的失效部分的方法来提供最大限度的系统功能.当系统出现失效部分的时候,这种方法不但涉及移除失效的构件,而且涉及移除失效的接口,甚至失效的绑定等等.与我们的研究相比,该方法忽略了用户的感知与应用的逻辑.

6 结论与展望

本文对构件系统的系统恢复问题进行了深入的分析,提出了一个基于可生存性规范的构件系统恢复的形式化模型.该模型通过由服务核所构成的可生存性规范,记录系统在不同的客观条件下所处的系统状态,利用构件间的依赖关系描述构件安装的顺序,同时,通过构件间的兼容关系和构件安装策略来建模服务启动的方式.特别地,提出了构件安装式基于该形式化模型,给出了推理构件系统恢复的成功性和安全性的规则和流程,并通过一个移动视频点播系统 MVoD 来展示该模型及其分析方法的可用性和有效性.

从软件工程的角度来说,“构件重用”是基于构件的软件开发中的核心过程之一.本文提出的方法对基于构件的软件开发具有一定的借鉴作用.在构件系统恢复的过程中,恢复一个服务的时候就面临着是安装所有支持

这个服务的构件,还是只重新安装部分构件而“重用”一部分已经安装的构件.其意义就在于提高了构件系统的可重用性,降低了系统的维护规模,减少了整个系统的负载,从而提高了系统的性能.当然,这还需要从构件库的组织结构中增加构件的家族、兼容性等分类信息,利用安装策略的表达,记录系统恢复步骤的过程来实现这种“构件重用”.

从实际开发代价的角度来说,基于本文方法的构件恢复系统的开发与实现正在进行当中.其具体的方案主要是在现有的构件库中增加构件的兼容性信息,同时修改构件注册中心的某些数据结构,如增加版本信息等.对于系统状态的记录、维护与判断,则通过相应的“状态维护模块”和“状态监控器”来共同实现.从现在开发的过程来看,此方法的代价是实际可以接受的.它不需要对原有构件系统做全新的改变,只需要开发相应的模块.

由于 Internet 开放、动态的环境下的构件系统可生存性研究是一个相对较新的研究领域,同时,对于可生存性本身的认识还在不断发生变化,所以,我们提出的模型仍需要进行不断的改进.今后主要的研究工作集中在以下两个方面:

(1) 在可生存性规范中引入概率模型,为量化评价系统恢复的程度提供支持.“恢复程度”的量化评价是一个非常复杂的问题,它涉及模型的量化指标的建立和评估机制的建立等方面.我们初步的想法是转移概率等指标引入现有的模型,为恢复程度的评价提供基础.

(2) 建立系统恢复性质推理规则的验证方法与工具.主要考虑如何定义这些性质推理规则的完备性等概念并开发相应的验证工具,实现这些规则验证的自动化,为本文方法在实际中的大规模应用提供支持.

References:

[1] Mei H, Chen F, Feng YD, Yang J. ABC: An architecture based, component oriented approach to software development. Journal of Software, 2003,14(4):721−732 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/721.htm

[2] Yang FQ. Thinking on the development of software engineering technology. Journal of Software, 2005,16(1):1−7 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/1.htm

[3] Mei H, Huang G, Zhao HY, Jiao WP. An internetware development approach centered on software architecture. Science in China (Series E), 2006,36(10):1100−1126 (in Chinese with English abstract).

[4] Li W, He J, Ma QK, Yen IL, Bastani F, Paul R. A framework to support survivable Web services. In: Proc. of the 19th IEEE Int'l Conf. on Parallel and Distributed Processing Symp. (IPDPS 2005). Denver: IEEE Computer Society, 2005.

[5] Cankaya HC, Nair VSS. A survivability assessment tool for restorable networks. In: Proc. of the 3rd IEEE Symp. on Application-Specific Systems and Software Engineering Technology. IEEE Press, 2000. 319−324.

[6] Jha S, Wing J, Linger R, Longstaff T. Survivability analysis of network specifications. In: Proc. of the Int'l Conf. on Dependable Systems and Networks. IEEE Press, 2000. 53−58.

[7] Louca S, Pitsillides A, Samaras G. On network survivability algorithms based on trellis graph transformations. In: Proc. of the Int'l Symp. on Computers and Communications. IEEE Press, 1999. 1008−1023.

[8] Snow AP, Varshney U, Malloy AD. Reliability and survivability of wireless and mobile networks. IEEE Computer, 2000,33(7): 449−454.

[9] Fung C, Chen YL, Wang XY, Lee J, Tarquini R, Anderson M, Linger RC. Survivability analysis of distributed systems using attack tree methodology. In: Proc. of the IEEE Military Communications Conf. (MILCOM 2005). IEEE Computer Society, 2005. 583−589.

[10] Hiltunen MA, Schliching RD, Ugarte CA. Building survivable services using redundancy and adaptation. IEEE Trans. on Computers, 2003,52(2):181−194.

[11] Bowen T, Chee D, Segal M, Sekar R, Shanbhag T, Uppuluri P. Building survivable systems: An integrated approach based on intrusion detection and damage containment. In: Proc. of the DARPA Information Survivability Conf. and Exposition (DISCEX 2000). Vol. 2. IEEE Computer Society, 2000. 25−27.

[12] Linger RC. Systematic generation of stochastic diversity as an intrusion barrier in survivable systems software. In: Proc. of the 32nd Annual Hawaii Int'l Conf. on System Sciences (HICSS-32). Wailea: IEEE Computer Society, 1999.

[13] Mead NR, Linger RC, McHugh J, Lipson HF. Managing software development for survivable systems. Annals of Software Engineering, 2001,11(1):45−78.

[14] Ma QK, Xiao LL, Yen IL, Tu MH, Bastani FB. An adaptive multiparty protocol for secure data protection. In: Proc. of the 11th Int'l Conf. on Parallel and Distributed Systems. Vol.1. IEEE Computer Society, 2005. 43−49.

[15] Bowers S, Delcambre L, Maier D, Cowan C, Wagle P, McNamee D, Le Meur AF, Hinton H. Applying adaptation spaces to support quality of service and survivability. In: Proc. of the DARPA Information Survivability Conf. and Exposition (DISCEX 2000). Vol.2. IEEE Computer Society, 2000. 271−283.

[16] Westmark VR. A definition for information system survivability. In: Proc. of the 37th Annual Hawaii Int'l Conf. on System Sciences (HICSS 2004). 2004. 303−312.

[17] Networked systems survivability program. 2007. http://www.sei.cmu.edu/programs/nss/nss.html

[18] Knight C, Strunk EA. Achieving critical system survivability through software architectures. In: de Lemos R, *et al*., eds. Architecting Dependable Systems II. LNCS 3069, Berlin, Heidelberg: Springer-Verlag, 2004. 51−78.

[19] Knight JC, Strunk EA, Sullivan KJ. Towards a rigorous definition of information system survivability. In: Proc. of the DARPA Information Survivability Conf. and Exposition (DISCEX 2003). 2003. 78−79.

[20] Szyperski C, Gruntz D, Murer S. Component Software: Beyond Object-Oriented Programming. 2nd ed., Massachusetts: Addison-Wesley Professional, 2002.

[21] Avizienis A, Laprie JC, Randell B, Landwehr C. Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. on Dependable and Secure Computing, 2004,1(1):11−33.

[22] Coupaye T, Estublier J. Foundations of enterprise software deployment. In: Proc. of the Euromicro Conf. on Software Maintenance and Reengineering. Zurich: IEEE Computer Society, 2000.

[23] Carroll JM. Scenario-Based Design: Envisioning Work and Technology in System Development. New York: John Wiley & Sons, 1995.

[24] Damm W, Harel D. LECs: Breathing life into message sequence charts. Formal Methods in System Design, 2001,19(1):45−80.

[25] Alda S, Won M, Cremers A. Managing dependencies in component-based distributed applications. In: Proc. of the Int'l Workshop on Scientific Engineering of Distributed Java Applications (FIDJI). LNCS 2604, Luxembourg-City: Springer-Verlag, 2002. 143−154.

[26] Parrish AS, Dixon B, Cordes D. A conceptual foundation for component-based software deployment. Journal of Systems and Software, 2001,57(3):193−200.

[27] Saridakis T. Surviving errors in component-based software. In: Proc. of the 31st EUROMICRO Conf. on Software Engineering and Advanced Applications (EUOMICRO-SEAA 2005). 2005. 114−123.

附中文参考文献:

[1] 梅宏,陈锋,冯耀东,杨杰.ABC:基于体系结构、面向构件的软件开发方法.软件学报,2003,14(4):721−732. http://www.jos.org.cn/1000-9825/14/721.htm

[2] 杨芙清.软件工程技术发展思索.软件学报,2005,16(1):1−7. http://www.jos.org.cn/1000-9825/16/1.htm

[3] 梅宏,黄罡,赵海燕,焦文品.一种以软件体系结构为中心的网构软件开发方法.中国科学(E 辑),2006,36(10):1100−1126.

卢暾(1977－),男,湖南湘潭人,博士,主要研究领域为服务计算,协同计算.