软件开发中的文档管理

周龙骧 柴兴无*

(中国科学院数学研究所,北京 100080)

DOCUMENTATION IN SOFTWARE DEVELOPMENT

Zhou Longxiang and Chai Xingwu

(*Institute of Mathematics, Academia Sinica, Beijing* 100080)

Abstract In this paper, we studied the principal points of documentation in software development, and discussed the functional design and implementation of a documentation tool which can be integrated in software engineering enviroments as well as be used independently by software developers.

摘要 在本文中,我们研究了软件开发中文档管理的主要内容,并提出了一个可由用户独立使用,也可用于软件工程支撑环境中的文档管理工具的功能设计及这样一个工具的实现.

§0. 引 言

在软件产品的开发过程中,文档管理贯穿于整个软件生存周期的每个阶段,它是软件开发中最基础的工作之一.其重要性主要体现在下述几个方面.

1. 文档是软件产品的组成部分

可以说,文档与软件是密不可分的.软件开发的最初需求是以文档形式提交的,而最终完成的产品不仅是软件,还有与其相关的所有文档.[3]中列举了通常被认为是软件产品的14个组成部分,这正是软件生产工程化的体现.

2. 软件开发过程是一个文档驱动的过程[2]

在软件生存周期的每个阶段中,都是以前一阶段的文档为起点,又以新的文档的完成为终点的.本阶段工作的圆满完成是以相应文挡的完成为标志,而这些文档又成为下阶段工作的基础,如图1所示.文档的质量在很大程度上决定了软件的质量,良好的文档管理是开发

* 本文1991年3月25日收到,1991年6月25日定稿.本文由青年自然科学基金资助.**周龙骧**,研究员,博士导师,主要研究领域为分布式数据处理,数据库.**柴兴无**,1990年博士毕业于中科院数学所,现读博士后,主要研究领域为分布式数据库,软件工程环境.

高质量软件产品的保证.

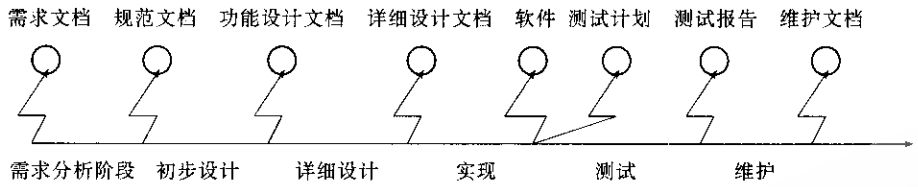

图1　软件开发过程作为文档驱动过程

3. 软件生产管理的重要组成部分

根据文档可以从事软件业务管理、人员管理、经费管理和进程管理的准备、计划、实施和检查等工作. 对文档的检查和评估也是软件生产管理的核心内容.

4. 提高软件生产率的关键

根据著名的 Brooks 定律,人之间的 intercommunication 越复杂,越会阻碍软件生产率的提高.[1]而有效的文档管理会降低 intercommunication 的复杂性,使人力集中,加速软件开发成为可能.

众所周知,管理大量的文档是极其繁重的工作,只有进行自动化的管理,即建立一个文档管理工具,才能提高文档管理的效率,从而提高软件生产率. 文档管理工具的优点在于:(1)协助用户处理大量的文件与表格,可节省人力;(2)自动解决文档的共享带来的问题,如并行操作,维护文档一致性等;(3)利用文档间的固有联系,实现文档自动生成,关联查询等功能以支持软件开发活动.

从文档管理的研究现状来看,工作大多集中在软件最终提交前的文档处理工作,工具大多是文档处理工具,如经扩充的编辑程序,它们功能单一,尚不能从整体上提高文档管理的效率. 近年来,出现了一些多功能的文档管理工具的设计模型,具有文档的定义和查询修改功能,还提出了文档一致性、完整性的概念. 这比原有的文档处理工具提高了一个层次.[4,6]但仍存在对用户不够友好、查询功能过于简单的问题,而且缺乏把它作为软件工程环境中基础工具来考虑的深度,因而其水平难于提升.

在我们设计的集成化的软件测试支撑环境 INSENST 中,设计并实现了一个既可由用户单独使用又可嵌入到软件工程环境中的文档管理工具 Doctool.[9]在本文的第一节中,我们将介绍这样一个工具应解决的基本问题,如文档的分类、标准化、查询语言特点及应具备的基本功能等. 第二节将介绍文档管理工具的面向对象的实现方法.

§ 1. 文档管理的基本问题

文档管理的目的是支持软件开发的各种活动,本节中结合 Doctool 的设计,介绍文档管理的几个基本问题.

1.1 文档的分类

软件产品的开发涉及了大量的文档,分类是最基本的工作. 最直接的方法是按照软件生存周期的各阶段划分,这种方法清晰简单,合乎软件研制的实际过程,对用户很自然. 然而对于文档管理工具的实现来讲,处理的效率不高. 因此,有人提出根据软件系统的各个组成部

分划分文档的方法,[6]即以软件系统的一个组成部分为单位,将所有文档中与之相关的部分提取出来作为文档的一个逻辑单位,这种方法便于系统内部的高效存取,但打破了传统的概念,对用户不方便.

在Doctool中吸收了这两种方法的优点,面向用户采用第一种分类方法,即按照软件生存周期的各阶段划分,而在系统内部根据文档间的联系(基本上以软件系统的组成部分为单位)实现文档的管理.

因此,从用户角度来讲,文档分为五类:分析文档Adoc、设计文档Ddoc、实现文档Idoc、测试文档Tdoc和维护文档Mdoc,它们分别代表在这五个阶段中所涉及到的所有文档,几个最典型的文档有软件需求规范SRS、设计规范DS、测试计划TP、测试设计规范TDS、测试实例规范TCS、测试过程规范TPS、测试报告TR和软件维护计划SMP.[7,8]

1.2 文档的标准化

软件生产的工程化是提高软件生产率的重要途径,而工程化必定要求标准化.在实际中,软件的设计、实现和测试是由不同的人来完成的,他们之间的交流工具就是文档.没有文档的标准化就失去了共同的基础.此外,在一个软件工程支撑环境中,基于标准化的文档,可自动进行一些软件开发和管理活动,如文档自动生成、完整性维护、检查开发进度、经费使用等,将提高软件开发的自动化程度.

在INSENST及Doctool中,我们强化文档的标准化,综合[7]及[8]制定了INSENST中的标准文档.为了不失灵活性,在标准文档中仍允许用户自定义某些栏目及内容.

1.3 查询语言的特点

同其它的信息管理系统一样,文档管理工具需提供一种查询语言.这种语言应针对文档的特点,高效且易于使用.

文档的最显著的特点是它们之间存在着紧密的联系,这些联系恰好反映了软件产品在软件生存周期中的发展过程.例如,SRS和DS间的联系对应了从需求分析阶段到设计阶段的开发过程.因此,在文档的查询应用中最常见的是查询文档之间相关联的部分,如查找软件的一个功能在SRS、DS及TP中的内容.这种查询对软件开发具有重要意义.所以说,文档间的联系对文档查询是非常关键的.

文档的联系有两种:

(1)就每类中的文档而言,基本上是层次关系,即一组相关的文档以一个文档为主,其余文档为它的子部分或补充.这些关系可从文档的栏目和内容中查到.图2给出了测试文档间的关系.[8]

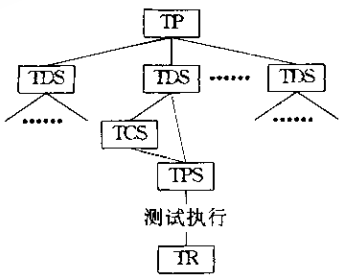

图2 测试文档间的联系

(2)就整个软件生存周期而言,是一简单的顺序关系,如图3所示.例如,对用户的任一需求,在SRS中都有着最为详尽的关于输入/处理/输出的描述.这些描述在DS中又转换成为程序模块的设计.

因此,文档查询主要有下述三种类型:

(1)同一内容在整个生存周期中各文档间的联系,例如,某个功能分别在SRS、DS或

TP 中的描述.用查询语言来表示,有如下例子:

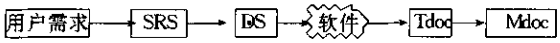

图3　软件生存周期中文档间的联系

SELECT function 1 FROM SRS,DS,TP;

SELECT function n FROM ALL;

(2)同一内容在同一阶段文档中的联系.例如,某个功能在 Tdoc 中的描述.

SELECT function 1 FROM Tdoc;

SELECT function n FROM TP,TCS;

(3)同一内容在同一文档中不同栏目间的联系.例如,某个功能在 DS 中既有功能描述,又有相对应的模块描述.

SELECT function 1 IN DS;

SELECT function n IN S1,S3 of DS;

1.4 文档管理工具应具备的功能

在 INSENST 中,我们设计了一个既可由终端用户独立使用,又可嵌入到软件工程支撑环境中的文档管理工具 Doctool,下面简单介绍它具备的功能.我们认为,这些也是软件工程支撑环境中的文档管理工具应具备的功能.

(1)注册和注销

为已注册的用户建立并维护一个完整的文档库,提供对完整文档库的复制功能,用户注销时,删除用户登录名及所有文档.

(2)安全性检查

除项目参加者外不允许任何人随意查看和更改文档内容.即使是同一项目的参加者,有时也应避免互相查阅对方文档,如测试者和实现者的工作应尽可能独立.对每类文档设置不同的口令,对项目的最高领导置特权口令.

(3)结构编辑器

编辑器是解决文档输入和直接修改的必要工具.结构编辑器为用户自动安排文档格式,用户只需关心和处理文档内容.从实际应用考虑,还应具备处理框图和程序注解的功能.多窗口功能可使用户同时参照其它文档的内容,对文档管理十分必要.

(4)版本管理

由于文档经常发生变化,有必要进行版本管理,可提高安全性,减少由意外事故造成文档的丢失,也可及时恢复用户不正确的修改.

(5)一致性维护

当用户对文档进行修改后,必然对其前和其后的文档产生影响,严重的会造成文档间的不一致.这就是文档间的修改依赖关系.文档间的修改依赖关系实际上对应于它们之间的层次和顺序关系.Doctool 在用户修改文档内容之后,根据文档间存在的联系,提醒用户可能受影响的文档及栏目.若用户没有及时作相应修改,则记录下变化,待以后用户使用这些文档时,再次提醒用户.

(6)文档自动生成

文档间存在着紧密的联系,这给自动生成带来了可能.自动生成是利用文档间的联系,在人完成了基础文档之后,由系统自动生成其后的文档的内容.例如,以 SRS 和 DS 为基础可推导出 TP 的部分内容.但自动生成困难较大,最好的方法是人机结合,即以自动生成为

基础,结合人工干预.干预的方法有事先干预、事后干预和过程干预三种.[9]

(7)文档核查

文档核查是加强软件管理的一个必要手段,它使 Doctool 更具有参与软件研制的主动性.核查内容之一是检查在软件生存周期中所有文档是否齐备,即文档库是否完整;之二是检查文档中所有栏目的内容是否齐备,即文档是否完整;之三是检查文档的一致性,即文档间的联系是否一致.

(8)文档存储

合理安排文档库结构,存储文档及目录.

(9)文档的查询与输出

提供一套完整的文档查询语言,实现各种关联查询方式.查询结果可屏幕显示、文件输出和打印输出.输出内容可由用户控制.

(10)提示帮助

提示内容包括:软件工程概念介绍、软件文档及标准介绍、Doctool 的功能及使用介绍等.

§2. 文档管理工具的实现

2.1 文档的定义

文档是由若干个栏目(section)组成,每个栏目又可由子栏目(sub—section)组成.表 1 以 TP 为例给出了文档的结构.[8]

表 1 文档及栏目的结构

Test Plan	2. Introduction
1. TP—identifier	2.1 Objectives
2. Introduction	2.2 Background
3. Test Items	2.3 Scope
4. Features to be Tested	2.4 References
……	……

从用户的角度出发,文档应具有良好的结构.由于每个栏目包括的内容较多,通常按照软件的构成或性质合理地划分子栏目.因此,子栏目是文档中描述软件产品一个部分或性质的基本单位,称为 component.它的内容一般可用子栏目的名称、关键字、文本和图表来代表.

定义 2.1:一个结构化的文档为 $d=(C,L_c)$,其中 C 是文档中 component 的集合,L_c 为 component 间联系的集合.联系是一个二元组 $link=(c_1,c_2),c_1,c_2\in C$.

这就是说,一个结构化的文档由它的构成部分及其之间的联系构成.文档间的联系同文档的内容同样重要.

定义 2.2:在结构化的文档中,component 的内容表示为 content=(TITLE,KEYWORDS,TEXT,FIGURE),其中 TITLE 为名称,KEYWORDS 是关键字集合,TEXT 为文本,FIGURE 为图表集合.

定义 2.3:一个文档类为 $CLASS=(D,L_D)$,其中 D 为结构文档的集合,L_D 为文档间联系的集合,即对结构化文档 $d_1=(C_1,L_{c_1}),d_2=(C_2,L_{c_2}),d_1\neq d_2,d_1,d_2\in D$,文档间的联系

$link=(c_1,c_2),c_1\in C_1,c_2\in C_2$.

一个文档类由结构文档的集合及结构文档间的联系的集合组成.按照 1.1 节中的文档的分类,我们定义测试文档 Tdoc 为一个 CLASS,则

$D=\{TP,TDS,TCS,TPS,TR\}$

$L_D=L_{\{TP,TDS\}}\cup L_{\{TDS,TCS\}}\cup L_{\{TDS,TPS\}}\cup L_{\{TCS,TPS\}}$

下面简单说明文档的几个重要性质.

(1)一致性

一致性包括文档间一致性和文档内部一致性两种.文档间或文档内部的联系为 $link=(c_1,c_2)$,c_1 与 c_2 属于相同的或不同的文档.一致性是指这种联系保证 c_1 与 c_2 描述的是同样的对象,只是从不同的方面去描述.实际的作法可判断 content 中的 KEYWORDS 中是否含有相同的关键字,即交不为空.

一致性的另一个含义与操作有关,当用户修改了 c_1 的内容,应检查 c_2 是否需进行相应的改变.

(2)完整性

完整性是一致性在整体上的体现,要求 CLASS 中(及 CLASS 之间)的所有文档之间与内部应保持一致性.

(3)可回溯性

在软件生存周期中,后继文档的内容应从前阶段文档中找到根据.因此,若 $link=(c_1,c_2)$,c_1 与 c_2 分别属于不同的文档,则 c_1 为 c_2 的根据,$c_2\rightarrow c_1$ 称为可回溯的.这种可回溯性表达了文档间的继承性.在软件测试中,利用可回溯性能帮助分析查找软件错误的原因.

2.2 文档管理的面向对象描述

在 1.4 节中,我们提出了文档管理工具应具备的功能.这些功能来自实际应用的总结,也是文档管理独具的.由于文档间存在着层次和顺序联系,具有较好的继承性.使用面向对象的概念和技术来描述文档的外部操作和内部实现过程是非常合适的.关键是利用文档的联系.

我们以测试文档为例,根据 TP、TDS、TCS 和 TPS 之间的联系,得到它们的对象分类结构,如图 4 所示.

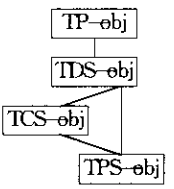

图4　文档管理的对象分类结构

对每个对象类的结构来讲,是由对应文档的 component 的状态表及文档内部和文档间的联系构成.例如,TP_obj 的结构,$S_{TP}=(SC_{TP},L_{TP}\cup L_{\{TP,TDS\}})$,其中 SC_{TP} 是 TP 中 component 的状态表,L_{TP} 是 TP 内 component 的联系集合,$L_{\{TP,TDS\}}$ 是 TP 与 TDS 间联系的集合,$L_{TP}=\{(c_1,c'_1)\mid c_1,c'_1\in C_{TP}\}$,$L_{\{TP,TDS\}}=\{(c_1,c_2)\mid c_1\in C_{TP},c_2\in C_{TDS}\}$.

TP_obj 提供的外部操作为:

(1)核查(checking)

核查的目的是检查文档的一致性和完整性.具体的做法是逐一检查文档内部及文档间的联系,利用 content 判断一致性.此外,还需检查状态表中的记录,即 component 的修改状态.对 TP 核查的内部过程为:

①检查 SC_{TP}及 L_{TP};

②若 $L_{\{TP,TDS\}}\neq\varnothing$,$\forall\ (c_1,c_2)\in L_{\{TP,TDS\}}$,$c_1\in C_{TP}$,$c_2\in C_{TDS}$,delegating(checking,$c_1$,TDS,$c_2$);

③delegating 为对象的代表性,[5]表明由子类继续实现超类的行为.在内部实现中,由 TDS_obj 检查 c_2 与 c_1 的一致性.在检查完整性时,可继续由 TDS_obj 的子类实现核查的代表性.

(2)修改(updating)

修改的实现分两步,一是对文档内容的修改,二是检查修改依赖性.

(1)对 TP 的修改

①修改 c_1 的内容,并记录 SC_{TP}状态,$c_1\in C_{TP}$;

②若 $L_{TP}\neq\varnothing$,$\forall\ (c_1,c_2)\in L_{TP}$,检查 $c_2\in C_{TP}$是否应修改;

③若 $L_{\{TP,TDS\}}\neq\varnothing$,$\forall\ (c_1,c_2)\in L_{\{TP,TDS\}}$,$c_2\in C_{TDS}$,delegating(updating,$c_1$,TDS,$c_2$),由 TDS_obj 检查 c_2;

④若 c_2 应修改,与③相同,检查 TDS 的子类是否应修改(利用代表性).

(2)继承性

根据对象的继承性,[5]应支持用户对子类文档,如 TDS 的直接修改,由 TP_obj 提供外部操作,如 updating(c,TDS),$c\in C_{TDS}$.在内部实现中,将修改 delegating 至 TDS_obj,在修改 c 之后,继续利用 delegating 检查对子类,如 TCS_obj 的相关修改,具体步骤同(1).并利用可回溯性,检查超类的相关修改,即若存在$(c_1,c)\in L_{\{TP,TDS\}}$,$c_1\in C_{TP}$,则 retrace(updating,$c_1$,TP,c).

(3)关联查询

关联查询的目的是根据 TITLE 查出 c 及与之关联的文档栏目,在 TP_obj 的外部操作中可出现对任一文档的查询,retrive(TITLE,d),其中 $d=(C,L_C)$,TITLE 为欲查询的 $c\in C$ 的名称.

①逐步 delegating 至所查询的对象类,delegating(retrive,TITLE,d),查出 c;

②若 $L_C\neq\varnothing$,$\forall\ (c,c_1)\in L_C$,查出 c_1;

③若 $L_{\{d,d_1\}}\neq\varnothing$,$d_1=(C_1,L_{C1})$,$\forall\ (c,c_1)\in L_{\{d,d_1\}}$,$c_1\in C_1$,delegating(retrive,c,$d_1$,$c_1$);

④若 $L_{\{d_1,d\}}\neq\varnothing$,$d_1=(C_1,L_{C1})$,$\forall\ (c_1,c)\in L_{\{d_1,d\}}$,$c_1\in C_1$,delegating(retrive,c,$d_1$,$c_1$).

(4)自动生成

具体的实现过程同(2)和(3)类似.由于自动生成的复杂性,最好的方法是利用联系为用户提示自动生成需考虑的因素.

(5)结构编辑

结构编辑是针对文档结构专用的编辑工具,它甚至也是文档查询的一种手段.这种编辑器同关联查询、修改一致性维护和自动生成等功能结合后,会成为一个功能很强的实用环境.例如,用户利用结构编辑器修改文档后,在屏幕上显示出与之相关的文档及栏目表,可由用户用光标选择进行相关修改的栏目.这种复杂的功能可在对象结构中的状态表中作标记,并利用联系实现.

①在用户对 $c\in C$ 编辑时,在状态表上作标记,flag(c,C);

②编辑完后,显示与 c 相关的各文档的 component 表. 若 $L_C \neq \varnothing$, $\forall (c,c_1) \in L_C, c_1 \in C$, 显示 c_1 的题目;若 $L_{(c,c_1)} \neq \varnothing$, $\forall (c,c_1) \in L_{(C,C_1)}, c_1 \in C_1$, 显示 c_1 的题目;若 $L_{(c_1,c)} \neq \varnothing$, $\forall (c_1,c) \in L_{(C_1,C)}, c_1 \in C_1$, 显示 c 的题目.

③用户挑选某个 c_1 后,显示 c_1 的内容,并作标记.

因此,我们对文档管理的五个重要功能,借助于面向对象技术,对它的结构、外部操作和内部实现过程进行了描述,实现的关键是文档的联系及对象的代表性和继承性. 这是根据文档实际应用的特点而采取的简单、高效的方法.

在 Doctool 的设计和实现中,同吴建敏博士和徐建礼博士进行了有益的讨论,在此表示感谢.

参考文献

1 Brooks, F. P. Jr, The Mythical Man—Month, Datamation, Dec. 1974.

2 Ceriani, M., Cicu, A. and Maiocchi, M., A Methodology for Accurate Software Test Specification and Audition, in Computer Program Testing, North—Holland Publishing Company, 1981.

3 Charette, R., Software Engineering Environments: Concepts and Technology, Intertext Publications, Inc. 1986.

4 Hozowitz, E. and Williamson, R. C., SODOS: A Software Documentation Support Environment —— Its Definition, IEEE Transactions on Software Engineering, Vol. SE—12, No. 8, Aug. 1986.

5 Lockemann, P. C., Object—Oriented Information Management, Decision Support Systems, North—Holland, 5 (1989), 79—102.

6 Sommerwille, I., Welland, R., Bernett, I, and Thomson, R., SOFTLIB —— A Documentation Management System, in Software —— Practice and Experience, Vol. 16, No. 2, Feb. 1986, 131—143.

7 Software Engineering Handbook, by General Electric Company, McGraw Hill Book Company, 1986.

8 Software Engineering Standards, ANSI/IEEE, 1984.

9 柴兴无, INSENST, 集成化的软件测试支撑环境, 中科院数学所博士学位论文, 1990.