

Open Source Portal Software as a Catalyst for Sound Software Engineering Practices at UTEP

Prepared by:
Kenneth R. Pierce
Kenneth@pierce.name

September 18, 2005

TABLE OF CONTENTS

TABLE OF CONTENTS	2
ABSTRACT	3
INTRODUCTION	3
Capability Maturity Model Integration	4
UTEP's Software Development Paradigm Shift	4
UTEP's Technology Platform	5
Impact of the Software Development Decision on UTEP's Capability Levels	5
AFFECTS ON THE ENGINEERING PROCESS AREA	6
Requirements Development	6
Technical Solution	7
Product Integration	9
Verification	10
Validation	10
AFFECTS ON THE SUPPORT PROCESS AREA	11
Configuration Management	11
Process and Product Quality Assurance	12
CONCLUSION	13
REFERENCES	13

ABSTRACT

As web sites become more and more the primary face of universities, it is essential that they provide a consistent, stable view to the user. Unlike many corporations with large web sites, universities have historically had difficulty in gaining the cooperation needed across their campuses to provide a uniform user experience. Colleges, departments, and programs are usually in charge of their own web site design and content, therefore consistency, usability, and reuse are of little consideration. Were there a catalyst to entice a university to consolidate its web infrastructure, it would most likely fall upon the university's Information Technology (IT) department to implement and support it. Given this responsibility, it is imperative that the IT department choose an appropriate foundation for the web infrastructure, and employ proven software development processes. One university, The University of Texas at El Paso (UTEP), recently made this major paradigm shift in preparation of its accreditation review from the Southern Association of Colleges and Schools (SACS). It now successfully hosts nearly 1000 integrated web sites in an infrastructure that is built upon an open source portal technology, and this technology infrastructure has assisted UTEP's IT department in formalizing and implementing key software engineering practices for their web infrastructure. UTEP's IT department has seen improvements in requirements gathering, coding standards, design templates, code reuse, testing, deployment, and project management. A thorough investigation of the deliberate technology decisions made by UTEP, and the process changes they made following these decisions, reveal dramatic increases in the UTEP development team's capability levels as measured by the Software Capability Maturity Model Integration (CMMI). This document recommends that all universities implement technology and processes that mimic those of the University of Texas at El Paso in order to establish a concise road to process maturity.

INTRODUCTION

Over the past two decades, the role software has played in society has grown substantially, and continues to be an ever-growing influence on the way that humans and entities interact with one another. Increased reliance on software brings increased requirements for quality and reliability, and increase expectations on functionality and the speed in which that functionality is delivered. As such, it is important that those involved in the development of software mature in their ability to meet these growing demands, which has led to the establishment of a software engineering discipline and various related methodologies. According to a Gartner Report, "Whereas IS organizations formerly minimized the significance of development engineering costs as non-recurring, the current critical nature of software in a product or service offering has elevated the importance of the development process in return-on-investment considerations. Thus, it is essential to maximize the use of any available tools or models that measure and improve the efficiency and effectiveness of software development." (Gartner, 2005)

Software engineering is the profession that creates and maintains software applications by applying technologies and practices from computer science, project management, engineering, application domains, and other fields. (WikiPedia, 2005) The software engineering profession has grown significantly, and its growth continues as software becomes more complicated to develop and support. Software engineers focus on applying systematic, disciplined, and quantifiable approaches to the development, operation, and maintenance of software.(Institute, 2002)

Capability Maturity Model Integration

One of the key entities involved with the leadership and evolution of the software engineering discipline is Software Engineering Institute (SEI) at Carnegie-Mellon University. The SEI recognized the need for systematic processes in software engineering, and developed the Capability Maturity Model (CMM) to address this need. Today, the next version of the Capability Maturity Model, CMMI (Capability Maturity Model Integration) which was built upon the original CMM is in use all around the globe as the foundation for processes in software engineering. According to the SEI, “the purpose of CMM Integration is to provide guidance for improving your organization’s processes and your ability to manage the development, acquisition, and maintenance of products or services. CMM Integration places proven approaches into a structure that helps your organization appraise its organizational maturity or process area capabilities, establish priorities for improvement, and implement these improvements.”

Within the CMMI, there are categories of process areas. The process areas are grouped into four categories:

- Process Management
- Project Management
- Engineering
- Support

(Institute, 2002)

Capability levels focus on growing the organization’s ability to perform, control, and improve its performance in a process area. Capability levels enable you to track, evaluate, and demonstrate your organization’s progress as you improve processes associated with a process area. Capability levels build on each other, providing a recommended order for approaching process improvement. (Institute, 2002) Table 1 below taken from the CMMI document shows the capability levels within the CMMI Continuous model.

0.	Incomplete
1.	Performed
2.	Managed
3.	Defined
4.	Quantitatively Managed
5.	Optimizing

Table 1 - CMMI Capability Levels

For an organization, the ultimate goal is the achievement of Level 5 in all four of the process areas. Such an accomplishment places a software development organization into category that few organizations will ever reach, as achieving such a level requires a significant effort on behalf of the organization.

UTEP’s Software Development Paradigm Shift

Less than three years ago, UTEP’s overall web / intranet implementation and strategy was non-existent. All managed and supporting separately, UTEP’s sites had no linkage or commonality between them. Any attempt in developing functionality for the web capable of reuse across the university web sites would be futile. Today, UTEP has a significant implementation of a centrally managed web platform (based on DotNetNuke) that has completely renovated its web presence

from a user standpoint, but has also contributed to solid software development principles and practices that have resulted in significantly improved software products produced by the Information Technology department.

Although UTEP has not followed the staged CMMI representation, it has followed closely with the CMMI continuous representation. The development team made several key decisions that would determine their future capability to produce quality software. The core of these decisions involved a standard development environment, development language, and coding standards. These decisions led to the selection of a new foundation platform for development of software applications, and resulted in significant advancements in its CMM level.

Although this document will reference the technology chosen by UTEP, any software development organization that follows a similar methodology to UTEP should see the same gains in their maturity level as outlined by the CMMI model.

UTEP's Technology Platform

The technology that UTEP elected to use as a standard for application development was Microsoft's .NET platform. This platform includes the Windows 2003 operating system, along with the .NET framework, and Visual Studio .NET for the Integrated Development Environment (IDE). All of UTEP's code is stored in a Visual SourceSafe repository.

Although the standardization on a particular development environment, i.e. .NET, was a significant step for UTEP software development team, it was only one of several decisions made. In addition to the use of the Microsoft .NET framework, UTEP also standardized on the use of an open-source portal environment built with (and for) the .NET environment. The open-source portal environment is "DotNetNuke". According to its website, DotNetNuke is an open-source content management system ideal for creating and deploying projects such as commercial websites, corporate intranets and extranets, and online publishing portals. (DotNetNuke, 2005) The use of open source for the framework was critical to UTEP, as it was going to be necessary for them to make one core change to the open source code in order to integrate it with their single sign-on infrastructure. In addition, UTEP would be able to take advantage of the benefits of open source software, such as lower acquisition cost, high quality and stability, and the ability to contribute to the effort and have their contributions managed by the owner of the open source initiative. (Surran, Sep2003)

Because of these key decisions, the development team was poised to make significant strides in its quest for higher CMM levels of maturity. The remainder of this document outlines the details regarding the changes UTEP made with the implementation of .NET and DNN, and the impact it had on the team's CMMI capability levels.

Impact of the Software Development Decision on UTEP's Capability Levels

Although the key decisions made by the UTEP development team do not necessarily seem like decisions any different from other development teams, it is important to realize the advantage that such decisions can have towards an organization's quest for higher capability levels. Although there may be some process advantages in the Process Management and Project Management key process areas, the development team has experienced a majority of its improvement in the areas of Engineering and Support. Table 2 below shows the specific areas (sub practices) within the Engineering and Support process areas that have experienced improvement because of the decisions made by the development team.

Process Area	Positively Impacted
Engineering	

Requirements Development	<input checked="" type="checkbox"/>
Requirements Management	<input type="checkbox"/>
Technical Solution	<input checked="" type="checkbox"/>
Product Integration	<input checked="" type="checkbox"/>
Verification	<input checked="" type="checkbox"/>
Validation	<input checked="" type="checkbox"/>
Support	
Configuration Management	<input checked="" type="checkbox"/>
Process and Product Quality Assurance	<input checked="" type="checkbox"/>
Measurement and Analysis	<input type="checkbox"/>
Organizational Environment for Integration	<input type="checkbox"/>
Decision Analysis and Resolution	<input type="checkbox"/>
Causal Analysis and Resolution	<input type="checkbox"/>

Table 2 - Engineering and Support Process Areas Affected

The Engineering and Support process areas are described in the sections “Affects on the Engineering process area” and “affects on the Support process area” below, along with the analysis of the improvements recognized by the development team.

AFFECTS ON THE ENGINEERING PROCESS AREA

Engineering process areas cover the development and maintenance activities that are shared across engineering disciplines (e.g., systems engineering and software engineering). The six process areas in the Engineering process area category have inherent interrelationships. These interrelationships stem from applying a product development process rather than discipline-specific processes such as software engineering or systems engineering.

The Engineering process areas of CMMI are as follows:

- · Requirements Development
- · Requirements Management
- · Technical Solution
- · Product Integration
- · Verification
- · Validation

(Institute, 2002)

Requirements Development

The purpose of Requirements Development is to produce and analyze customer, product, and product-component requirements. (Institute, 2002) Being one of the most critical stages of the development process, the strategy of implementing .NET/DNN allowed the development team to establish specific requirement documents and templates related to the development of the software products. The development team has found that the requirements gathering process has been significantly more effective due to increased customer understanding of the targeted DNN environment. Since DNN is the sole target environment for development, combined with the fact that nearly all UTEP web sites are hosted in this environment, the user understanding is much higher than if there were several target environments of which they had no familiarity. Developers can now work with users more effectively by demonstrating standard functionality

and applying it to the requirements gathered. In fact, the development team has had customers provide storyboard templates in DNN format to demonstrate their requirements.

The requirements documents and templates have undergone several revisions based upon lessons learned from the first few development projects. These templates are used by all of the developers on the team, and there is now a high level of consistency within the requirements gathering process. Prior to implementation of DNN, each developer used their own template for gathering requirements, as they targeted multiple disparate environments.

An example of this consistency would be in the area of module settings. The DNN infrastructure provides for a standard method for accessing the settings and configurations of a module. The requirements documents generated by the development team have a template for what module-level configurations parameters can be changed. See *Figure 1 - DNN Module Settings* below for a visual representation of the settings requirements.

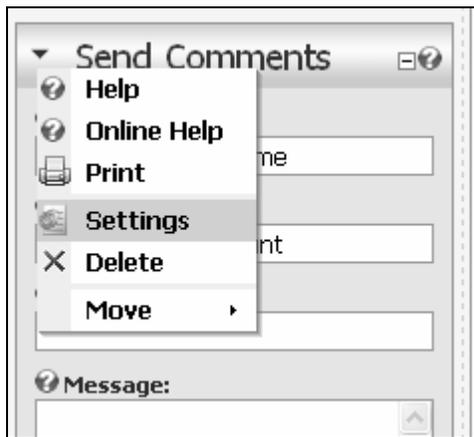


Figure 1 - DNN Module Settings

In addition to the templates and standards that follow along with the DNN implementation are standards and guidelines for security. UTEP uses a single sign-on infrastructure that was built in-house. All web applications that require user authentication use this single sign-on infrastructure. For applications that need further security, such as authorization, the same set of credentials are used for querying the Lightweight Directory Access Protocol (LDAP) implementation present at UTEP. By ensuring that all web applications use this infrastructure, the UTEP development team has eliminated much of the need to outline security design in its requirements; only details beyond authentication need be documented.

The DNN infrastructure has assisted greatly in the process for eliciting needs (which is one of the key practices in the Requirements Development area) by providing a demonstration and brainstorming framework, as well as a prototyping environment. Users can visualize their requirements more easily, and thus the development team can design them without the confusion that once existed.

Technical Solution

The purpose of Technical Solution is to design, develop, and implement solutions to requirements. Solutions, designs, and implementations encompass products, product components, and product-related life cycle processes either singly or in combinations as appropriate. (Institute, 2002)

Advances in the processes related to developing the technical solution have been made through the streamlining of technical alternatives. In this process area, development teams frequently

look at the feasibility of using commercial off-the-shelf (COTS) products or performing in house development. This feasibility study is still performed by the UTEP development team; however, the decision has been reduced to 100% in-house development, or purchasing pre-built modules (with source code) and using these modules as the foundation for designing a solution. One of the benefits that DNN has is its long list of 3rd party controls that can be easily and seamlessly placed within the environment. These controls may perform specific tasks (such as displaying local weather) or they may be generic in nature (such as providing data grid or calendar functionality). It is the latter of these two types of controls that are looked at by the development team. In addition, a large number of vendors offer specific controls that are not built specifically for the DNN environment, but only for the Microsoft .NET environment. In a situation where the requirements warrant such functionality, the development team “wraps” the control in a DNN container object, and then provides access to it as if it was a native part of DNN. By taking this approach, UTEP’s development team can keep the solution technically consistent.

UTEP can cite several examples of this process, one of which is online classified advertisements. The Student Government Association (SGA), regarding the development of online classified advertisements for students, approached the UTEP development team. Since the SGA web site resides in DNN, the development team performed an analysis of available DNN modules that could fulfill this requirement, and found one module for purchase from a vendor of DNN modules. The analysis showed that the module could perform nearly 95% of the requested requirements out of the box, and would only need minor code changes to integrate into the UTEP DNN environment. The key change was enabling the module to recognize the single-sign on credentials of the user and not allow anonymous posts of items. The result of this activity was a product ready for test within 3 days of finalizing the requirements, along with a non-labor cost of \$20 for the module and its code.

Although the previous example represents an example of a technical solution that used a component with a narrow focus, there is one other example worth citing which is was development of a class schedule module for departmental web sites. The basic requirement provided to the development team was to provide students and faculty with real-time class schedules during registration periods, including the number of available seats in each class. Again, since the targeted environment was DNN, the development team’s technical solution would be based upon that environment. In this instance, there was no COTS control that would fulfill that requirement, however, the development team was able to use a more generic control to provide the grid functionality for display, sorting and filtering of the information. Use of this control reduced the development time significantly. This control is depicted in Figure 2 - Course Availability Module.

Course Availability

Maymester

Marketing and Management Search department: Course:

TITLE	CRN	SUBJ	COURSE NO	SEATS AVAIL.	TIME, DATE & LOCATION	INSTRUCTOR NAME	SCHEDULE TYPE	TEACHING METHOD	DEPT. RESTRICTION
Principles of Marketing	34350	MKT	3300	0	TBA		Lecture		Department
Principles of Marketing	33540	MKT	3300	7	MTWRF 0800-0100 pm BUSN 319	Donald Michie	Lecture		
Buyer Behavior	34351	MKT	3302	0	TBA		Lecture		Department
Intro-Mgmt/Organizational Beha	33539	MGMT	3303	10	MTWRF 0130-0630 pm BUSN 318	Sigrid Khorram	Lecture		
Global Business Environment	33538	BUSN	3304	0	MTWRF 0130-0630 pm BUSN	J. Foster	Lecture		

Figure 2 - Course Availability Module

In addition to streamlining the make versus buy decision, the decision to develop in the DNN environment by default provided the development team a detailed technical architecture for its development. All development for DNN is accomplished at a module level, with inheritance from DNN provided object classes, which emphasizes component-based development. Products produced by the development team are modular in design, and sit within the DNN framework.

The technical solution area also houses the sub-practices related to coding and architectural standards within engineering. UTEP’s development team created a set of development and coding standards based on this single environment. These standards address language, naming conventions for variables, formatting of code and comments, as well as the structures within the code. These standards are very detailed and reflect best practices they have established with the environment.

UTEP is also seeing a significant amount of code and module reuse in its development team. As mentioned earlier, reuse of components is one of the first considerations of the development team. In many instances, reuse manifests itself through the object-oriented concept of inheritance.

Product Integration

The purpose of Product Integration is to assemble the product from the product components, ensure that the product, as integrated, functions properly, and deliver the product. (Institute, 2002)

In terms of product integration, the architecture provided by DNN has been used as the standard for interfacing between objects and data repositories. The architectural design of DNN is very sophisticated, and follows nearly all best practices outlined by Microsoft. This includes n-tier development architecture.

UTEP has built several modules that reside in the DNN infrastructure for integration with its other systems. Examples of these modules include integration with SCT Banner, its student information system, which is hosted on an Oracle Database platform. All development projects that require access to Banner are required to utilize the UTEP-built data access object layer. This eliminates variability in the design, and ensures a smooth integration with each system.

This same methodology has been applied to access of UTEP's smart card system, police and parking system, and data warehouse.

Verification

The purpose of Verification is to ensure that selected work products meet their specified requirements. (Institute, 2002)

Before UTEP made their decision to standardize on .NET / DNN, there were a myriad of technologies used for development. This variation in technologies made it very difficult for the organization to establish standard software practices such as peer code reviews and standard testing scenarios.

Peer reviews are an important part of verification and are a proven mechanism for effective defect removal. An important corollary is to develop a better understanding of the work products and the processes that produced them so defects can be prevented and process-improvement opportunities can be identified. (Institute, 2002) UTEP now performs peer reviews of code in an efficient manner. Verification by a peer as to whether the developer followed the design and coding guidelines is a very simple task, as all developers in the development team follow the guidelines.

In regards to verification of performance and functionality, UTEP hosts a test environment where all new modules must be tested prior to entering production. These tests include user verification that the functionality requirements were met, and the development team ensures a proper integration with the DNN environment. In addition, the test environment is a mirror of production, and is used to ensure that performance and reliability requirements of the module are satisfactory and will not affect the hosted environment.

One of the other areas where UTEP plays a large role is in performance testing. As the largest implementation of DNN in the world, the UTEP development staff is always concerned with the performance of its environment, and DNN itself. UTEP performs load and performance testing on its environment with each new release of DNN to ensure that an implemented version does not produce undesirable performance on its web sites. If performance problems are found, they are isolated and reported back to the DNN development team. In some cases, UTEP has provided the DNN development team with required changes to correct the performance problems. This sort of collaboration and problem solving is available to UTEP as DNN is open-source software.

Validation

The purpose of Validation is to demonstrate that a product or product component fulfills its intended use when placed in its intended environment. (Institute, 2002)

The UTEP development team has setup an application environment that allows them to perform the necessary product validation. Previously, the overall environment in which developers developed may have consisted of their own workstation as a development and test platform, and the production environment. Since there was no standard for development environments, creating a separate development, test, and production environment would not only be cost prohibitive, but would have added increase overhead burden on the developers by way of support and maintenance of these environments.

Today, UTEP has a development environment that is regularly refreshed and updated (as necessary) from the production environment. Product development may occur in the development environment, or on the workstation of the developer. However, all work products must be installed and tested in the test environment, which mimics the current production

environment,. This environment includes the load balancing capabilities of the production environment. The setup and maintenance of these environments could not have occurred had the development team not standardized on the platform and processes.

AFFECTS ON THE SUPPORT PROCESS AREA

Support process areas cover the activities that support product development and maintenance. The Support process areas address processes that are used in the context of performing other processes. In general, the Support process areas address processes that are targeted towards the project, and may address processes that apply more generally to the organization. For example, Process and Product Quality Assurance can be used with all the process areas to provide an objective evaluation of the processes and work products described in all of the process areas.

The Support process areas of CMMI are as follows:

- Configuration Management
- Process and Product Quality Assurance
- Measurement and Analysis
- Organizational Environment for Integration
- Decision Analysis and Resolution
- Causal Analysis and Resolution

(Institute, 2002)

Configuration Management

The purpose of Configuration Management is to establish and maintain the integrity of work products using configuration identification, configuration control, configuration status accounting, and configuration audits. (Institute, 2002)

In terms of configuration management, the key CMMI sub practices that are affected are those that revolve around the versioning of the components used by the development team. In other words, even though configuration management covers control and versioning of customer requirements, it also covers configuration of elements used by developers. Areas that this covers for the developers are versions of:

- Windows Operating System
- The .NET Framework
- Visual Studio Environment
- DNN Release

Another aspect integrated with configuration management is code versions. UTEP uses Microsoft's Visual SourceSafe as its code repository. All code is placed into the repository, and the production environment is produced from the code in the repository. Developers must place their production code into the repository as part of the implementation and configuration management processes in place at UTEP.

In terms of defect tracking, the UTEP development team relies on two systems for management of defects: the DNN bug tracking system and the UTEP Service Desk. All bugs generated by the development team's customers are logged into the UTEP Service Desk, which is used by UTEP's help desk. If the problem is not with the core of DNN, the UTEP development team manages the correction of the bugs through Service Desk and their own development process. If the problem is determined to be within the DNN core, the bug is entered into the web-based

bug tracking system provided by the DNN development team, and referenced in the Service Desk call. This provides the development team with correlation between their customer defects and those that are passed on to the DNN development team.

OSS (Open Source Software) development emphasizes the maintainability of the software released. Making software available on the Internet allows developers around the world to contribute code, adding new functionality (parallel development), improving the present one, and submitting bug fixes to the current release (parallel debugging). (Sarnoladas *et al.*, Oct2004)

Choosing open source software is a big step for organizations, and process such as software configuration management are required for success. An organization such as UTEP that implements open source software and deploys processes for configuration management benefit as follows:

- Existence of a formal process

The existence of a formal process is always reassuring to the management. It signifies "professional approach" and existence of control in the project.

- Merges well with corporate culture

Most commercial software organizations will have one form of SCM or another. Thus any OSP that is going to obtain a corporate backing will benefit from SCM since the Software Configuration Management will help integrate the OSP better with the organization's culture.

- Satisfies one or more KPAs of SEI's CMM

(Projects, 2005)

These ideas provide the linkage between open source, configuration management processes, and the SEI's CMM. UTEP has leveraged this linkage in its implementation of DotNetNuke and the Microsoft .NET framework.

Process and Product Quality Assurance

The purpose of Process and Product Quality Assurance is to provide staff and management with objective insight into processes and associated work products. (Institute, 2002)

In the area of Product Quality Assurance, UTEP has seen the benefits of using open source software. Because the source code is open to peer review, open source software is known for its superior quality and stability. (Surrán, Sep2003) DNN is no different. The DNN development team is very active and responsive to the requests of their users. There are a significant number of entities involved in the DNN open source project, and in fact, UTEP has been a key player.

In one of the previous releases of DNN, UTEP had made hundreds of code changes within DNN to ensure that it was Section 508 compliant (section 508 is the section in the Americans with Disabilities Act that references accessibility requirements of software). UTEP contributed these code changes back to the open source project. These changes were incorporated into a future release of DNN, and UTEP was able to acquire the benefit of their work by having access to the functionality in the core DNN product. Following this process ensures that UTEP receives software meeting its requirements, and that a consistent process is followed when implementing changes. In the example of the ADA compliance code, UTEP benefits from the fact that they do not need to maintain separate code from DNN to achieve ADA compliance.

CONCLUSION

Today, UTEP has a significant implementation of a centrally managed web platform (based on DotNetNuke) that has completely renovated its web presence from a user standpoint, but has also contributed to solid software development principles and practices that have resulted in significantly improved software products produced by the Information Technology department. Although UTEP has chosen a Microsoft .NET platform with the open source portal software (DNN), other development teams, regardless of whether they follow the same technology solution, can realize the significant progress made by UTEP's development team. For example, Java developers could mimic UTEP's processes and standards with the Java programming language and the open source uPortal initiative. Again, it is not the actual technology, but the theory behind why making deliberate technology decisions can dramatically increase a development team's capabilities, and therefore improve their levels of capability as measured by the Capability Maturity Model. Bill Phifer, an EDS fellow and SEI-authorized lead CMMI appraiser, states "One of the biggest complaints voiced by the software professionals forced to implement the rigors of the CMMI is that it hurts productivity. "What many people don't realize is that if your organization is rigorously applying the principles of the Agile software development methodology, you're already doing much of CMMI without knowing it." (DevSource, 2005) In regards to UTEP, they are realizing that their decisions have allowed them to accomplish quite a bit in the CMMI without specifically attempting it.

REFERENCES

- DevSource. (2005). <http://www.Devsource.Com/article2/0,1895,1840238,00.Asp>: accessed September 15, 2005.
- DotNetNuke. (2005). <http://www.Dotnetnuke.Com>: accessed July 11, 2005.
- Gartner. (2005). http://www.Gartner.Com/4_decision_tools/measurement/measure_it_articles/2002_10/six_sig.Jsp: accessed September 15, 2005.
- Institute, S. E. (2002). *Capability maturity model® integration (cmmi) version 1.1*. Pittsburgh, Pennsylvania: Carnegie Mellon.
- Projects, S. C. M. f. O. S. (2005). <http://www.Ibiblio.Org/gferg/ldp/scm-opensource/scm-opensource.Html>: accessed on September 1, 2005.
- Sarnoladas, I., Stamelos, I., Angelis, L., & Oikonomou, A. (Oct2004). Open source software development should strive for even greater code maintainability. *Communications of the ACM*, 47(10), 83-87.
- Surran, M. (Sep2003). Making the switch to open source software. *T H E Journal*, 31(2), 36.
- WikiPedia. (2005). http://en.Wikipedia.Org/wiki/software_engineering: accessed September 1, 2005.