Web 软件的生长式开发过程模型及其支撑工具

朱鸿

（国防科技大学计算机科学系，长沙 410073）

摘要： Web 软件多具有分布式、超媒体、自治、协作等性质，其生命周期多具有明显的动态性，其开发和维护面临着巨大的挑战。本文提出一个适合于 Web 软件开发的生长式过程模型与生命周期策略，并进一步提出了支持这一开发策略的软件工具模型－软件生长环境 MAISGE，介绍了正在以生长策略开发的针对 Web 信息系统质量管理与测试的 MAISGE－AquIS 软件生长环境元系统。

关键词：Web 软件；软件工程；软件过程模型；生命周期策略；软件环境与工具；Agent

中图分类号：TP311.5

A Growth Process Model and Its Supporting Tools for Developing Web-Based Software

Zhu Hong

(Dept of Computer Science, National University of Defence Technology,

Changsha 410073)

Abstract: Web-based application software is usually distributed, hypermedia, autonomous and cooperative. Its life-cycle is more dynamic than traditional applications. These features impose tremendous challenges to the existing theories and methods of software development. This paper proposes an evolutionary process model called growth model for developing web-based software and discusses different possible strategies in life-cycle planning for such software. It also proposes a novel type of software environment, called software growth environment, to support the growth process. A meta-system of growth environments for testing and quality management of web-based information systems is presented, which is also being developed in growth strategy.

Keywords: Web-based application software; software engineering; software process models; life-cycle strategies; software environments and tools; software agents

1 引言

因特网和 Web 技术带来了一个具有分布式、超媒体、自治、协作等性质的应用软件实现平台，刺激了许多新的计算机应用领域的发展[1]。然而，Web 软件的开发和维护面临着巨大的挑战［2］。当前 Web 软件开发实践与传统软件开发在许多方面有显著区别［3］。例如，与传统软件相比，Web 软件开发队伍规模小，开发周期短，力求尽快地把高质量的产品推向市场，而不像传统软件开发那样要求以最低的开销开发软件产品。开发过程混乱、产品质量低下是当前 Web 软件的开发普遍现象［4］，传统软件生命周期和过程模型在 Web 软件的开发中遭到了抛弃，传统的软件开发方法和工具显得软弱无力。Web 软件的开发急需方法学指导和工具支持。

除用户需求的变化之外，造成上述现象的主要外在原因有：（一）Web 软件的运行环境具有开放性和动态性，其正确运行常常依赖于万维网上具有自主性的信息资源，甚至硬件资源，而传统软件开发方法则往往假设系统所依赖的资源在系统的有效控制之下，并通过修改控制和配置管理等手段，以保障其一致性等质量要求，从而保障系统的正确和高效运行，但这些手段对于具有自主性的第三方

资源无能为力。（二）由于运行环境的开放性，Web 软件还具有用户类型复杂、差异巨大等特点。传统软件开发中行之有效的用户分析和人机交互设计以及用户培训等方式方法一部分失去了可行性。（三）Web 软件技术本身处于一个快速发展的阶段，Web 软件的运行平台、信息表示形式以及各个构件的开发方法和实现技术往往随着 Web 软件技术的发展而采用最新技术，从而在同一个系统中也呈现出多样性，而传统软件设计时，往往在系统范围内采用统一的标准的信息表示形式，针对选定的运行平台，使用一种开发方法和实现技术，以简化其实现，保障一致性。因此，传统的以支持一种方法和技术为主的软件开发工具对多种开发方法和实现技术的混合失去了效用。

此外，Web 软件的内在性质也决定了它的生命周期的特点。许多基于 Web 的信息系统，如电子商务, e-learning, e-science 等等，属于 Lehman 软件分类体系中的 E-类［5］，即“它所解决的问题和实现的应用属于现实世界的领域，其正确性是由程序在一定操作条件下的行为所决定的，并制约于人们的经历、认识、决策水平、以及种种由此而蕴含的人为因素”。这类软件的演变过程通常满足 Lehman 的七条软件演变规律：变化的持续性、复杂度的递增性、演化过程的自我规范性、机构组织的稳定性、变化规模的均衡性、功能的持续增长性［5］。因此，Web 软件的生命周期多具有明显的动态性。

本文从适于 Web 软件开发的过程模型和支撑工具的结构两个方面，探讨 Web 软件工程的新途径。

2　软件生命周期的生长模型

我们在文［2］中提出了软件过程的生长模型，见图 1。该模型把软件生命周期分为三个阶段：萌芽期、成长期、和衰亡期。

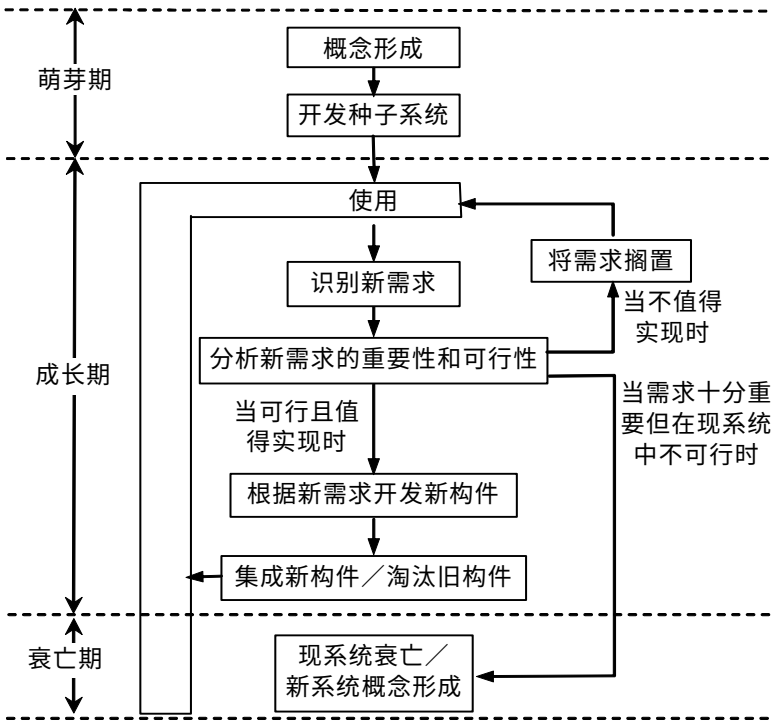

图 1、软件过程的生长模型

在这个模型中，萌芽期包括了软件系统从概念形成到第一个可使用的系统开发完成的过程。这个过程可以通过各种软件开发过程模型来完成。成长期包括了

软件在使用过程中不断发现新需求、不断开发并集成新构件、并淘汰旧构件的成长过程。其主要活动包括新需求的发现与识别、新需求的重要性分析、新需求与现系统的相容性分析（即分析新需求在现系统中的实现是否能行及其开销与收益的比例等问题）、新构件的开发及其与现系统的集成、无用构件的淘汰、过时构件的更新，等等。值得指出的是，多个需求可能以并行的形式被识别、分析、实现和集成到现系统中去。在这种软件并发成长的情况下，还必须分析新需求之间的相容性。衰亡期是一个软件系统在不能满足新需求的情况下，被淘汰而停止使用的过程。

软件过程模型具有两大作用：一是对软件生命过程的认识理论，在一定的抽象层次上客观地刻划软件生命过程的规律；二是指导软件开发，提供软件工具设计的依据。软件生长模型不仅刻划了 E－类型软件演变的过程，反映了用户需求变化所引起的软件演化过程，它也适用与其它类型的软件。例如，对于 S-类型的软件，其程序要求满足预先指定的规约，其正确性是程序与规约之间的绝对一致性。因此，S-类型的软件的唯一可能的‘ 新需求 ’是改正程序中的错误，软件演变的过程也就是纠正错误的维护过程。

软件过程的生长模型对软件开发过程的指导作用集中表现为它提供了针对不同类型软件的特征，进行生命周期策略性规划和分析的手段。

例如，对于 S－类型软件，由于其规约的稳定性，软件的变化即改正错误，变化越多，开销就越大。因此，规划整个软件生命周期的策略应该以尽量减少改变为重点。从成长模型来看，达到这一目的的唯一途径是所开发的种子系统应该尽量满足规约。也就是说，对于 S－类型软件，较好的生命周期策略是着力于种子系统的开发，力求减少软件的变更。我们称这一策略为静止策略。

而对于 E－类型软件而言，受用户使用计算机的经历的变化和外在条件的变化等因素的影响，其用户需求具有变化的持续性等规律，因此，为满足用户的新需求而产生的软件演变是不可避免的。适用于 S－类型软件的静止策略不再适用于 E－类型。而适用于 E－类型的较好的生命周期策略则是：（1）在种子系统的开发时，不必追求满足所有的需求，而应该区分需求的轻重缓急，首先实现关键的核心需求，并在系统结构设计中，充分估计今后对系统进行扩充、修改等变更的要求。（2）在系统使用过程中，充分听取用户的反馈意见，从而识别用户在使用中产生的新需求，并根据其轻重缓急，选择核心需求加以实现，而不必追求满足用户所有的需求。每一次更新都应该看作是其漫长的生命周期中的一小步，因此，每一次更新都必须为今后的更新作好准备。也就是说，对于 E－类型软件，较好的生命周期策略是时刻准备着变更，不必追求完美，从而使变更的开销降低，软件的生命周期延长。我们认为，这一策略也就是 Web 软件开发所应该采取的策略。事实上，如果把整个万维网看作是一个大信息系统，其演变过程充分反映了这一生命周期的特点［2］。下文中我们称这一策略为生长策略。

文［2］中还分析了在当前万维网时代有意识地采取生长策略的优越性，在此不一一细述。

3　支持生长模型的 Web 软件开发工具

在生长策略下开发 Web 软件，对软件工具的功能和结构都提出了新的需求。其中主要功能需求有：（1）支持新需求的识别，使用户的反馈和环境的变化能够更准确、更及时地反映给开发人员；（2）支持系统对新需求与现有系统的关系的分析，例如对实现一个新需求可能产生的影响的分析；（3）支持的各种变更操作，尤其是新构件与现有系统的集成，以及过时构件与系统的脱离。除了这些功能需

求之外，Web 软件开发还要求软件工具能够支持多样性和组合性， 即：(4) 支持多种开发方法和实现技术的有效混合使用；(5) 支持多种信息表示标准的混合使用；(6) 支持较方便地对软件工具进行扩展，使之在新标准、新技术和新方法出现时，支持其使用。由上述需求可知，软件集成机制在 Web 软件工具中起着十分重要的作用。

分析当前各种软件集成技术，我们可以区分三种软件集成机制：（1）静态集成机制通过将各个软件构件的源代码在一起编译和连接形成一个完整的系统；（2）动态集成机制将目标代码通过连接形成完整的系统，并在系统运行开始之前完成集成；（3）活态集成机制在各个集成成分都在运行状态下通过通讯和交互形成一个协调工作的系统。当前 Web 软件技术正在朝着愈来愈多地采用活态集成机制的方向发展［2］。例如，Web Service［6］和多 Agent 技术［7］都是活态集成机制的典型例子。针对这类系统的软件环境也必须采用活态集成机制。Agent 技术在 Web 软件中正在得到越来越广泛的应用。多 Agent 系统所提供的方便灵活的活态集成机制十分适合万维网上资源的开放性、动态性和自治性。而且，万维网服务器上运行的软件在 Wooldridge 和 Jennings ［8］的弱 Agent 概念下均可视为 Agent。我们认为，对以 Agent 技术为主体的 Web 软件的开发、维护和管理也必须应用 Agent 技术。而以其它集成机制构造的 Web 软件，Agent 技术也提供了一种实现满足上述需求的软件工具环境的较好的途径。此外，Agent 技术还具有其它诸多优越性，在此不一一细述。

图 2 给出了一个基于 Agent 技术的具有如上功能的信息系统支撑工具环境的概念模型 MAISGE。

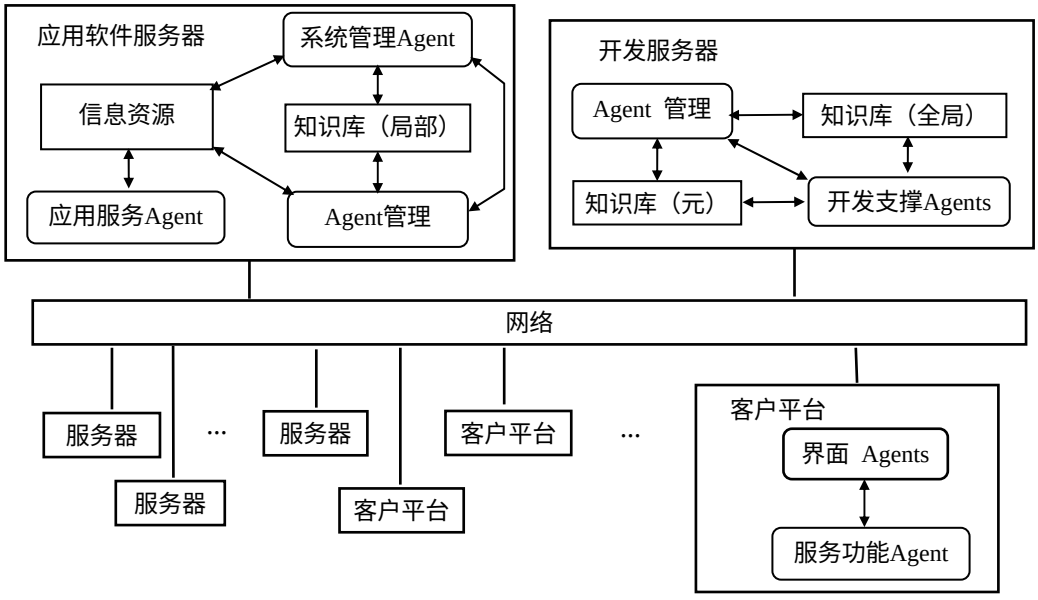

图 2、多 Agent 软件生长环境 MAISGE

MAISGE 模型基于多服务器和多客户平台的网络计算环境，是一个针对 E－类型分布式信息系统生长式开发的多 Agent 软件系统。应用系统中的软件构件以 Agents 的形式出现，软件构件的加入和删除成为 Agent 的动态加盟和退出。支持软件开发活动的各种工具也以 Agent 的形式出现，并与应用系统中的作为软件构件的 Agents 同时共存，在系统管理 Agents 的协调下，相互协助，共同完成应用

系统边开发边使用、稳步成长的过程。因此，整个系统中包括两类 Agents：应用服务 Agents 和开发服务 Agents。

应用服务 Agents 提供对用户的直接的服务并完成系统的各种功能。应用服务 Agent 又可以进一步分为两类，界面 Agents 和服务功能 Agents，前者提供友善的人机交互界面与工具，并担负部分接受用户反馈的责任。后者完成各种系统功能。

开发服务 Agents 提供对软件开发人员的各种开发活动的支持，它们可进一步分为系统管理、开发支持和 Agent 管理三类。系统管理 Agents 负责对应用系统的监视、管理和日常维护，并通过与应用服务 Agents 中的部分界面 Agents 的通信，获得用户的反馈。开发支持 Agents 提供软件开发工具以及对软件开发人员的各种开发活动的服务，除各种较传统的开发工具外，它们还通过对系统知识的采集、维护和使用，帮助或独立完成各种修改影响分析、一致性和相容性分析等工作。Agent 管理 Agents 负责对系统中的各种 Agents 进行管理，其主要职责有三：(1) 注册：在 Agent 加入系统时，对 Agent 的功能、职责、能力、运行环境需求等信息进行注册登记，在 Agent 退出系统时，对 Agent 进行注销；(2) 调度：当开发、维护和使用任务出现时，将任务分配给适当的 Agent 去完成；(3) 记录：并对任务的状态、进展过程、结果以及各个 Agent 的表现进行记录和分析，为将来更有效地进行任务调度积累信息。

MAISGE 模型不同于现有软件开发环境，它具有如下特征：(1) 开发工具与信息系统共存于同一环境之中，从而使生长策略下信息系统在漫长的开发与使用同步进行的生命周期得到全面的支持；(2) 为了支持信息系统中日益增加的智能性服务以及信息系统开发本身所必须具备的智能性，采用分布式数据库和知识库，全面记录并使用有关系统及其软件开发的知识与信息，如 A) 有关信息系统本身的结构、内容、版本、配置、演化过程等信息；B) 系统中各个 Agent 的能力、需求和业绩，以及相互之间的关系；C) 各种软件开发任务和开发过程的知识，如开发任务如何分解为子任务、开发任务之间的逻辑和时序关系，等等。这些通常散布于多个不同种类的软件开发工具的知识与信息被集成于同一系统中，可以提供信息系统稳步成长的良好环境。(3) 采用便于系统扩展的软件体系结构，即多 Agent 系统，使环境系统与信息系统本身都具有良好的可生长性。由于这些特点，我们称该模型为信息系统生长环境。

4 MAISGE－AquIS：Web 软件生长环境实例

基于 MAISGE 模型，我们正在以生长策略开发 MAISGE－AquIS 系统，这是一个针对 Web 软件特点的软件生长环境。该系统当前的主要开发重点是 Web 软件的质量管理，其第一阶段的结构如图 3 所示。该系统是一个通用 Web 软件生长环境元系统，其中只包含一组开发服务 Agent 和信息系统知识库框架。给定一个 Web 信息系统，该元系统可以通过一组知识抽取 Agent，自动抽取其结构信息并存放于系统知识库中，而从嵌入到生长环境中，成为一个特定系统的生长环境。在对信息系统的开发、测试和维护中产生的元信息也被存放在知识库中，为自动进行回归测试提供必要的信息。它还对信息系统本身及其环境的变化进行监控，当变化发生时，分析变化的影响，自动产生由此而必须进行的测试工作任务，并由 Agent 管理与调度 Agent 产生测试计划、分配测试工作，由测试助手 Agent 协助测试人员完成质量保障工作。

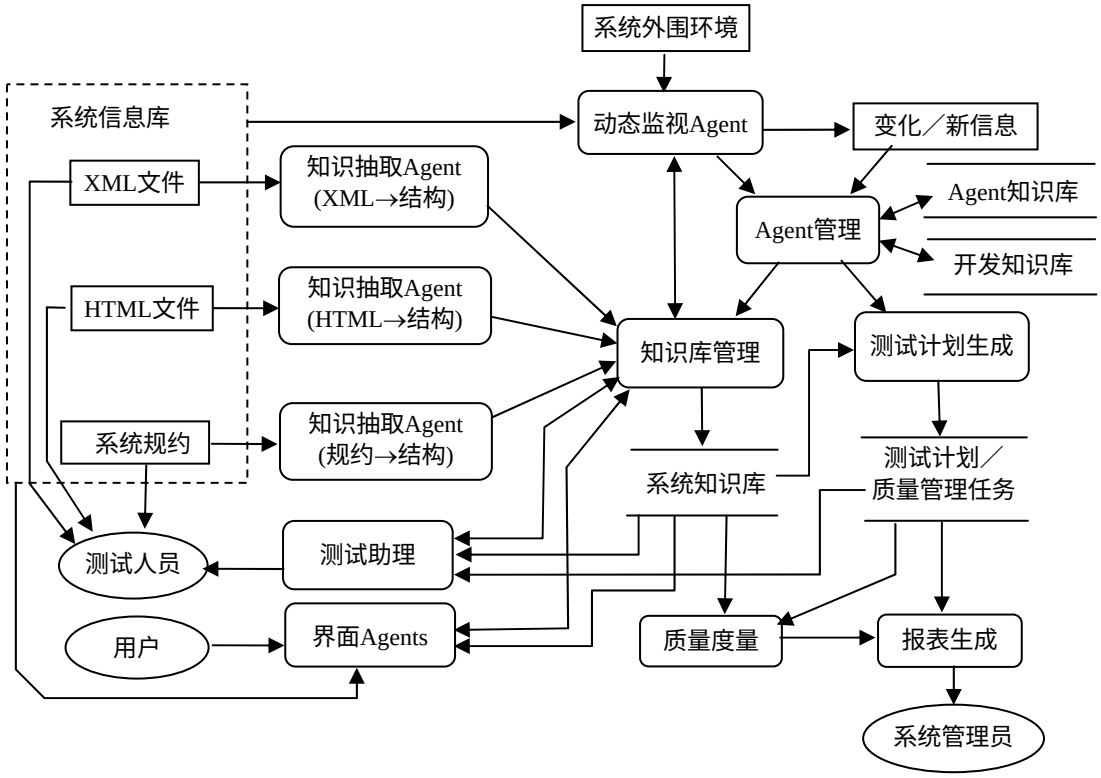

图 3、MAISGE－AquIS 结构示意图

MAISGE－AquIS 的实现分两个抽象层次，在较低的抽象层次上实现了 Agent 通信基础设施，它由一组支持移动 Agent 的通信原语组成，实现 Agent 之间的信息传递、加密与广播等功能［9］。在较高的抽象层次上，实现了一个软件测试的 Ontology 作为 Agent 通信语言［10］，以及一组管理基本 Agent 和一组应用于质量管理和测试的开发服务 Agents，如 Web 的一些特定质量度量 Agents［11, 12］、知识自动抽取 Agents、测试计划自动生成 Agents、系统环境监视 Agents，等等。还有一些 Agents 正在开发与调试之中。

5 结语

本文提出了刻划软件生命周期的生长模型。该模型与现有的瀑布模型、螺旋模型、RAD 模型等传统软件过程模型相比，较好地反映了软件维护、进化等变更活动在软件生命周期中的作用，由此引出了软件生命周期规划策略的新概念。文中分析了 Web 软件的特点，指出了 Web 软件具有 Lehman 的 E－类型软件特征，并针对其特征提出了适合于 Web 软件生命周期的生长策略。该模型与目前正在形成的极端程序设计(Extreme Programming)［13］、轻快软件开发(Agile Software Development)［14］等软件工程新思维相比，在应付 Web 时代软件开发所面临的困境的策略上具有许多共同的出发点，但生长模型明确地提出了软件生命周期不同策略的存在性和可规划性，提出了生命周期策略应该与软件的 Lehman 类型相适应。生长模型更强调了继承传统软件过程模型的基本原理，而不是完全抛弃现有软件生命周期和过程模型的理论。事实上，各种开发过程模型（包括传统的和上述新兴的途经）都可以用于生长模型中种子系统和新成分的开发。

为了减少 Web 软件生长过程的开发和维护开销，本文进一步提出了支持生长策略的基于 Agent 技术的软件生长环境 MAISGE 模型，介绍了正在以生长策略

开发的 Web 软件生长环境元系统 MAISGE－AquIS 的结构以及实现现状。软件生长环境模型与现有的各种软件开发环境和运行时刻的支撑环境都具有本质区别，克服了现有软件环境对生长式生命周期支持不力的缺点。

Web 软件是软件工程研究的一个新课题，还有许多问题有待深入研究。

致谢：英国牛津布偌克斯大学计算机系张燕龙、火清宁和 Sue Greenwood 等人参与了本课题的部分研究，在此致谢。

参考文献

[1] Crowder, R., Wills, G., Heath, I., and Hall, W. Hypermedia Information Management: a New Paradigm, in Proceedings of 3rd International Conference on Managing Innovation in Manufacture, University of Nottingham, 6-8 July 1998, pp329-334.
[2] Zhu, H., Greenwood, S., Huo, Q. and Zhang, Y., Towards agent-oriented quality management of information systems, in Workshop Notes of 2nd International Bi-Conference Workshop on Agent-Oriented Information Systems at AAAI'2000, Austin, USA, July 30, 2000, pp57-64.
[3] Reifer, D., Ten Deadly Risks in Internet and Intranet Software Development, IEEE Software, Vol. 19, No. 2, Mar/Apr 2002, pp12-14.
[4] Taylor, M. J., McWilliam, J., Forsyth, H. and Wade, S., Methodologies and Website Development: A Survey of Practice, Information and Software Technology, Vol. 44, 2002, pp381-391.
[5] Lehman，M. M., Laws of Software Evolution Revisited, Proc. Of EWSPT96, Nancy, Oct. 1996.
[6] Gottschalk, K., Graham, S., Kreger, H., and Snell, J., Introduction to Web services architecture, IBM Systems Journal, Vol. 41, No. 2, 2002, pp170~177.
[7] Wooldridge, M., An introduction to Multi-Agent Systems, Wiley, 2002.
[8] Wooldridge M. J. and Jennings, N. R. , Agent theories, architectures, and languages: a survey in Intelligent Agents: Theories, Architectures, and Languages, LNAI 890, Springer-Verlag, 1995, pp1-32.
[9] Huo, Q.，Zhu, H., Message Communication Model for Mobile Agents, Proc. of CACSUK'2000.
[10] Huo, Q., Zhu, H. and Greenwood, S., Using Ontology in Agent-based Web Testing, Proc. of ICIIT'2002, 22~25 Sept., 2002, Beijing, China. (In press)
[11] Zhang, Y., Zhang, L., Huo, Q., Zhu, H., Greenwood, S., Structure and page complexity metrics for Web applications, Proc. of 4th WWW Workshop on Web Engineering at WWW'2001, Hong Kong, May 1, 2001, pp72-81.
[12] Zhang, Y., Huo, Q., Zhang, L., Zhu, H., Greenwood, S., Measuring the timeliness of websites, Proceedings of the Fourth European Conference on Software Measurement and ICT Control in Co-operation with DASMA, Heidelberg, Germany, 8-11 May 2001, pp139-147.
[13] Beck, K., Extrem Programming Explained, Addison Wesley, 2000.
[14] Cockburn, A., Agile Software Development, Addison Wesley, 2002.