超市信息系统开发过程中的规范化研究

张亚明 黄浩 (秦皇岛燕山大学 066004)

摘要	本文通过阐述超市信息系统开发过程中规范化工作的方法和内容，说明软件工程规范化的重要意义，呼吁中国软件企业以CMM(Capability Maturity Model)为标准，建立起适合我国国情的企业规范制度，增强企业市场竞争力。
关键词	信息系统 软件工程 规范化 PowerBuilder CMM

随着计算机应用在各行各业的全面普及，各种应用软件的需求量也随之不断扩大，这既是我国软件行业不断壮大的机遇，同时也是对我国软件生产能力的考验。目前，我国许多软件企业仍然处在手工作方式生产阶段，产品开发周期过长，质量不高，功能无法满足客户的要求。造成这种结果的主要原因之一是软件开发过程中，软件架构总体设计，代码的编写方式，开发人员之间的协调都缺乏统一的规范。

因此，本文通过作者在实际项目开发过程中的经验，阐述软件开发规范化的意义和方法。

1 软件工程规范化的意义和层次

1.1 软件工程标准化的意义

仅就一个软件开发项目来说，有许多层次、不同分工的人员相互配合，在开发项目的各个部分及各开发阶段之间也都存在着许多联系和衔接问题。如何把这些错综复杂的关系协调好，需要有一系列统一的约束和规定。在软件开发项目取得阶段成果或最后完成时，需要进行阶段评审和验收测试。投入运行的软件，其维护工作中遇到的问题又与开发工作有着密切的关系。软件的管理工作则渗透到软件生存期的每一个环节。所有这些都要求提供统一的行动规范和衡量准则，使得各种工作都能有章可循。

软件工程的标准化会给软件工程带来许多好处，比如：

(1) 可提高软件的可靠性和可移植性（这表明软件工程标准化可提高软件产品的质量）；

(2) 提高软件的生产率；

(3) 提高软件人员的技术水平；

(4) 提高软件人员之间的通信效率，减少差错和误解；

(5) 有利于降低软件产品的成本和运行维护成本；

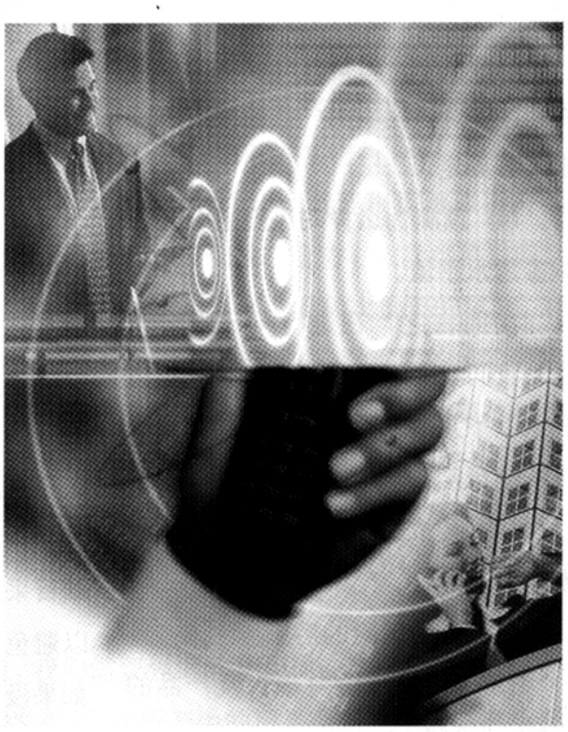
(6) 有利于缩短软件开发周期。

1.2 软件工程标准的层次

根据软件工程标准制定的机构和标准适用的范围有所不同，它可分为五个级别，即国际标准、国家标准、行业标准、企业（机构）标准及项目（课题）标准。以下分别对五级标准做一简要说明：

(1) 国际标准。由国际联合机构制定和公布，提供各国参考的标准。如：

ISO (International Standards Organization) 国际标准化组织。这一国际机构有着广泛的代表性和权威性，它所公布的标准也有较大的影



响。60年代初，该机构建立了"计算机与信息处理技术委员会"，简称ISO/TC97，专门负责与计算机有关的标准化工作。

(2) 国家标准。由政府或国家级的机构制定或批准，适用于全国范围的标准。

(3) 行业标准。由行业机构、学术团体或国防机构制定，并适用于某个业务领域的标准。

(4) 企业规范。一些大型企业或公司，由于软件工程工作的需要，制定适用于本部门的规范。例如，美国IBM公司通用产品部（General Products Division）1984年制定的《程序设计开发指南》，仅供该公司内部使用。

(5) 项目规范。由某一科研生产项目组织制定，且为该项任务专用的软件工程规范。

上述五个标准层次之间相互联系，底层标准是制定上层标准的基础，同时，上层标准是底层标准的具体化和延伸。其中，项目规范是最为详细具体的，它直接指导软件产品开发的全过程，是项目成败的关键。下面，我们以超市信息系统开发过程中的标准来体会一下项目规范的方方面面和它的重要性。

2 超市信息系统的开发规范

2.1 代码编写的规范化

现代软件工程不仅要求程序算法规范化，同时要求代码的编写形式尽量统一。这不仅有利于程序的改写和维护，也为程序开发人员之间交流提供了方便。在超市系统开发过程中，代码编写约定大致分为：

(1) 命名约定

命名规则主要采用匈牙利命名法。本项目中作如下规定：

数据字段的命名按照汉语拼音的首写字母组合。如果字段名称汉语拼音的缩写字母组合相同，那么选择其中一个有区别的汉字，写出它的汉语拼音全称。

① 对象的命名规则：

框架窗口。单框架应用:w_frame

多框架应用：w_frame_＜模块汉语拼音字头＞

工作表窗口前缀:w_＜上层模块汉语拼音字头＞_

应用级响应窗口前缀：w_pub_

模块级响应窗口前缀：w_pop_＜调用窗口自身名＞_

DATAWINDOW对象前缀：d_

DATAWINDOW控件前缀：dw_

用户自定义可视对象：vu_ 标准可视对象：vu_st_

非可视对象：cu_

项目：pr_

其他窗口内控件采用PB系统默认前缀。

② 变量命名规则：

公共变量前缀：g_

实例变量前缀：i_

共享变量前缀：sh_

局部变量前缀：l_（或）无(常用数据类型的局部变量不加作用域前缀，对象类的变量加作用域前缀。)

数组：在类型前缀后加a

DATAWINDOWCHILD:dwc_

③ 其他

应用级函数：f_ 应用级结构：s_ 应用级管道对象：pl_

应用程序函数：af_ 应用程序结构：as_

窗口函数：wf_ 窗口结构：ws_

菜单函数：mf_ 菜单结构：ms_

用户对象函数：uf_ 用户对象结构：us_

窗口自定义事件：ue_

传递的参数(函数，事件)：类型前缀前加a

在消息中传递的参数：类型前缀前加mg

通过约定，方便了程序员记忆、理解、使用各种变量，提高了编程效率和质量。

(2) 注释规范

① 函数注释

在函数开头，用如下格式注释：

```
////////////////////////////
///Function: f_save_toolbar_profile
//Access: public
//Arguments: aw_Window The window whose toolbars are being saved.
//Returns: None.
//Description: Save the current toolbar information.
////////////////////////////
```

通过对函数的说明，方便了其他人员使用和维护函数。

② 变量注释

对于非通用的变量，在定义时加以注释说明，变量定义尽可能放在最开始处。

③ 文件注释

在文件开头注释以下内容：

```
////////////////////////////
// Project: 文件所在的项目名，如：CSXT
// By: 作者、修改者、...，如：HHAO
// Description: 说明文件的功能。
////////////////////////////
```

④ 其他

函数内各功能模块，如：循环、流程的各分支等，尽可能多的加以注释。

(3) 编程风格

① 缩进形式

缩进(indent),是一个保证代码整洁、层次清晰的主要手段。良好的缩进形式有助于程序的阅读和调试。

② 错误处理方式

编程中要考虑函数的各种执行情况，尽可能处理所有流程情况。

将函数分两类：

一类为与屏幕的显示无关，

二类为与屏幕的显示有关。

对于与屏幕显示无关的函数，函数通过返回值来报告错误。

对于与屏幕显示有关的函数，函数要负责向用户发出警告，并进行错误处理。

错误处理代码一般放在函数末尾。

(4) 模块化

某一功能如果重复实现三遍以上，即应考虑模块化，将它写成通用函数或类。并向小组成员发布。同时要尽可能利用其他人的现成函数或类。模块化的一些注意事项如下：

① 设计好模块接口，用面向对象的观点看，包括：函数接口和变量接口。

② 定义好接口以后不要轻易改动，并在模块开头（文件的开头或函数的开头）加以说明。

③ 不要在外部使用接口以外的其他函数或变量。

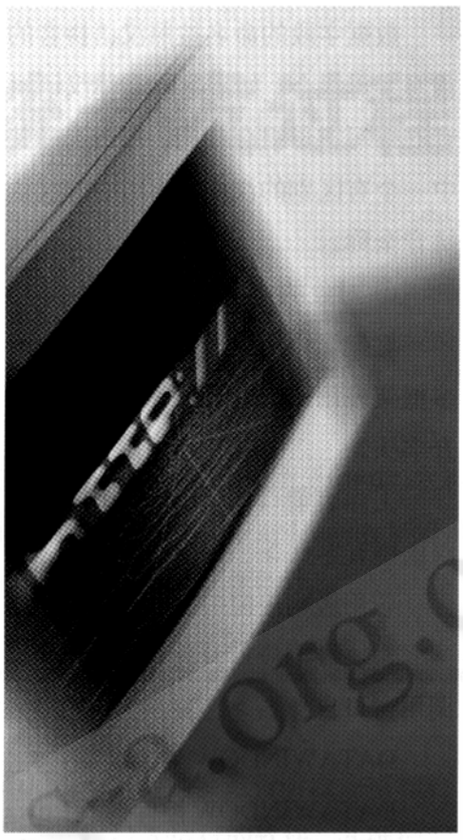
④ 注意全局变量也是一种接口。

⑤ 接口即是罗列出一个模块的所有的与外部打交道的变量、函数等。

下面是用PowerBuilder开发超市系统中三数据窗管理（w_mst_dj）基础类的一部分设计：

edit

ue_addrow isr_wzt='add' -->dw_head_detial.ue_addrow() isr_dzt='add'



ce_dwaddrow()--> dw_head_grid.uf_dwsetdzt()

uf_dwsetdzt()

规则:Uf_detailadd() 设置初始值置 il_addrow

ue_updrow isr_wzt='upd'-->dw_head_detial.ue_updrow() isr_dzt='upd'

ce_dwupdrow()-->dw_head_grid.uf_dwsetdzt()

ue-save dw_head_detial.ue_save()-->ce_save<----> parent.wf_head_save()

规则:uf_save()

<>dw_head_grid.uf_setdzt()

Uf_dwsetrow()

dw_head_detial.ue_enter<>lastcol-->ce_enter<----> dw_head_detial.ue_save-->ce_save<----> parent.wf_head_save()-->规则:uf_save

<>成功-->parent.isr_wzt='addmx';parent.ue_mxadd

ue_cancel dw_head_detial ue_cancel-->ce_cancel<----> parent.wf_edit()<---->规则:uf_cancel()

<>dw_head_grid.uf_setdzt()

Uf_dwsetrow()

ue_mxadd parent.wf_editmx() dw_head_detial.enabled=false

dw_head_grid.enabled=fasle

ue_mxupd parent.wf_editmx()

ue_mxsave parent.wf_editmx()

ue_mxcancel parent.wf_editmx()

close closequery

通用对象的设计关键是根据业务逻辑进行接口设计，上面ce_为前缀的事件是接口事件，大多数对象操作、校验规则和商业逻辑都可以在此编写，极大方便了程序的扩展。因此在程序开发中通过对某类操作的规范，使之通用化，可以提高开发效率。

2.2 软件开发人员之间的协调规范

软件开发需要一个团队的共同努力，各成员之间的工作必须保持协调才能使整个项目顺利进行，因此有必要制定一个开发组协同准则来规范开发人员的开发行为。例如在超市系统开发中，小组成员之间的协同守则如下：

(1) 本守则适用于超市系统开发组。

(2) 组内所有开发人员连接同一个网络数据库。对数据库的任何修改以日志文档形式记录。

(3) 公共库文件tzxc_1.pbl和tzxf_1.pbl存放在\\sever\Pbgroup\目录下。

超市系统应用库文件cs.pbl、公共库文件pub.pbl存放在\\sever\Pbgroup\cs\drv目录下，其中tcspub.pbl包含在每一个组员应用程序的库搜索列表中。

(4) 网络中共享的库文件含义：

csc_1.pbl与csf_1.pbl为部门级应

用框架、类库。适用于任意一种应用。

pub.pbl为应用级类库，存放在超市应用中可能为所有人员共享的对象。

每一对象必须在库同名文档中说明。格式：

<对象名>，<功能说明>，<制做人>，<更新日期>，<接口>

应用的其他库文件存放在\\sever\Pbgroup\cs\drv目录下，在各自的机器上存有备份，注意保持同步。

(5) 如果应用中要使用其他人员所创建的对象，必须将其拷入pub.pbl中，并在日志文件中记录，如需对其进行修改，必须先与对象的创建者商讨，并记录修改。

(6) 对发布到网络上的所有库(pbl)文件中的对象的修改，必须采用注销的方式进行，并将修改内容记录到与库同名的日志文档中。

(7) 自我测试后的应用部分拷入\\sever\Pbgroup\cs\test目录下，但要保持与原库文件同步，此目录下文件禁止修改。

开发组协同规范是为了避免由于开发人员的开发习惯不同，造成应用程序各个模块之间的冲突以及整个系统的混乱。因此它是程序开发中必不可少的。

2.3 软件外观界面的规范化

软件的界面美观、方便易用往往决定了软件的成败，特别是商品化的小软件，故外观界面规范更应该仔细制定实施。现在的界面一般都基于微软的Windows风格，它包括各种界面元素如按纽、树、页面等，对这些可视元素（控件）应该制定一套涉及大小、颜色、字体的一般规则及功能作用的约定。要严格区分使用各种容易被混淆的元素如RadioButton与CheckBox,要确定一些控件的合理使用限度如Listbox与DropdownListbox的列表数目，要严格要求在开发过程使用统一的、持续的一套界面风格，在多人开发时更应该注意。

2.4 软件测试流程的规范化

软件测试是软件开发过程中关键环节之一，必须对软件功能进行详尽的全面测试，尽可能发现所有隐藏的错误，只有这样才能提高软件质量。按照规范的流程测试软件则是发现错误的最佳方法。具体项目的测试内容、测试方法、测试流程不尽相同，但大致应按如下的规范去测试：

① 代码的逻辑测试；

② 业务的逻辑流程测试；

③ 单机条件下软件的整体测试；

④ 联网条件下软件的整体测试；

⑤ 软件与系统平台的兼容测试。

2.5 文档制作和管理的规范化

软件文档(document)也称文件，文档本身就是软件产品的一部分。没有文档的软件，不成其为软件，更谈不到软件产品。软件文档的编制(documentation)在软件开发工作中占有突出的地位和相当的工作量。软件文档对软件的开发效率、软件质量等有着重要意义。软件的整个寿命周期存在大量文档，其中程序开发阶段大致要生成如下一些文档：

用户文档：用户手册，操作手册，维护修改建议，软件需求说明书。

开发文档：软件需求说明书，数据要求说明书，概要设计说明书，详细设计说明书，项目开发计划。

管理文档：项目开发计划，测试计划，测试报告，开发进度月报，开发总结报告。

上述文档都有各自的制作规范，按规范生成各类文档，才能使软件开发的整个过程受到良好的控制，保证软件开发的顺利进行。

3 结束语

本文所论述的只是软件生产过程中某一阶段的规范化管理，除此之外，其他各阶段也有大量的已经规范化的标准和有待于规范化的工作去完成。事实上，软件开发过程的规范化是一个渐进的过程，是在长期的软件生产过程中不断积累，逐步走向完善的。所幸的是，我们有CMM(Capability Maturity Model)的指导，只要软件企业按照CMM的要求去规范企业的生产与经营，就会大大提高自身的竞争力。■

参考文献

1 PowerBilder程序设计大全，机械工业出版社，1997.9。

2 软件工程，清华大学出版社，1997。