Z-Model：一种支持 Web 应用快速开发的框架模型

曾滢著[1]　沈立炜[1]　赵文耘[1]

(复旦大学 计算机科学技术学院,上海　201203)[1]

摘　要　通用的 Web 开发框架难以满足快速应用开发的需求。同时，软件企业提出的 Web 应用快速开发框架也存在诸如缺乏建模分析能力、不适应变更需求、页面交互不够灵活等问题。针对这些关注点，提出一种支持 Web 应用快速开发的框架模型 Z-Model 及其图形化流程建模机制。Z-Model 从实际 Web 应用的需求着手，发掘并整理出 Web 应用中的基本元素，并使用元模型对其进行系统化的整合。在此基础上，开发者可使用基于 Z-Model 的开发平台 WF，通过对业务流程图建模以及对已配置功能单元绑定的方式快速地开发出应用产品，同时能够迅速适应变更的用户需求。最后，通过对一个员工信息管理 Web 应用的开发与维护展示了 Z-Model 的有效性。

关键词　Web 应用快速开发框架，业务流程图，上下文变量，功能单元，事件和动作

中图法分类号　TP301　　**文献标识码**　A

Z-Model：A Framework Model Supporting Rapid Web Application Development

ZENG Ying-Zhu[1]　SHEN Li-Wei1[1]　ZHAO Wen-Yun[1]

(School of Computer Science and Technology, Fudan University, Shanghai 201203, China)[1]

Abstract　Common Web development framework is hard to fit the rapid development of Web applications. Meanwhile, frameworks for fast Web application development proposed by software enterprises include the problems such as the lack of modeling and analyzing capability, the inadaptation towards the changing requirements and the inflexibility of page interaction. Aiming at these concerns, this paper proposes a framework model supporting fast Web application development, named Z-Model, as well as its graphic representation mechanism. Z-Model extracts and classifies the Web elements from the real Web application requirements, and integrates them systematically in the meat-model. Based on it, developers can generate application in a fast mode by means of modeling the business flow chart and binding the configured functional units in the WF platform that is created on Z-Model. In addition, the Web application can quickly adapt to the changing user requirements. Our paper validates the effectiveness of Z-Model through the developing and maintaining upon an Employee Management Web Application.

Keywords　Rapid Web application development framework, transaction chart flow, context variable, functional unit, event & action

1　概述

随着互联网技术的普及，基于 B/S 架构的 Web 应用系统已被广泛使用。Web 应用系统相对于传统的 C/S 架构软件而言，其易用性、通用性及可扩展性等诸多优势使得用户可以通过浏览器使用系统带来的功能，免除本地安装、升级等繁琐工作。在 Web 应用系统开发领域，研究者或企业界提出了一系列通用开发框架，其中包括被广泛接受的 Struts[1]、Spring[2] 等。大部分框架遵循 MVC（Model-View-Controller）[3]规范，使得不同角色的开发人员可以在关注点分离（Separation of Concern）的原则下进行开发，从而使得项目团队职责更为明晰，同时能有效保障产品质量。

通用的 Web 开发框架规定了系统的组成架构，开发者可以在预定义规范的指导下通过编写程序代码开发出灵活的、适应不同业务需求的 Web 应用产品。由于业务需求的复杂程度不同，客户的偏好性差异以及程序员的开发习惯，导致了当前 Web 应用系统操作界面风格的多样性和庞杂性。显然，这种多样性增加了软件的开发成本和维护成本，制约了软件规范化的进程和工程工业化的发展。因此，在高度竞争的 Web 应用开发领域，准确、经济且高效的开发模式显得尤为重要。一种有效提高生产率的方法即降低软件的复杂程度[4]。尤其对于已被大规模使用的 Web 信息管理系统而言，界面的多样化已不再是开发的重点。所以，当前已有许多软件企业开发出不同类型的 Web 应用快速开发框架。每套框架提供统一的用户界面，并尽量采用配置的操作方式开发 Web 业务逻辑，从而大幅度降低对开发者编码能力的要求。另外，快速开发框架中均包含可复用的功能工具，例如报表工具、工作流引擎等。现阶段，国内比较著名的商用快速开发框架包括东大金智的 EPStar[8]，上海普元的 EOS[9]，北京金富瑞的 UCML[10]，上海群萃的 Extraction[11]以及上海锐道的 Dorado[12]等。其中，东大金智推出的企业级应用系统集成环境 EPStar 能在网络办公、工作流等方面通过动态建模技术迅速实现具体应用的企业开发，但其表现层是以页面为单位，因此页面基本固化，无法在单页面内实现复杂的业务操作[8]。另外，上海锐道公司推出的拥有独立 IDE 的 dorado studio 能快速实现数据表到浏览器视图的映射和简单页面生成，使程序员不必关心系统各层次的具体实现，但是它不支持逆向操作，无法对生成的页面进行修改，以完成更复杂的业务逻辑[12]。

本文受国家自然科学基金（No.90818009）资助

曾滢著(1985－),男,硕士研究生,研究方向为 Web2.0 及框架开发技术; **沈立炜**(1982－),男,博士,助理研究员,主要研究领域为软件产品线,自适应系统等; **赵文耘**(1964－),男,教授, 主要研究方向为软件工程、基于构件的软件开发等。

因此，在对以上快速开发框架调研的基础上，我们总结出现有框架在以下三个方面的局限性：

1) 缺乏建模分析能力，未能将应用场景抽象，造成业务要素缺失，导致了应用范围的局限性。
2) 未从业务需求的角度入手，开发过程与实际业务场景结合不不紧密，导致了应付需求变化的局限性。
3) 过分强调 web 应用的功能实现，缺乏基于用户交互设计理论驱动的系统设计[6]，导致页面交互的局限性。

针对以上问题，本文提出了一种支持 Web 应用快速开发的框架模型 Z-Model 及其图形化流程建模机制。Z-Model 从实际 Web 应用的需求着手，发掘并整理出 Web 应用中的基本元素，并使用元模型对其进行系统化的整合。在此基础上介绍开发平台 WF，开发者可在此平台上通过对业务流程图建模以及对已配置功能单元绑定的方式快速地开发出应用产品，同时能够迅速适应变更的用户需求。

2 Web 应用实例及其元素抽象

在本小节中，我们将通过一个具体的系统实例挖掘并整理出 Web 应用中存在的元素。图 1 展示一个用户在浏览器中维护员工信息的人机交互场景及步骤。与图中用户操作步骤相应的机器处理过程如下：

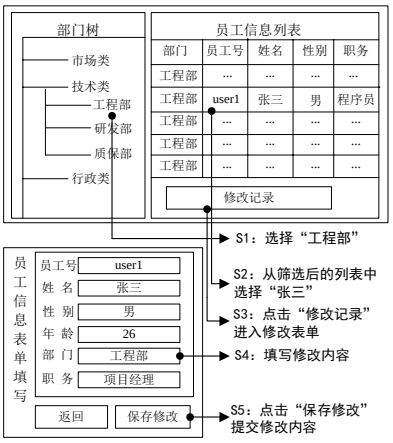


图 1 员工信息管理操作场景

S1：1）部门树获得了焦点，触发点击事件；2）提取焦点数据（上下文）“工程部”；3）查询列表获得菜单树的提供的上下文“工程部”；4）将“工程部”作为筛选条件发送到服务器请求数据，返回结果后加载数据刷新列表。

S2：员工信息查询列表获得了行焦点。

S3：1）触发按钮点击事件；2）检查查询列表，确保列表中有一被选中 （“张三”被选中）；3）提取“张三”所在行的所有字段信息（上下文）；4）跳转到表单；5）表单获得查询列表提供的字段信息（上下文）；6）将各个字段信息，填写到相应字段输入框中。

S4：表单处理用户编辑。

S5：1）触发按钮点击事件；2）检查表单各个输入字段的正确性，如年龄位于 1-100 之间等等；3）提取表单各个输入字段信息（上下文）；4）发送表单所有字段信息至服务器，并执行对“张三”员工信息的修改；5）服务器返回成功后，跳转到查询列表主界面；6）查询列表刷新修改后的行记录。

我们在此场景基础上抽取 Web 应用系统中的基本元素并加以分类。分类结果如表 1 所示。

要素被分类为组织元素与过程元素两大类别。组织元素主要表示 Web 应用的界面要素，包括功能单元、业务窗口与按钮，这些元素都能显式地被用户所感知。另一方面，过程元素主要表示组成业务逻辑处理的要素，包括事件、上下文、动作、约束以及持久化请求。这类元素在前台运行，支持着 Web 应用系统的业务操作表现。

表 1 Web 应用系统要素分类

	要素名称	示例
组织元素	功能单元	部门树、人员信息列表、修改表单
	业务窗口	盛放各种功能单元的容器
	按钮	“保存修改”,“返回”
过程元素	事件	树点击节点、列表选中行、按钮点击、表单输入键盘敲击
	上下文	用户选中树节点后提取的数据，选中列表行后提取的数据，表单输入框编辑后的内容
	动作	跳转到窗口，筛选人员信息列表
	约束	第三步中列表检查是否选择了行；第五步中表单提交前检查数据合法性，且表单提交成功后才执行跳转主界面
	持久化请求	向服务器提交修改表单

3 支持快速应用开发的 Z-Model

在本小节中，我们将介绍支持快速 Web 应用开发的框架模型 Z-Model。Z-Model 来源于对大量信息管理业务系统的调研结果，并从这些系统中抽象出较为通用的使用模式与界面风格。因此，Z-Model 可被视为一种在不丢失功能的前提下简化实际业务系统体系结构的 Web 开发框架。

3.1 Z-Model的元模型

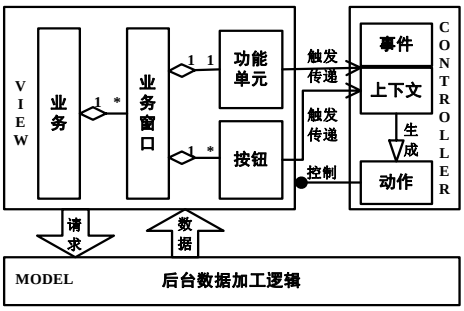


图 2 符合 MVC 的 Z-Model 元模型

Z-Model 的元模型如图 2 所示。该元模型涵盖前一小节所抽象出的 Web 系统基本元素，同时该模型符合 MVC 模式。后台业务数据加工逻辑（Model）为接受前台的数据请求并为功能单元提供基础数据。数据由功能单元组织表达并展现在窗口中（View）。同时位于前端动作处理机制通过事件的触发和上下文的传递实现了对业务窗口的展现控制（Controller）。从 Z-Model 的元模型中可以归纳出 Z-Model 的如下特征：

1) 一个业务系统由一组业务构成（在 B/S 风格下，业务即表现为一个具体的页面）；
2) 一个业务由若干功能完备的业务窗口构成；
3) 一个业务窗口包含一组可操作的按钮和一个独立的功能单元；
4) 窗口与窗口之间的消息传递通过事件的触发和捕获来完成；
5) 业务窗口的展现顺序和状态由事件串联；
6) 事件只能由按钮和显式注册对其捕获的功能单元所触发；
7) 事件是单一的，即不存在一个事件触发后同时关联触发其他事件；
8) 用户最终的业务提交请求（数据持久化）通过点击可罗列的通用按钮完成。

第 2 节中所介绍的 Web 实例即符合 Z-Model 的组织架构。首先它由三个基本业务窗口构成，分别包含了树、列表、表单三个功能单元。同时，窗口的消息传递通过事件触发来

完成：点击窗口上的按钮会后触发事件，这些事件产生相应动作并决定随后展现的具体业务窗口，用户最终的修改通过表单窗口的“保存修改”按钮实现。

Z-Model 能够有效地简化复杂的 Web 应用系统架构。例如，特征 1、2 限定了系统整体的组织形式，同时遵循 B/S 的系统风格。特征 3 限定了功能单元组装的范围，并相应明确该模型下功能的构成粒度。特征 4 减少了功能与功能之间的耦合。特征 5 限定了用户与业务对象交互的过程。特征 6、7 减少了不必要的事件处理并将事件单一化，降低业务操作难度。特征 8 最为重要，它将实现数据持久化（包含一定数据加工的增、删、改）的驱动限定在按钮上，使得功能单元只需关心数据的组织和展现（查），实现了功能单元的高度内聚，降低了功能与事件驱动的耦合性。Z-Model 在允许的范围内牺牲了用户体验，但更为重要的是为大规模构件集成复用和在信息管理领域内实现软件工业化工程化提供了可行性。

3.2 上下文变量

上下文变量是从用户进入业务系统开始直至退出系统的过程中，由用户操作所产生的所有相关数据的集合（上下文、变量、上下文变量皆为同义词）。在员工信息维护示例中，各个功能单元包含自己的数据，用户在操作（点击节点、选中行、编辑输入框）这些功能的同时产生了上下文。这些上下文变量在各个功能单元之间传递，有些则提交至后台服务器进行处理。在整个业务操作过程中，上下文变量的产生和传递起到了至关重要的作用，它的产生方式反映了功能单元的特性，同时它的传递过程则表达了业务操作的实际过程。我们将上下文变量分为三种类型：

1）全局变量：在用户登入系统初始化时产生的一组数据，只要用户不重新登入，则其值不会改变。例如：用户名，用户 ID,用户角色,登录 IP，当前年份等。

2）常规变量：用户在操作具体功能单元时产生的变量，随着用户的操作焦点的改变而改变，是功能单元包含的业务数据的子集。例如：树选中的节点内容（“工程部”），查询列表选中的行内容（“张三”），表单输入的字段内容。

3）特征变量：用户在操作具体功能单元时产生的具有功能构件特征的变量，随着用户的操作焦点的改变而改变。例如：当前操作节点所在树中的层次，查询列表当前被选中的行的数目，列表查询的总结果数。

变量的分类具有重要意义。一方面，它在一定程度上表述了变量与事件的关系，因为除全局变量外的上下文产生都依赖于事件的触发；另一方面，它规定常规变量和特征变量都是功能单元的数据内容，这样就进一步明确了该模型下的功能具有维护数据的职责。为下文表述方便，我们引入变量符号 #变量名# 描述一个具体的变量。例如，#userID# 可以表示当前用户的 ID 号。

3.3 业务窗口和功能单元

业务窗口是包含一个功能单元和一组事件按钮的容器，但窗口本身不具有任何动作行为。其中，功能单元是一种将相关数据按照一定的形式组织表达而成的变量容器，对内负责维护数据，对外负责触发事件和接受动作。Z-Model 中，功能单元之间不能直接交互，单元的消息传递由事件响应处理机制和上下文机制完成。因此，当功能单元被实例化时，它将被赋予相关的上下文与事件信息，从而构成具体的业务逻辑。在员工信息维护示例中，查询列表是一个实例化后的功能单元，它将数据以列表的组织形式表现出来。同时，它能够响应处理“选择行”事件，产生上下文变量，并且接受来自外部的“影响”动作，最终刷新列表。

Z-Model 在简化业务操作模型的情况下，也为运行其上的功能单元提出了要求。根据上述关于功能的定义和其业务窗口的关系以及功能实例化的定义，我们归纳出以下功能单元必备的特征：

1）完整的生命周期(初始化、运行和销毁的过程)；

2）含有业务数据和特征数据；

3）提供 get 和 set 数据的方法；

4）按一定形式提供数据展现；

5）含有可触发的事件列表；

6）提供接受外来影响的方法；

7）提供对自身数据完备性检测方法（上下文相关性检测）；

8）一组供开发者使用的可配置项。

另一方面，按照数据源的不同，我们将功能单元划分为三种类型，如表 2 所示。

表 2 功能单元的分类

类型	数据源	举例
输入型功能单元	用户输入	表单，图像扫描
输出型功能单元	持久层（数据库或文件系统）	菜单树、图
混合型功能单元	用户输入和持久层	可编辑的 grid

3.4 事件、动作及其约束

3.4.1 事件

Z-Model 中的事件涵盖以下两方面的概念：首先，事件由功能单元触发，如选择树的节点，选择查询的行等操作行为。此类事件要求功能单元必须显式地注册或声明对其的捕获。例如，特定功能单元在集成时显式地声明了对‘选中节点’事件的处理，而未显式声明对‘鼠标移入’事件的处理，则在树节点被点击时捕获了‘选中节点’事件，但无法捕获‘鼠标移入’某节点这个事件。类似的，如果一个查询列表要捕获‘数据完成加载’事件，就必须显式地在功能单元集成时声明对‘数据完成加载’事件的捕获。其次，事件由按钮触发，如点击‘修改记录’按钮可触发‘修改记录’事件，点击‘保存修改’按钮可触发‘保存修改’事件，即按钮名称可以表达其触发的事件的名称。

Z-Model 下的事件，无论是由功能单元触发还是由按钮触发，都是以显式触发形式存在，未注册的事件将无法被系统所关注。另外一方面，从事件动作处理的过程中可以看出事件动作的触发、处理与上下文之间存在依赖关系，即事件与上下文之间的约束。例如，当用户点击按钮“修改记录”时，相应事件被触发，同时要求查询列表必须选择了某一行记录，否则下一步动作无法进行。又如，表单在提交之前需要保证各个字段数据的合法性，否则不予保存。由此，根据约束条件，我们进一步引出事件的两种分类：

1） 上下文相关：事件执行依赖于某些功能单元产生的上下文，称该事件上下文相关。例如“修改记录”是一个上下文相关事件。

2） 上下文无关：事件的执行不依赖于任何功能单元产生的上下文，称该事件上下文无关。例如“返回上一步”是一个上下文无关事件。

3.4.2 动作

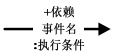
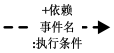
动作是事件触发后产生的效果。目前，随着基于 Z-Model 的开发框架不断发展，动作的形式也有进一步扩充，本文只介绍最基本的两种：转移和影响。在员工信息维护示例中，点击“修改记录”按钮后产生跳转动作可被视为一个“转移”动作，点击树节点后筛选了查询列表的记录则是一个“影响”动作。转移动作引起了窗口展现的变化，影响动作引起了功能单元内容的变化。

动作的执行是有条件的，即动作与上下文之间的约束。例如在员工信息维护示例中，当用户选择菜单树中第一层上的“技术类”节点时，虽然菜单树触发了“选择节点”事件，但并未执行对查询列表的影响动作。其原因在于仅当用户点击第二层上的“工程部”节点时，才能产生对查询列表记录筛选的影响动作。那么假设该菜单树用变量 currLevel 表示当前点击的层,我们使用变量条件式来描述执行该动作的约束： #currLevel# == 2

一组事件通过约束能够组合成更大粒度的业务流程，同时流程可以通过类似于 IF-ELSE 的程序结构进行描述。事件动作过程的程序化和通用化可以减少程序开发过程中事件动作的不确定性，明确了 Z-Model 下的事件动作处理机制。

4 基于 Z-Model 的图形化建模

表 3 业务流程图图元详解

图 元	名 称	元素对应关系	说 明
	起点		业务流程图的入口，流图有且只有一个开始标记,且只能连出不能连入。
	终点		业务流程图的结束标记，流图可以有 0 个或者多个结束标记，且只能连入不能连出。
	主窗口	一个包含功能单元的窗口	包含了一个实例化构件单元的窗口，窗口拥有系统唯一的实例化编号，是同步页面的驱动窗口。
	从窗口	一个包含功能单元的窗口	包含了一个实例化构件单元的窗口，窗口拥有系统唯一的编号。
	同步条		可以挂载一组需要同时展示的业务窗口。一个同步条下有且仅有一个主窗口。
	同步线		用于节点的连接，线的起始节点只能是起点、同步条和标记；终止节点只能是主窗口、从窗口。
	转移线	对应一个事件的触发，约束，和跳转动作，也即一个按钮事件触发后执行跳转过程。	表示一个动作转移。‘：’后面表示动作执行条件，该条件是一个含有上下文变量的布尔表达式，缺省表示 1==1。‘+’后面表示事件依赖的上下文窗口编号。分三种情况：1.不写表示该事件是上下文无关的；2.只写‘+’则表示默认依赖事件源所在的窗口号；3 事件依赖多个窗口产生的上下文时，窗口编号用逗号分隔。线的起始节点只能是主窗口和从窗口；终止节点只能是主窗口、从窗口和同步条、标记
	影响线	对应一个事件的触发，约束，和影响动作，也即一个按钮或功能单元事件触发后执行影响的过程。	表示一个影响动作，线上标注含义与转移线相同。线的起始节点和终止节点都只能是主窗口、从窗口；且影响线不可夸越同步条，源节点和目标节点只能由同一个同步条可达。
	标记		表示了一根连线，当业务流程图庞大时，需要用标记来标注线的走向，以简化绘图。标记一定是对偶出现的，即 “ ->标记 ”和 “标记->”一定同时存在于图中。

在实践工作中，我们开发出一套图形化机制用以描述 Z-Model 的业务流程，称其为业务流程图。该流程图可方便且精确地描述符合 Z-Model 的业务场景，使得开发人员能够基于该图形开发、定制出符合 Z-Model 框架的 Web 应用系统。

表 3 列出了业务流程图最基本的图形元素。业务流程图元按类型可以分为两大类，第一大类表示业务组织成分，包括主窗口、从窗口；第二类表示连接线，包括：转移线、影响线和同步条标记等。一个业务流程图除了其每个图元需要符合表中说明的自身约束条件之外，还需满足其他整体约束。由于业务流程图的约束特性与完备特性不是本文的重点，因此不做详细展开。我们列出流程图需要遵循的最基本的两点约束：

1) 一个业务流程图必须开始于一个起始标记，但可以没有结束标记；

2) 通过起点沿转移线或同步线可以到达图中任何一个节点。

5 案例研究与讨论

5.1 案例实现

基于 Z-Model 及业务流程表示机制，复旦大学软件工程实验室已经开发出支持快速 Web 应用开发的平台 WF。WF 中，目标业务系统可通过对业务流程图建模以及对已配置功能单元绑定的方式实现。

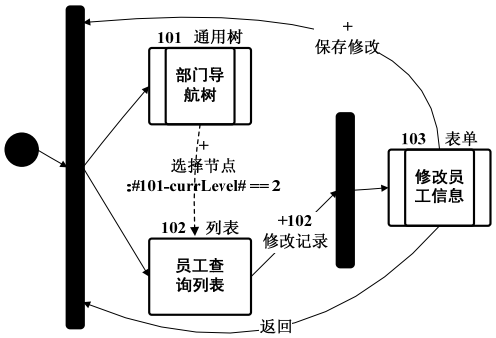


图 3 员工信息管理的业务流程图

图 3 描绘了第 2 节中员工信息管理系统的业务流程图，其中包含了各类窗体、窗体间的连接及约束等信息（使用表 3 中的图元）。在具体定制阶段，窗体可绑定特定的已配置功能单元，从而引入功能单元所包含的业务逻辑。

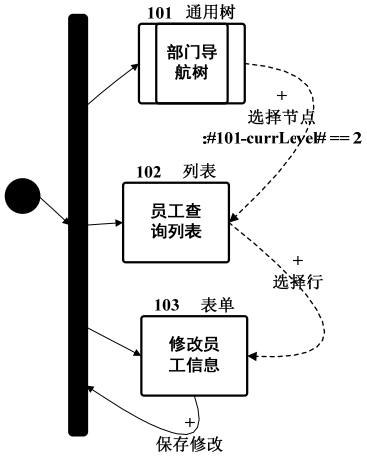


图 4 更改后的员工信息管理的业务流程图

图 3 展示的业务流程图表示在初始页面中仅显示部门树与人员列表，而信息填写表单则通过按钮事件转移出现。当用户希望初始操作界面中同时出现树、列表与表单时，可在 WF 中简单修改业务流程图，包括修改窗体内容（如去除

修改记录的按钮）以及相关事件动作信息（列表影响表单，自动填入），即可满足该操作变更。修改后的业务流程图如图 4 所示。由此对应图 4，通过 WF 配置生成的员工信息维护 Web 应用系统的操作界面如图 5 所示。

图 5 员工信息管理 Web 应用的操作界面

5.2 讨论

基于 Z-Model 的 WF 快速开发平台已被推广至企业应用，并成功地开发出数十个 Web 业务系统。与其他流行的 Web 开发框架相比，使用 WF 进行开发可以提高 3 倍以上的效率，尤其在常规业务场景的开发中实现了零编码。此外，Z-Model 所带来的优势使得 WF 开发具有高度的便捷性，可维护性和扩展性，这些优势包括如下几点：

1）Z-Model 缩小了问题规模，在不丧失功能和灵活度的前提下简化人机交互模式，将问题专注于信息管理领域，提供了更明确的业务场景建模。

2）基于 Z-Model 开发的功能粒度是业务级的而非对象级的。用基于 Z-Model 的业务流程图建模，能够快速、精准地描述用户操作的具体业务场景，图中的要素与业务系统的运行元素之间具有明确的对应，因此可基于图形化建模结果自动生成可运行的配置信息，同时有效缓解开发人员与需求分析人员之间的鸿沟。

3）Z-Model 明确表述了功能的特征和运行表现，为大规模制品复用和扩展提供了可行性。基于 Z-Model 的开发框架 WF 包含一套的功能单元集成规范和接口，并且已经集成 9 种基本功能单元。最常用的三种单元(通用查询、通用表单、通用树)可以满足 90%的业务系统需求。WF 采用业务流程图以及功能配置绑定的开发方法可加快开发速度，降低开发难度，从而使得非编码人员也能着手于 Web 系统开发。

4）Z-Model 遵循 MVC 架构，可提高服务器的执行效率，减轻服务器压力。Z-Model 下运行的视图是以业务窗口为单位的功能单元，其控制器（事件动作处理机制）位于客户端，可减少了对服务器的请求。另一方面 Z-Model 下的视图是功能单元组织数据的表现，加工视图的工作在前端完成，使得服务器端只需要提供最基本的数据结构访问，进一步减轻了服务器的负担。

5）基于 Z-Model 的开发可适应高度变化的用户需求。Z-Model 将数据展现，事件动作及数据持久化操作分离，有效地控制了需求变更范围，防止变更扩散。另一方面，开发人员可简便、随意地更改业务流程图结构或修改功能单元的配置与绑定，有效避免修改源代码所带来的工作量。

结束语

在基于 B/S 架构的 Web 应用开发已经成为主流的趋势下，诸多软件企业已提出各自的 Web 应用快速开发框架。然而，缺乏建模分析能力、不适应变更需求以及页面交互不够灵活等问题阻碍了这些框架的使用。针对这些问题，本文提出了 Z-Model 元模型，为在信息管理领域实现准确、经济且高效的开发提供了行之有效的方案。然而，Z-Model 本身的问题决定了它在描述非业务化的场景时，如信息门户、论坛等方面存在局限性。另一方面，Z-Model 将业务持久化操作限定在功能按钮上，在一定程度上限制了用户的交互方式。这些将是以后模型改进或演化、进而对 Web 快速开发框架实现完善和扩展的工作重点。

参 考 文 献

[1] Ted Husted. Struts in Action [M]. Manning Publications Company, 2003.

[2] C.Wall, R.Breidenbach. Spring in Action [M]. Manning Publications Company, 2007.

[3] F.Buschmann, R.Meunier, H.Rohnert, P.Sommerlad, et al. A System of Patterns: Pattern-Oriented Software Architecture [M]. John Wiley & Sons, 1996.

[4] J.Arthur,S.Azadegan. Spring framework for rapid open source J2EE Web application development: a case study. Proceedings of the Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Networks. 2005 [C]. pp 90-95.

[5] R.Johnson. J2EE Development Frameworks [J]. IEEE Computer, 2005, 38(1):107-110.

[6] Ping Zhang, R.V.Small, G.M.von Dran, et al. Websites that Satisfy Users: A Theoretical Framework for Web User Interface Design and Evaluation. Proceedings of the 32nd Hawaii International Conference on System Sciences. 1999 [C].

[7] M.Mezini, K.Lieberherr. Adaptive plug-and-play components for evolutionary software development. Proceedings of the 1998 ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages & Applications. 1998. [C].

[8] 东大金智 EPStar. http: //www.wiscom.com.cn/. [OL]

[9] 上海普元 EOS. http://www.primeton.com/. [OL].

[10] 北京金富瑞 UCML. http://www.ucml.com.cn/. [OL].

[11] 上海群萃 Extraction. http://www.extract.com.cn/. [OL].

[12] 上海锐道 Dorado. http://www.bstek.com/. [OL].

通讯作者：沈立炜

联系方式：上海市张衡路 825 号复旦大学软件楼 403 室。

（Phn）+86-021-51355343

（E-mail）shenliwei@fudan.edu.cn