集成架构风格的SA软件部署可靠性

苏喜红，刘宏伟，吴智博，杨孝宗，左德承

（哈尔滨工业大学 计算机科学与技术学院，150001 哈尔滨）

摘 要：已有的架构风格可靠性研究主要针对单一软件架构风格，为研究软件部署阶段的多种架构风格的系统可靠性，设计用连接器的方式来建模部署阶段的集成架构风格系统和软件组件可靠性. 实验结果表明，随着系统运行时间的增加，集成架构风格的系统可靠性和组件可靠性主要依赖于处于执行状态的软件组件数目和软件组件处于执行状态的时间. 通过对集成架构风格系统可靠性和软件组件可靠性的分析，有利于选择合适的时间通过系统重部署或者软件组件复制来提高系统可靠性.

关键词：架构风格；可靠性；软件部署；软件架构

中图分类号：TP311.5 **文献标志码**：A **文章编号**：0367－6234(2012)05－0071－04

Reliability of SA based software deployment considering integrated architectural style

SU Xi-hong, LIU Hong-wei, WU Zhi-bo, YANG Xiao-zong, ZUO De-cheng

(School of Computer Science and Technology, Harbin Institute of Technology, 150001 Harbin, China)

Abstract: Existed reliability researches of architectural styles are mainly aimed at individual architectural style. To study system reliability of multiple architectural styles during software deployment stage, the system of integrated architectural style and reliabilities of software components are modeled by the connector method. Experiments show that with the increasing of system run time, the system reliability of the integrated architectural style and component reliabilities mainly depend on the number and time of software components which are in the executed state. On the basis of system reliability analysis of the integrated architectural style and reliabilities of software components, the reasonable time is decided by redeployment or replicating components to improve the system.

Key words: architectural style; reliability; software deployment; software architecture

随着分布式软件与网络的快速发展，软件部署成为整个软件生命过程中的一个独立阶段[1]. 软件工程研究者和实践者[2]通过利用软件架构(Software Architecture，SA)的原则来开发软件系统，已经成功地处理软件系统复杂性的增长.

架构风格决定了在风格实例中可以使用的连接器与组件的词汇表，限制架构元素的功能和角色、以及可以存在的架构元素之间关系的一组架构约束[3－5]. Wang[6]根据组件的可靠性、操作剖面和软件架构提出了一个架构层次的软件可靠性评估模型，该模型分析了调用和返回风格、容错风格、并行风格的软件可靠性，但是该方法只分析单个架构风格的可靠性. 然而，在大规模的复杂系统中，往往存在多种架构风格来满足不同性能需求. 例如，对于一个用J2EE开发的应用程序，可能既是“3层C/S风格”，又是“面向对象风格”，然而这种风格的混合有可能导致风格的适配问题[7]. 而风格的集成是分别把风格应用到软件的不同组

收稿日期：2011－05－05.

基金项目：国家高技术研究发展计划基金重大资助项目(2008AA01A201)；科技部国际科技合作计划资助项目(2010DFA14400).

作者简介：苏喜红(1981—)，女，博士生；
刘宏伟(1971—)，男，教授，博士生导师；
吴智博(1954—)，男，教授，博士生导师；
杨孝宗(1939—)，男，教授，博士生导师；
左德承(1972—)，男，教授，博士生导师.

通信作者：苏喜红，suxihong07@gmail.com.

成部分上,不会导致风格适配问题.

另外,现有的部署阶段可靠性研究主要为设计组件重部署算法[8]、组件复制算法[9]等,缺乏对部署阶段的集成架构风格的可靠性分析. 因此,本文通过连接器的方式,对软件部署阶段的集成架构风格进行建模. 分析部署阶段的组件失效原因,对不同架构风格的软件组件可靠性进行建模,并分析集成架构风格的组件可靠性和系统可靠性.

1 集成架构风格建模

通过连接器方法将多个架构风格集成在一个架构结构中. 软件组件间通过不同架构风格的连接器来进行交互. 设系统包括 n 个软件组件 $C = \{C_1, C_2, \cdots, C_n\}$,$m$ 个主机节点 $H = \{H_1, H_2, \cdots, H_m\}$,$k$ 个连接器 $L = \{L_1, L_2, \cdots, L_k\}$ 和 s 种架构风格 $A = \{A_1, A_2, \cdots, A_s\}$,如图 1 所示. n 个软件组件被分为 m 个软件组件子集合,$CS_1, CS_2, \cdots, CS_m$,其中 CS_i 为第 i 个软件组件子集合. $CS_1 = \{C_{11}, C_{12}, \cdots, C_{1i}\}, \cdots, CS_m = \{C_{m1}, C_{m2}, \cdots, C_{ml}\}$,并且 $C = \bigcup_i CS_i$. k 个连接器被分为 m 个连接器的子集合,$LS_1, LS_2, \cdots, LS_m$,其中 LS_i 为第 i 个连接器子集合. $LS_1 = \{L_x, \cdots, L_y\}, \cdots, LS_m = \{L_i, \cdots, L_j\}$,并且 $L = \bigcup_i LS_i$,连接器的可靠性研究超出了本文范围,假定连接器是完全可靠的.

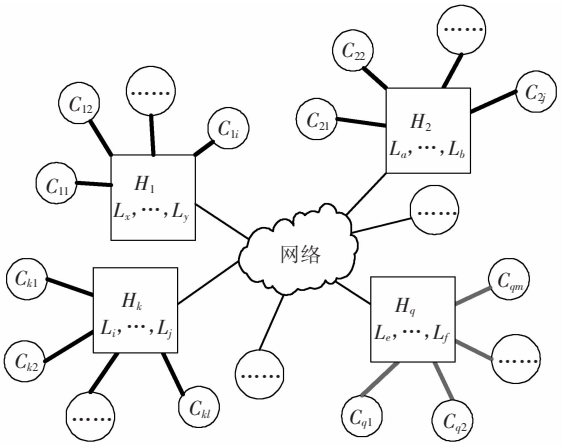

图 1 集成架构风格的部署架构图

2 软件组件可靠性建模

软件组件部署到主机节点上,当软件组件执行时,软件组件的失效原因主要有两种情况:1)软件组件部署到的主机节点失效和软件组件内部引起的组件失效;2)当软件组件未执行时,如果软件组件部署到的主机节点失效,那么软件组件则不能提供服务或者不能请求服务,此时软件组件被认为是失效. 这时,软件组件的失效率近似为主机节点的失效率. 例如,假定软件组件软件在不同时间单元下的可靠性阶段性变化如图 2 所示. 其中,在 3、5、6 和 9 min,软件组件处于未执行状态. 随着系统运行时间的增加,主机节点的某些部件有损耗,其可靠性略微降低. 在 1、2、4、7、8 和 10 min,软件组件处于执行状态.

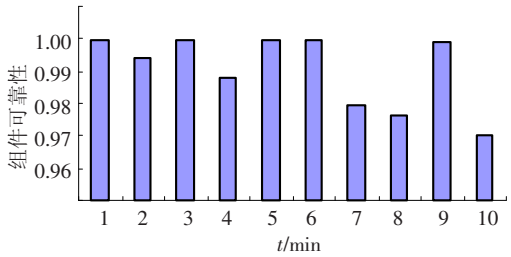

图 2 不同时间单元单个软件组件可靠性值

集成架构风格系统既有同构的软件组件,也有异构的软件组件. 同构软件组件是传统的完全由同一组织并且使用同一种技术来开发的软件组件. 异构的软件组件指包括 COTS(commercial off-the-shelf)组件、Services 组件等第三方开发的多种异构的软件组件,这些软件组件可能由不同的提供者并且使用不同的技术来开发完成的[10]. 因此,所有软件组件的失效行为服从同一分布的假设不合理. 对于存在异构软件组件的系统,其软件组件失效行为用不同的函数分布来建模为

$$RC_i = \begin{cases} f(t), & \text{组件处于执行状态;} \\ \psi_i, & \text{组件处于未执行状态.} \end{cases}$$

式中:$f(t)$ 为软件组件的可靠性为随时间变化的某一函数分布,如指数分布、威布尔分布等;ψ_i 为软件组件 C_i 处于未执行状态时的可靠性值,近似为部署到的主机节点的可靠性值.

3 实验及结果分析

3.1 实验输入

实验系统包括 CS、PF 和 C2 这 3 种架构风格、5 个主机节点 $H = \{H_1, H_2, H_3, H_4, H_5\}$ 和 17 个软件组件 $C = \{C_1, C_2, \cdots, C_{17}\}$. 软件组件间通过不同架构风格的连接器进行交互,这里认为连接器是完全可靠的.

随机产生 17 个软件组件在 5 个主机节点上的部署架构,软件组件 C_4 和 C_5 部署到 H_1 上,C_1、C_2 和 C_3 部署到 H_2 上,C_6、C_7 和 C_{14} 部署到 H_3 上,C_{12}、C_{13}、C_{15} 和 C_{16} 部署到 H_4 上,C_8、C_9、C_{10}、C_{11} 和 C_{17} 部署到 H_5 上. 这些软件组件类型如表 1 所示.

17 个软件组件的处于执行状态和未执行状态的时间单元如表 2 所示. 软件组件处于执行状态,用“✓”表示,软件组件处于未执行状态,用

“ × ”表示.

表 1　软件组件类型

软件组件类型	软件组件
服务器	C_2, C_4
客户端	C_1, C_3, C_5
管道	C_7, C_{13}, C_{15}
滤波器	C_6, C_{12}, C_{14}, C_{16}
C2	C_8, C_9, C_{10}, C_{11}, C_{17}

表 2　不同时间单元的软件组件状态

状态	1	2	3	4	5	6	7	8	9	10
C_1	✓	×	✓	×	×	✓	×	×	×	×
C_2	✓	✓	×	✓	×	✓	✓	✓	×	✓
C_3	×	×	✓	×	×	×	✓	×	×	✓
C_4	✓	✓	×	✓	×	✓	✓	✓	×	✓
C_5	✓	×	✓	×	✓	×	×	×	✓	×
C_6	✓	×	✓	×	×	×	✓	×	×	✓
C_7	×	✓	×	×	✓	×	×	×	✓	×
C_8	✓	×	✓	×	×	✓	×	✓	×	×
C_9	×	✓	×	×	✓	×	✓	×	×	✓
C_{10}	×	✓	×	✓	×	✓	×	✓	×	✓
C_{11}	✓	×	✓	×	✓	×	✓	×	✓	×
C_{12}	✓	×	✓	✓	×	✓	×	✓	×	×
C_{13}	×	✓	×	×	✓	×	✓	×	×	✓
C_{14}	×	×	✓	×	×	×	✓	×	×	×
C_{15}	×	✓	×	✓	×	×	×	×	✓	×
C_{16}	×	×	✓	×	×	✓	×	✓	×	✓
C_{17}	✓	×	✓	×	✓	×	✓	×	✓	×

PF 风格的软件组件可靠性用指数分布来建模. 其中: 如果软件组件 C_i 为 pipe 组件, 则 $\lambda_i = 0.000\,4$; 如果组件 C_i 为 filter 组件, 则 $\lambda_i = 0.000\,2$.

$$RC_i = \begin{cases} e^{-\lambda_i \times t}, & \text{组件处于执行状态;} \\ \psi_i, & \text{组件处于未执行状态.} \end{cases}$$

CS 和 C2 架构风格的软件组件可靠性用威布尔分布来建模为

$$RC_i = \begin{cases} e^{-(\frac{t}{\eta})^{\beta_i}}, & \text{组件处于执行状态;} \\ \psi_i, & \text{组件处于未执行状态.} \end{cases} \quad (1)$$

β_i 可由式(2)求出:

$$\beta_i = \beta_0 - \tau_i. \quad (2)$$

在式(1),(2)中,根据文献[11],对于包含 20 个软件组件的系统,如果用威布尔分布建模软件组件的可靠性,则 $\eta = 2\,917$, $\beta_0 = 1.29$. τ_i 为在同一架构风格的不同类型的软件组件对系统可靠性的影响,如果组件 C_i 是 C2 风格的软件组件,则 $\tau_i = 0.003\,1$; 如果 C_i 是 server 组件, 则 $\tau_i = 0.007\,225$; 如果 C_i 是 client 组件, 则 $\tau_i = 0.004\,725$.

3.2　实验结果分析

随着系统运行时间增加,不同架构风格的软件组件可靠性变化不确定,这里以 C_{13} 和 C_2 为例. C_{13} 为 pipe 组件,其可靠性变化情况如图 3 所示. C_2 为 server 组件,其可靠性变化情况如图 4 所示.

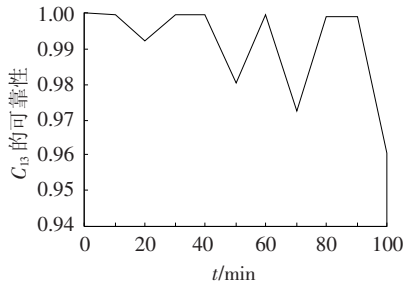

图 3　C_{13} 的可靠性变化

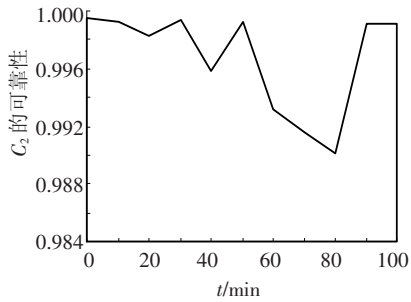

图 4　C_2 的可靠性变化

由图 3 和图 4 可知软件组件在哪些时间单元处于执行状态和未执行状态. 随着系统运行时间的增加,软件组件的可靠性总体趋势是在下降. 当系统处于未执行状态时,软件组件的可靠性会增加.

在特定的时间下,各个软件组件可靠性的变化情况如图 5 所示. 选取的时间为系统运行到 $t = 30$、60、90、100 min. 其中, $t = a$(a 为呈) 为系统运行时间为 a 时各软件组件的可靠性.

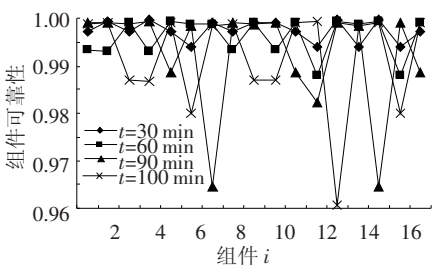

图 5　不同时间下各组件的可靠性

从图 5 可以看出,随着系统运行时间的增加,软件组件可靠性变化不确定. 当系统运行到 $t = 30$ min 时,各软件组件的可靠性差别并不很明显,系统中没有较弱的点. 当系统运行到 $t = 90$ min 时,软件组件 C_7 和 C_{15} 的可靠性变得很低,成为系统中最弱的两个点. 对于软件组件 C_7,在 $t = 60$、90 min 时,组件的可靠性比较高, C_7 应处于未执

行状态,此时组件的可靠性为主机节点 H_3 的可靠性值. 但是由于在其他时间段内处于执行状态时情况比较多,导致 C_7 下降明显. 对于软件组件 C_{16},系统执行到 t = 30 min 时,C_{16} 处于未执行状态,而在 t = 60、90、100 min 时,C_{16} 均处于执行状态,其可靠性逐渐下降.

系统可靠性变化情况如图 6 所示,随着系统运行时间的增加,系统可靠性总体趋势是下降的. 在 t = 40、60、80 min 时,系统可靠性有所提高,可能的原因是处于未执行状态的软件组件数目增多,可靠性高的软件组件数量增加,从而使系统可靠性增加. 在 t = [60,80] min 内,系统可靠性波动比较大,尤其在 t = 70 min 时,系统可靠性很低,可能的原因是处于执行状态的软件组件数目增多,或者某些软件组件已失效. 因此,要重点分析系统可靠性波动比较大的时间段内的可靠降低的原因. 在 t = 70 min 时,可以考虑通过软件组件复制或者系统重部署来提高可靠性,从而使系统更稳定地运行.

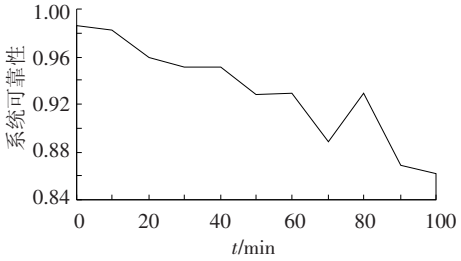

图 6 系统可靠性

4 结 论

1)通过连接器的方式,对部署阶段的集成架构风格进行建模;

2)分析集成架构风格的组件失效原因,对不同架构风格的软件组件可靠性进行建模;

3)在系统运行一段时间内,分析集成架构风格中的软件组件可靠性和系统可靠性.

参考文献:

[1] 梅宏, 申峻嵘. 软件体系结构研究进展[J]. 软件学报, 2006, 17(6): 1257 - 1275.

[2] MALEK S, KRISHNAN H R, SRINIVASAN J. Enhancing middleware support for architecture-based development through compositional weaving of styles [J]. Journal of System and Software, 2010, 83(12): 2513 - 2527.

[3] FIELDING R T. 架构风格与基于网络的软件架构设计 [D]. 李锟, 廖志刚, 刘丹, 等译. Irvine: dissertation of University of California, 2000: 10 - 74.

[4] GARLAN D, ALLEN R, OCKERBLOOM J. Architectural mismatch: why reuse is so hard [J]. IEEE Software, 1995, 12(6): 17 - 26.

[5] SHAW M, CLEMENTS P. A field guide to boxology: preliminary classification of architectural styles for software systems[C]//Proceedings of the 21st Annual International Computer Software and Applications Conference (COMPSAC). Washington: IEEE Computer Society, 1997: 6 - 13.

[6] WANG W, WU Y, CHEN M. An architecture-based software reliability model [C]//Proceedings of Pacific Rim International Symposium on Dependable Computing. Hong Kong: IEEE Computer Society, 1999:143 - 150.

[7] 杨芙清, 梅宏. 构件化软件设计与实现[M]. 北京: 清华大学出版社,2008: 3 - 297.

[8] MIKIC-RAKIC M, MALEK S, MEDVIDOVIC N. Improving availability in large, distributed, component-based systems via redeployment [J]. Lecture Notes in Computer Science, 2005, 3798: 83 - 98.

[9] POPESCU D. Framework for replica selection in fault-tolerant distributed systems[R]. Technical Report USC-CSSE-2007-702, 2007. http://csse.usc.edu/csse/TECHRPTS/2007/2007_main_exp.htm.

[10] 张弛. 异构组件互操作技术研究[M]. 合肥:中国科学技术大学出版社,2008:17 - 18.

[11] WANG P, JIN T. Complex systems reliability estimation considering uncertain component lifetime distributions [C]//Proceeding of the 8th International conference on Reliability, Maintainability and Safety (ICRMS). San Diego, CA: IEEE Computer Society, 2009: 20 - 24.

(编辑 张 红)