软件体系结构层次的适应性度量技术研究*)

高 晖 张 莉
(北京航空航天大学软件工程研究所 北京 100083)

摘 要 软件适应特定变化的能力对软件开发和维护都具有重要的影响。软件适应性(Software Flexibility)是反映软件适应变化能力的一项重要质量特征。实践说明软件体系结构设计在很大程度上决定了软件的适应性。在软件体系结构层次上量化地度量软件适应性,有助于设计人员改善和比较软件体系结构的设计方案,并可以辅助设计人员评价软件体系结构的质量。本文首先定义了软件适应性的因素-准则-度量模型(FCM模型);然后在此基础上提出了一种利用变化影响图(CIG)对软件适应性进行度量的方法。

关键词 软件适应性,软件体系结构,软件度量,变化影响图

Research on Flexibility Metrics in Software Architecture Level

GAO Hui ZHANG Li
(Software Engineering Institute, Beihang University, Beijing 100083)

Abstract The ability of adapting to changes is important in software development and maintenance. Software flexibility is a significant quality feature to reflect the ability of adapting to software changes. In practices, we find that software architecture impacts on the software flexibility. Measuring software flexibility in software architecture level is helping designer to improve and compare software architecture models, and assess the qualities of software architecture. In this paper, a Factor-Criteria-Metric Model is purposed for software flexibility in software architecture level. Then some metrics are developed to measure software flexibility according to the new kind of Change Impacted Graph (CIG).

Keywords Software flexibility, Software architecture, Software metrics, Software quality

1 介绍

在软件的开发、运行、维护过程中,很多因素都可能导致软件发生变化,这些因素可能是:软件功能变化、软件规模发生变化、软件运行环境变化、安全性要求提高等等。软件适应特定变化的能力对软件开发和维护都具有重要的影响。软件适应性(Software Flexibility)是反映软件适应变化能力的一项重要质量特征。不少研究人员和机构对软件适应性进行了定义。其中,IEEE给出了软件适应性的定义[1],软件适应性是指一个系统或构件为适应业务应用和环境的改变(不包括设计时能够确定的改变)进行变化的难易程度。Lassing等人认为软件适应性是适应系统在需求和环境变化而付出的努力[2]。软件设计技术的发展对软件适应性的理解和认识起到了重要作用[3],结构化设计、模块化设计、面向对象设计到软件体系结构设计对提高软件适应性都起到了积极的作用,所以在软件设计阶段对适应性进行度量和评价是非常重要和有效的。特别是在软件体系结构设计中,软件体系结构利用构件、连接件等高度抽象概念描述了软件的顶层设计,是软件开发过程中非常重要的设计制品之一。对几十种软件体系结构模式[5~7]的分析中可以发现,65%以上的模式对软件适应性产生影响。结合实践经验,我们发现软件体系结构设计在很大程度上决定了软件的适应性。

为了能够在软件设计的初期,在软件体系结构层次上量化软件适应性,本文首先定义了软件适应性的因素-准则-度量模型(FCM模型);然后在此基础上提出了一种利用变化影响图(CIG)测量软件适应性的方法。在相关的研究方面,Amnon H. Eden, Tom Mens提出了利用软件变更复杂性对软件适应性进行度量的方法[8],提出了通用的软件变更复杂度可以对软件体系结构层次、软件设计层次和代码层次的适应性进行度量。该研究借用计算复杂度的思想对变更难易程度进行刻画,但该研究并未给出在软件体系结构层次上对复杂性进行度量的具体方法,也没有对变更的传播进行说明。本文的研究重点在软件体系结构层次上,并具体定义了变更复杂性的计算方法。T. Nakamura, V. R. Basili提出软件体系结构变化度量方法[9],该方法基于Graph Kernel的结构距离度量软件体系结构在开发过程的变化,可以高效地计算软件体系结构的差异。该方法的核心是计算软件体系结构模型之间的差异。但本文则通过变化影响图(CIG)来表示软件体系结构发生的变化,CIG所包含的信息更加丰富,使得我们可以直接利用CIG的性质来对软件适应性度量。

本文第2节说明了一些研究软件适应性的基本观点;第3节定义了软件适应性的FCM模型;第4节研究了软件体系结构的变化影响图,形式化地定义了变化、变化传播图(CIG);第5节利用CIG定义了软件体系结构层次的软件适应性度量组的计算方法;第6节给出了软件适应性的度量数据和分析结果。

*)本文受新世纪优秀人才支持计划资助(Supported by Program for New Century Excellent Talents in University)。高 晖 博士生,主要研究领域为软件体系结构,UML相关技术;张 莉 教授,博士生导师,主要研究领域为企事业过程工程,软件体系结构。

2 基本观点

软件发生变化是由于其实现的问题集发生了变化，见图1。当原有问题集变化为新的问题集时，原有的软件模型可能需要通过发生一系列的变化才能实现新的问题集。在这里软件模型可以是可执行系统、代码，甚至设计模型。从图1可以看出，当问题集产生变化时，可能导致软件体系结构模型发生变化。根据软件适应性的定义和特点，在软件体系结构层次研究软件适应性度量，就是要通过软件体系结构模型的变化的特点来度量软件适应性。

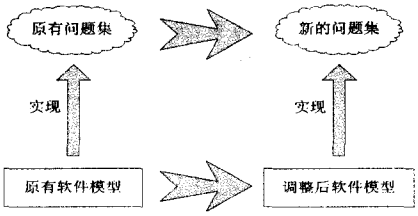

图1 软件变化的语境

下面就从一个实例来说明软件变化的特点及适应性度量定义中应该注意的几个问题。

实例

软件需求简述：考虑一个采用比例表示的用于政治选举的信息系统。这个系统提供了一个输入数据的电子数据表和表示当前结果的几种图表。用户可以通过图形接口与系统交互。所有信息现实必须立即反映出选举数据的变化。

软件体系结构设计：采用模型-视图-控制器(MVC)软件体系结构模式来设计该系统。在MVC模式中，模型构件封装了核心数据和功能。模型独立于特定输出表示法或输入方式。视图构件向用户显示信息。视图从模型获得数据。可能有模型的多个视图。每个视图都有一个相关的控制器构件。控制器接受输入，通常作为鼠标移动、鼠标按钮的活动或键盘输入编码的事件。事件被翻译成模型或视图的请求。用户仅仅通过控制器与系统交互。

需求变更和实现方案：

需求变更1. 提供一种新的数据表示方法，比如：增加政党间的议员席位分配视图。为了实现需求变更1，只需要增加一个视图(View)构件，并同时增加一个控制器(Controller)构件，即可实现政党间的议员席位分配的显示。

需求变更2. 将系统移植到其它图形界面库上，比如：从QT界面库移植到MFC界面库。为了实现需求变更2，我们需要修改所有的视图(View)构件，将这些构件使用到QT界面库API的部分全部修改为MFC界面库API。

分析：对以上两个需求变更的实现方案进行比较分析，可以发现，对于需求变更1，在软件体系结构层次，增加两个构件即可实现，不会对软件的其它部分造成影响；而对于需求变更2，由于所有视图构件都依赖于QT界面库的API，因此当替换API时，所有视图构件都需要修改。按照开发人员的经验，人们往往认为MVC模式对需求变更1的适应性更好，而对需求变更2的适应性较差。在一个利用MVC模式构造的软件系统中，将所有模型构件归为一类构件，同理，视图构件和控制器构件也可以分别归为一类。我们发现，软件变化涉及到一类构件时，人们往往从直观上认为这种情况下的软件适应性较差；而当软件变化只涉及到某类构件中的一个或多个，而不是所有构件时，人们往往从直观上认为这种情况下的软件适应性较好。

综上所述，可以形成本文定义软件适应性度量的几个基本观点。

观点1. 考虑设计开发人员对软件适应性的上述直观认识，本文认为在定义软件适应性的度量时应该根据变化的影响范围来区分不同类型的变化。

观点2. 软件元素之间存在依赖关系使得变化并不是孤立的。当某个元素发生变化时，可能会波及到软件的其它元素。所以，本文在定义软件适应性度量时，考虑的并不是一个一个孤立的软件变化，而是利用了软件变化及变化之间的关系来定义软件适应性度量。

观点3. 在软件体系结构层次的软件适应性度量，其度量对象是软件体系结构模型，可以通过软件体系结构的构件和连接件的变化及其变化之间的关系来对软件适应性进行度量。另外，由于构件和连接件本身是非常抽象的，其内部实现在软件体系结构层次上是未知的，如何考虑内部实现对软件适应性度量的影响是非常重要的。

观点4. 软件适应性不是绝对的，同一个软件对于不同的需求变更体现的适应性是不同的。

3 软件适应性的FCM模型

根据McCall定义FCM模型[10]的基本思想，软件适应性是一个高层的质量因素，类似于可靠性、可使用性，高层质量因素不易于直接进行测量。需要为软件适应性定义一些易于理解和易于测量的层次较低的质量准则。根据软件适应性与软件变化的关系，本文定义了软件适应性的FCM模型，见图2。

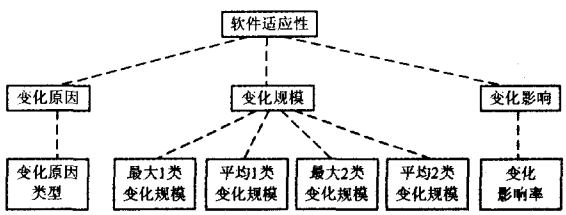

图2 软件适应性的FCM模型

·变化原因

变化原因是软件发生变化的原因，本文中规定了三种导致软件发生变化的原因：1. 功能变化；2. 质量调整；3. 技术变化。在定义软件适应性度量时，根据上一节中的观点4，不能绝对地看待某个软件的适应性，而是说明软件是否适应某种问题域变化。所以在软件适应性中，本文定义了变化原因类型对软件变化的原因进行度量。由于问题域变化是无穷无尽的，但问题域变化的类型是有限的。可以用软件变化原因类型来描述问题域变化的类型，本文定义的软件变化原因类型有三种：功能需求变更，质量调整和技术变化。功能需求变更指软件在功能上发生增加、减少或修改。质量调整指对软件的可靠性、可维护性、性能等质量特征方面的要求发生变化。技术变化指软件的外部环境发生变化，比如：操作系统升级或变化，中间件层的变化等。

·变化规模

变化规模是要衡量软件变化的范围和数量。根据观点1，软件变化的影响范围是有区别的，所以本文定义了两类变化来区分影响范围不同的两类变化。首先，本文认为软件体

系结构由许多构件构成，而这些构件又可以分为不同的类别。比如：基于 MVC 模式的软件系统中一定包含了模型类型的构件，控制器类型的构件和视图类型的构件。当变化的构件范围只是一个或某几个构件，而不是某一类型中所有构件都发生变化时，本文将这种变化称为 1 类变化。当变化的构件范围是某一种类型中的所有构件时，本文将这种变化称为 2 类变化。一般认为如果软件只发生 1 类变化，其适应性是较好的；如果软件发生较多的 2 类变化，其适应性则较差。

为了量化地在软件体系结构层次表示变化规模，根据观点 2、3，就必须能够描述软件体系结构的变化及其关系。本文定义了软件体系结构的变化影响图来描述软件体系结构变化及其关系。然后基于这种变化影响图定义了 4 个度量来量化地表示变化规模：最大 1 类变化规模、平均 1 类变化规模、最大 2 类变化规模和平均 2 类变化规模。这 4 个度量的定义和计算方法依赖于变化影响图，所以本文将在下一小节形式化地定义变化影响图，然后再根据变化影响图来定义这 4 个度量。

· 变化影响

变化影响是要说明变化的影响范围与软件系统整体的关系。本文定义了变化影响率来度量变化影响。与变化规模的 4 个度量一样，变化影响率的定义和计算方式也依赖于变化影响图。

4 软件体系结构的变化影响图

从软件适应性的 FCM 模型可以看出，软件适应性与“变化”密切相关。软件体系结构模型利用构件和连接件描述软件结构和行为方面的特征。所以，在软件体系结构层次上可见的变化包括：构件的增/删/改和连接件的增/删/改。然而，由于构件之间的依赖关系、连接件中的角色与构件之间的绑定关系，使得变化之间并不是独立的，而是存在一定影响关系。为了更加准确地描述软件体系结构变化以及变化之间的影响关系，本文定义了一种变化影响图（Change Impacted Graph，简称 CIG）对变化及其影响关系进行描述。变化影响图是计算变化规模和变化影响的基础。

本节中首先形式化地定义了软件体系结构的变化以及变化影响图的概念；然后研究了软件体系结构变化的传播关系；最后在这些研究的基础上给出了在给定一组变化和软件体系结构模型的基础上自动生成变化影响图的算法。

4.1 基本概念

4.1.1 变化

为计算软件适应性的各种度量提供有效的信息，变化的描述必须包括四个方面：变化作用的实体、变化的规模、变化的类型、该变化引入的其它变化。所以，我们采用一个四元组来描述变化。

定义 1 变化（Change）为一个四元组，ElementTemp 为变化的作用实体；ChangeSize 为变化规模；ChangeType 为变化的类型；ImpactedChanges 为该变化引发的其它变化集合，Weight 为变化加权系数（该系数可以省略），其形式化表示为：

$$
\begin{aligned}
&Change::=\langle ElementTemp, ChangeSize, ChangeType, ImpactedChanges\rangle;\\
&ElementTemp::=CompnentTemp\mid ConnectorTemp;\\
&ChangeSize::=\langle sixel, size2\rangle, size1 \text{ 为 1 类变化规模，} size2 \text{ 为 2 类变化规模；}\\
&ChangeType::=ADD\_COMP\mid MOD\_COMP\mid DEL\_COMP\mid ADD\_CONN\mid MKOD\_CONN\mid DEL\_CONN;\\
&ImpactedChanges::=\{ch\mid ch \text{ 为引发的其它变化}\} \qquad (1)\\
&Weight \text{ 为自然数，缺少为 1}
\end{aligned}
$$

变化作用的软件元素 ElementTemp 是软件体系结构层次上的基本元素。结合在实践中软件体系结构设计的经验，我们发现软件体系结构具有不同的抽象层次。通常情况下，可以发现软件体系结构中的基本建模元素，比如：构件和连接件等，是可以按照其特征进行划分的。例如：在分层的软件体系结构中，往往会根据构件所在的层次不同，而将构件划分为 UI 类型的构件、逻辑处理类型的构件、数据实体类型的构件；同理，根据不同的交互方式，也可以将连接件进行分类。这种分类实质上是对构件、连接件进行更高层次上的抽象。在本文中，定义“构件模板”的概念来表示具有相似功能的一组构件的抽象表示；定义“连接件模板”的概念来表示完成某组需求时构件模板之间通信、协调和合作机制的一种抽象表示。在软件体系结构模型发生变化时，当发生 1 类变化时，则只有一个或几个构件或连接件发生变化；当发生 2 类变化时，相同类型的构件或连接件都会发生变化。为了能够同时表示这两种变化，本文定义的 ElementTemp 就是“构件模板”或者“连接件模板”。

变化规模 ChangeSize 是一个二元组，1 类变化规模是指当发生 1 类变化时软件元素变化的个数；而 2 类变化规模是指当发生 2 类变化时同一类软件元素变化的次数。

变化类型 ChangeType 在软件体系结构模型上表现为 6 种，分别是：增加构件（ADD_COMP）、修改构件（MOD_COMP）、删除构件（DEL_COMP）、增加连接件（ADD_CONN）、修改连接件（MOD_CONN）、删除连接件（DEL_CONN）。

变化引入的其它变化 ImpactedChanges 则描述了该变化发生可能引发的其它变化，通过 ImpcatedChanges 就可以将软件系统的变化之间的关系进行描述。

变化加权系数 Weight 描述了软件元素的内部实现对软件变化规模的影响，该系数可以建立一个基准，使具有不同内部复杂性的软件元素在发生变化时可以比较。如果假设所有软件元素在发生变化时其成本或者难易程度都相同时，该系数可以省略。

下面给出了一组表示变化的语句，见式 2。

$$
\begin{aligned}
&addEntity=\langle Entity, 2, 0, ADD\_COMP, NULL\rangle\\
&modColl=\langle Collection, 0, 1, MOD\_COMP, \{modControl, modUI\}\rangle\\
&modControl=\langle Control, 0, 1, MOD\_COMP, \{modUI\}\rangle\\
&modUI=\langle UI, 5, 0, MOD\_COMP, NULL\rangle
\end{aligned} \qquad (2)
$$

addEntity 表示增加两个 Entity 类型的构件；modColl 表示修改属于 Collection 类型的所有构件，并且该变化会引起名为 modControl 和 modUI 的变化；modControl 表示修改属于 Control 类型的所有构件；modUI 表示修改属于 UI 类型的 5 个构件。

4.1.2 变化影响图

当软件系统发生变更时，其变化及其关系可以利用变化影响图进行描述。变化影响图是一个有向图，利用有向图的节点表示变化，利用有向边表示变化之间的影响关系。

定义 2 变化影响图（Change Impacted Graph，CIG）是一

个以变化为节点，变化的影响关系为边的有向图，其形式化表示为：

$$CIG=\langle V\_Changes, E\_Impacts\rangle \quad (3)$$

其中

$$V\_Changes=\{v_i \mid v_i \in Changes\}$$

$$E\_Impacts=\{\langle v_i, v_j\rangle \mid v_i, v_j \in V\_Changes \wedge Impact(v_i, v_j)\}$$，$Impact(v_i, v_j)$表示 v_i 导致 v_j

4.2 变化影响图的构造

继续以基于 MVC 模式的政治选举信息系统为例，当系统中的选举数据发生变化，即模型构件发生变化，那么软件的其它构件会发生什么样的变化？变化之间的关系又是如何？

本文可以利用变化影响图来定义变化及其关系。一般来说，设计人员将首先定义变化 modModel，见式(4)。该式说明所有 Model 类型的构件发生一次变化。

$$modModel=\langle Model, 0, 1, MOD\_COMP, NULL\rangle \quad (4)$$

然后，根据 MVC 模式中构件模板的依赖关系，Controller 和 View 都直接依赖于 Model，那么 Model 的变化，可能导致 Controller 和 View 的变化，式(4)就演变为式(5)。式(5)说明了变化 modModel 引起了变化 modController 和 modView。式(5)可以表示为一个变化影响图，见图 3。

$$modModel=\langle Model, 0, 1, MOD\_COMKP, \langle modController, modView\rangle\rangle$$

$$modController=\langle Controller, 0, 1, MOD\_COMP, NULL\rangle \quad (5)$$

$$modView=\langle View, 0, 1, MOD\_COMP, NULL\rangle$$

从上述的变化影响图的定义过程中，我们发现通过构件模板之间的依赖关系，再加上一组变化影响的假设，就可以构造变化影响图。

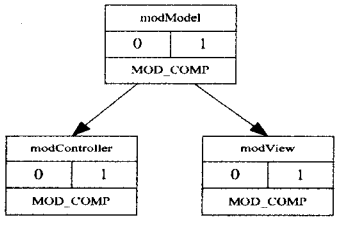

图 3 变化影响图示意图

5 度量组定义

在定义了变化影响图的基础上，就利用对变化影响图的各种特征来计算软件适应性 FCM 模型中的变化规模和变化影响。

5.1 度量 1：最大 1 类变化规模

定义 当软件体系结构发生变化时，其变化影响图为 CIG，则最大 1 类变化规模定义为：

$$MaxChangeSize1=\max(\{v_i, ChangeSize1 \mid v_i \in CIG, V\_Changes\}) \quad (6)$$

说明：最大 1 类变化规模的取值为变化影响图 CIG 中 1 类变化规模的最大值。

5.2 度量 2：平均 1 类变化规模

定义 当软件体系结构发生变化时，其变化影响图为 CIG，则平均 1 类变化规模定义为：

$$MeanchangeSize1=\frac{\sum_{v_i \in CIG.V\_Chagnes} v_i.ChangeSize1}{|GIG.V\_Chagnes|} \quad (7)$$

说明：平均 1 类变化规模的取值为变化影响图 CIG 中 1 类变化规模的平均值。

5.3 度量 3：最大 2 类变化规模

定义 当软件体系结构发生变化时，其变化影响图为 CIG，则最大 2 类变化规模定义为：

$$MaxChangeSize2=\max(\{v_i.ChangeSize2 \mid v_i \in CIG.V\_Changes\}) \quad (8)$$

说明：最大 2 类变化规模的取值为变化影响图 CIG 中 2 类变化规模的最大值。

5.4 度量 4：平均 2 类变化规模

定义 当软件体系结构发生变化时，其变化影响图为 CIG，则平均 2 类变化规模定义为：

$$MeanChangeSize2=\frac{\sum_{v_i \in CIG.V\_Chagnes} v_i.ChangeSize2}{|CIG.V\_Chagnes|} \quad (9)$$

说明：平均 2 类变化规模的取值为变化影响图 CIG 中 2 类变化规模的平均值。

5.5 度量 5：变化影响率

定义 当软件体系结构发生变化时，则变化影响率定义为：

$$ChangeRatio=\frac{\text{修改的构件数}+\text{增加的构件数}}{\text{系统原有构件数}+\text{增加的构件数}-\text{删除的构件数}} \quad (10)$$

说明：变化影响率是变化的构件数量与变化后系统构件数量的比值。

6 数据收集和分析

6.1 数据收集

在软件工程领域数据收集，特别是收集大量的软件体系结构设计模型是很困难的。然而，模式在软件工程领域的提出，为数据收集提供了新的思路。在软件工程领域中，专家从实践经验中获知了许多模式，并在开发具有特定属性的应用系统中遵循这些模式[5]。通常，软件工程中的模式(比如：设计模式、软件体系结构模式)具有以下一些特点使其能够作为经验设计模型被应用在研究中。(1)模式关注一个在特定设计环境中出现的重现设计问题，并为它提供一个解决方案。模式中包括了经过充分考验的设计经验。所以，模式实际上是对大量实际项目的经验总结，比某个独立的项目中的软件设计模型更具有代表性。(2)在软件开发过程中，特别是软件体系结构设计阶段，基于模式的设计方法已经被大量运用。由于设计问题将不断的重复出现，可以预测现有的各种模式将大量重用于其它的项目中。所以，利用模式来研究软件在体系结构层次的质量特征不仅能够对以前的项目所反映的特征进行说明，而且也适用于未来一段实践内的软件项目。(3)模式在一定程度上是可以度量的。与软件设计模型类似，软件体系结构模式和设计模式中以结构和动态特征来描述其组成软件元素以及软件元素之间的相互关系。所以，一般情况下模式是可以进行度量的，但也存在一些粗粒度的软件体系结构模式(比如：分层模式等)是无法直接度量的，对这类模式只需要加入一些应用假设，也是可以进行度量的。(4)模式的数量随着设计经验的积累而不断增加，现在模式的数量已经达到了一定的规模。在文[4～7]中给出了大量可以利用的软件体系结构模式和设计模式。所以，利用模式作为经验设计模型在数据量上也可以有一定的保证。综上所述，利用模式(或者加入一些应用假设的模式)作为经验设计模型，从

“质”和“量”两个方面都能得到保证。

本文从模式中收集了 21 种软件体系结构模式和 7 种设计模式作为研究的对象。对每种模式给出 3 个变更场景(变更需求),并针对每个变更场景给出相应的变化影响图。然后利用第 5 节的计算方法就可以得到软件适应性的经验数据。由于篇幅的原因,本文省略这些原始数据,下节将给出对这些数据分析的结果。

6.2 数据分析

6.2.1 变化原因类型与变化规模数据分析

在本文收集的数据中,变化原因类型是功能需求变更的数据有 28 个;质量调整的数据有 19 个;技术变化的数据有 8 个;变化原因类型同时为功能需求变更和质量调整的数据有 23 个;同时为功能需求变更和技术变化的数据有 2 个;同时为质量调整和技术变化的数据有 4 个。

下面分别对不同变化原因类型的数据,分析其变化规模。

(1)变化原因类型为功能需求变更的情况,见图 4(a)。在只发生功能需求变更的情况下,这组数据反映的现象与设计人员的常识是相符的。通常,软件对于与原有系统相似功能的扩展是发生局部性的变化,从数据中可以看出发生 1 类变化的概率是较大的。而发生 2 类变化的情况,一般是由于较底层的功能的变更,也就是通常所谓的一些“横切(cross-cut)”的功能发生变化,比如:在协调者模式中的变更场景(S19_C1),要求通过增加一个注册器来实现对任务参与者的控制。所以,软件对于功能需求的变更,特别是与原有系统具有相似功能的扩展时,一般具有较好的适应性。

(2)变化原因类型为质量调整的情况,见图 4(b)。在只发生质量调整的情况下,这组数据反映的现象与设计人员的常识是相符的。通常,对于软件质量的调整,比如:性能、可靠性,对软件或者软件某个部分来说是一种全局性的调整,从数据中可以看出发生 2 类变化的概率是较大的。所以,软件对于质量的调整,一般适应性是较差的。但如果在软件体系结构设计时,能够充分考虑质量的分解,即将软件质量分解到软件的不同部分(比如:性能的分解),这样也可以使得质量调整相对容易。比如:在 MVC 模式的变更场景(S5_C2)中,为了提高软件可维护性,可以利用 Command 模式来减少系统之间的耦合性。

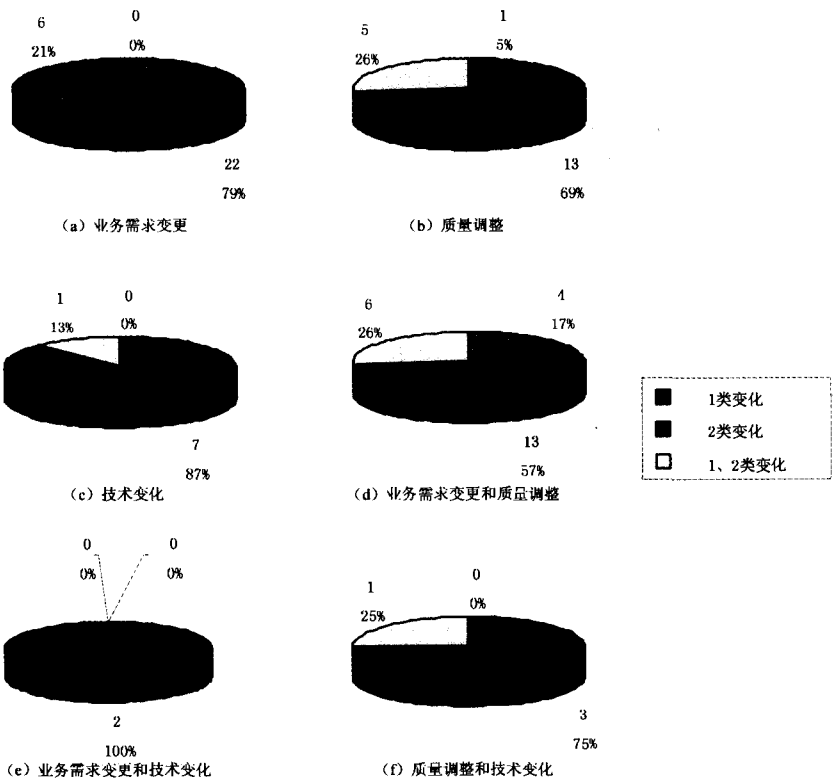

图 4 变化规模饼图

(3)变化原因类型为技术变化的情况,见图 4(c)。在只发生技术变化的情况下,这组数据反映的现象与设计人员的常识是相符的。通常,对于技术变化,也就是软件的外部环境发生变化时,对软件来说也是一种全局性的调整,从数据中可以看出发生 2 类变化的概率是很大的。所以,软件对于技术变化的调整,一般适应性是较差的。一般技术变化包括:操作系统变化、底层技术的变化等,只要在设计时将这些可能的变化隔离在软件的某些部分,也可以部分地解决技术变化造成的软件适应性较差的问题。

(4)变化原因类型为功能需求变更和质量调整的情况,见图 4(d)。在同时发生功能需求变更和质量调整的情况下,这组数据反映的现象与设计人员的常识是相符的。通常,在这种情况下的软件变化规模应该与仅发生质量调整的软件变化规模的特征相似,从数据中也可以看出这一点。所以,在这种情况下,软件适应性与仅发生质量调整的情况下是相似的。

(5)变化原因类型为功能需求变更和技术变化的情况,见图 4(e)。在同时发生功能需求变更和技术变化的情况下,按照常识,软件适应性在这种情况下应该与仅发生技术变化的软件适应性相似,应该是发生 2 类变化的概率较大。但从实际测量的数据看,结果却相反。这是由于只有 2 个变更场景(S4_C3 和 S7_C3)的变化原因类型是功能需求变化和技术变化。分析这两个变更场景,S4_C3 是在代理者模式中增加对

一种网络协议UDP的支持，由于在代理者模式中很好地将网络协议的支持封装在桥(Bridge)中，因此只需要增加一个支持UDP的桥即可实现；S7_C3则是要求微核模式支持另一个操作系统，由于在微核模式本身能够很好地适应操作系统的变化，因此只需要增加一个外部服务器和一个内部服务器即可。

(6)变化原因类型为质量调整和技术变化的情况，见图4(f)。在同时发生质量调整和技术变化的情况下，这组数据反映的现象与设计人员的常识是相符的。通常，对于质量调整和技术变化，对软件来说都是一种全局性的调整，从数据中可以看出发生2类变化的概率是很大的。所以，软件对于该类情况下，一般适应性是较差的。

6.2.2 变化规模与变化影响数据分析

下面分别对发生1类变化和2类变化的数据，分析其变化影响。本文使用了盒形图对数据进行分析。盒型图用中位数和四分位数来定义所有取值的中心位置和分布情况。这在测量数据的分布未知时，比常见的均值和方差更恰当一些。将软件适应性数据根据变化类型(1类变化、2类变化或者同时发生1、2类变化)分为3组数据，并对每组的变化影响率绘制其盒形图，见图5。从图5中可以看出，发生1类变化的数据其变化影响率较低，而发生2类变化和同时发生1、2类变化的数据其变化影响率较高。这也说明1类变化对软件的影响范围较小，而2类变化对软件的影响范围较大。所以，该组分析数据也可以从一个侧面说明，本文将变化分为1类变化和2类变化是有利于对软件适应性进行度量的，是符合人们对软件适应性理解的一般观点的。

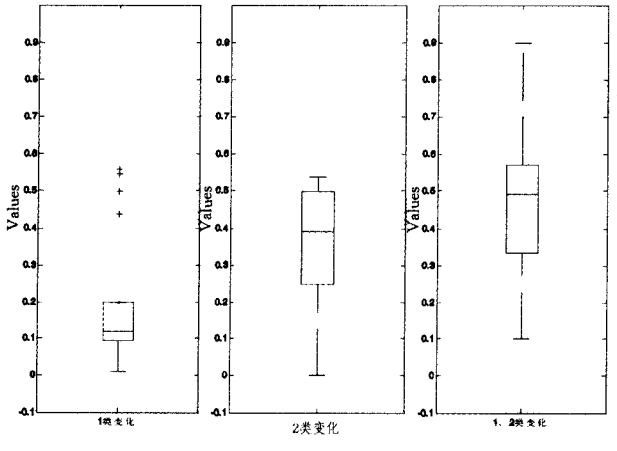

图5 变化影响盒型图

综上所述，通过对数据的初步分析，本文可以通过数据得到以下一些基本的结论。

(1)不同软件变化原因类型可能引发不同类型的软件变化，一般来说，功能需求变更引起1类变化的可能性较大；而质量调整和技术变化则引起2类变化的可能性较大。

(2)1类变化的影响范围较小，其变化影响率较低；而2类变化的影响范围较大，其变化影响率较大。

(3)软件适应性的质量子特征之间存在一些相关的关系，而这些关系属于不确定性的关系。

总结 本文的主要贡献在于首次提出了软件适应性的FCM模型；在此基础上提出了变化影响图的概念和形式化定义，并给定了利用变化影响图计算软件适应性度量的方法。本文的研究给出了量化软件适应性的基本方法，为进一步研究软件体系结构层次的软件适应性预测打下基础。我们已经在Rational Software Architecture平台上开发了软件体系结构的变化影响图的自动生成工具及适应性度量工具。本文实例研究中的数据都是由该工具自动计算得到。本文工作是作者博士论文工作的一部分，在进一步的研究中，我们将进一步收集更多的度量数据(包括结构度量和适应性度量数据)，再利用贝叶斯网学习理论在软件体系结构层次上建立软件适应性的预测模型。

参考文献

1 IEEE. Standard Glossary of Software Engineering Terminology 610.12-1990, Vol. 1. Los Alamitos: IEEE Press, 1999

2 Bengtsson P, Lassing N, Bosch J, et al. Architecture-level modifiability analysis (ALMA). The Journal of Systems and Software, 2004

3 Parnas D L. Software Aging. In: Proc. Inte'l Conf. of Software Engineering-ICSE94, Los Alamitos: IEEE Computer Society Press, 1994. 279～287

4 Gamma E, Helm R, Johnson R, et al. 设计模式－可复用面向对象软件的基础. 李英军，马晓星，蔡敏，等译，北京：机械工业出版社，2000. 16～17

5 Buschmann F, Meunier R, Rohnert H, Sommerlad P, Stal M. Pattern-Oriented Software Architecture Volume 1: A System of Patterns, China Machine Press, 2003

6 Buschmann F, Meunier R, Rohnert H, Sommerlad P, Stal M. Pattern-Oriented Software Architecture Volume 2. China Machine Press, 2003

7 Kircher M, Jain P. Pattern-Oriented Software Architecture Volume 3: Patterns for Resource Management. China Machine Press, 2005

8 Eden A H, Mens T. Measuring Software Flexibilty. IEE Software, 2005

9 Nakamura T, Basili V R. Metrics of Software Architecture Changes Based on Structural Distance. In: 11th IEEE International Software Metrics Symposium(METRICS 2005), 2005

10 Kan S H. 软件质量工程－度量与模型[M]. 第2版. 吴明晖，应晶，等译. 北京：电子工业出版社，2004