METHODS & TOOLS

Global knowledge source for software development professionals **ISSN 1023-4918**

Winter 2003 (Volume 11 - number 4)

Going against the stream

You will find sometimes in the software world some major trends or dominant opinions that are hard to fight. Alan Greenspan once coined the expression "irrational exuberance" for the Internet bubble and these same words could be used sometimes for some software technologies with more hype than substance. Overhyped trends tend to fade with the passing of time and their encounter with reality. We can look backward for instance to predictions like "WAP is the future of mobile phones" or "Java will be the dominant software development language". It is however more difficult to argue with some opinions that are "politically correct". You will not read many articles saying that "The open source movement is a bad thing" or "Microsoft products are very innovative and cheaply priced"... unless you look at Microsoft Web site or you find people that have a strong commercial relationship with the Redmond's giant. ;-)

More seriously, it is still possible to find some people who have a fair critical view about major trends. One of the aims of Methods & Tools is to provide opinions on every aspects of the software development world. As an editor, my goal is also to publish articles from people who don't think like me. I do not pretend to possess the truth or even that what I think is valid is also good for everybody else. In this issue, you will find a critical look at one aspect of eXtreme Programming (XP), the most famous approach of the popular Agile methods trends. Although this article is relatively focused on the specific aspect of pair programming, it provides a vision about what could be the negative consequences of adopting pair programming.

As this is the last issue of 2003, I wish to all our readers all the best for 2004 and thank all the sponsors that have supported our work during this year. I would also take the opportunity to apologise for my Northern Hemisphere occidental world references for calendar and seasons, because my goal is to make Methods & Tools a truly open and worldwide media, even when it concerns best wishes for a Happy New Year.

Franco Martinig

Inside

Understanding the Project Management Body of Knowledge Guide

Sinan Si Alhir, salhir@earthlink.net
http://home.earthlink.net/~salhir

Introduction

Every discipline focuses on an elegant universe we call reality wherein the two dimensions of time and space establish the landscape for the intertwining dance between the two natural forces of change and complexity. It is within this arena that the key ingredients of teams and people, methodologies and processes, and tools and enabling technologies converge to bridge the chasm between vision and reality. However, throughout our endeavors across domains (and independent of any particular domain), the more complexity one attempts to address, the more change that occurs, and the more change that occurs, the more complexity one breeds -- it is a vicious circle that exposes challenges and veils opportunities.

There are various means for confronting these forces, but project management as a discipline and profession is at the heart of integrating all of these means to confront the forces and achieve an endeavor's ends -- seizing opportunities to improve an organization's bottom line by delivering successful projects.

The Project Management Institute (PMI) and the Project Management Body of Knowledge (PMBOK) Guide

The Project Management Institute (PMI) (http://www.pmi.org), a not-for-profit project management professional association, publishes "A Guide to the Project Management Body of Knowledge (PMBOK Guide)," a globally recognized project management stranded. The PMBOK Guide is approved as an American National Standard (ANS) by the American National Standards Institute (ANSI), recognized by the Institute of Electrical and Electronics Engineers (IEEE) as an IEEE standard, and used as an underlying reference in an International Organization for Standardization (ISO) Technical Report on managing software projects.

Fundamentally, the PMBOK Guide is similar in nature to the IEEE's "Guide to the Software Engineering Body of Knowledge (SWEBOK)," which focuses on the software engineering profession. The term PMBOK references the sum of knowledge within the profession of project management similar to how the term SWEBOK references the sum of knowledge within the profession of software engineering. The PMBOK Guide is a document that identifies and describes the generally accepted subset of the PMBOK similar to how the SWEBOK document identifies and describes the generally accepted subset of the SWEBOK. Generally accepted involves being applicable to most projects most of the time and being widely regarded as valuable and useful. Such bodies of knowledge (BOKs) provide a common lexicon and basic reference as a foundation for evolving a discipline into a profession wherein practitioners and academics collaborate, similar to how other professions such as accounting, law, and medicine have evolved.

The PMBOK Guide describes a project management framework. The PMBOK Guide is broadly applicable to different types of projects, including small-scale and large-scale projects having various degrees of managerial and technical complexity, across different domains or industries and organizational cultures. The PMBOK emerged from the efforts of the PMI, which was founded in 1969, and its constituent members through various milestones to culminate in the publication of the PMBOK Guide 1996 edition, which was subsequently followed by a PMBOK Guide 2000 edition, and is currently evolving into the upcoming PMBOK Guide 2004 edition.

Projects

A *project* is a temporary endeavor that focuses on creating a unique result. The endeavor is temporary in that it has a definite beginning and a definite end. The end is reached when a project's objectives are achieved, when it is determined that the objectives will not or cannot be achieved, or when the need for the project no longer exits, in which case the project is terminated. The result, which may be a product or service, is unique in that it is distinguishable from all other results. The temporary aspect of a project does not necessarily apply to the result of the project and the product or service may outlive the project. Furthermore, projects are critically distinguished from operations, which are ongoing and have regular or rhythmic results.

Scope

As a project integrates the concepts of temporary and unique, the distinguishing characteristics of a product or service must be *progressively elaborated*. Progressively indicates that a project must proceed in steps or increments and the distinguishing characteristics of its results must be understood in steps or increments. Elaborated indicates that a project must be developed more thoroughly as it progresses and the distinguishing characteristics of its results may be broadly defined early in the project and made more explicit and detailed as the project progresses. Progressive elaboration emphasizes that project management processes are *iterative* in nature and ongoing throughout a project such that processes may be iterated several times. However, the scope of a project, the work to be done, remains fairly constant while the scope of the result, the characteristics of the product or service, is progressively elaborated.

The distinction between project scope and product scope is essential. *Product scope* involves the characteristics of a result, the functions and features that characterize a product or service, and the product's completion is measured against the requirements of the product or service. *Project scope*, which is also known as the *work scope of a project*, involves the work that must be done to deliver a product or service with the characteristics determined by product scope, and the project's completion is measured against its plan. Thus, product scope relates to a product or service while project scope relates to a project.

Lifecycle

A project is divided into generally sequential but possibly somewhat overlapping *phases*, also known as *stages*, which are collectively known as a *project lifecycle*. Each phase results in one or more tangible and verifiable work products, which are known as *deliverables*. Each phase ends with a review, known as a *phase exit* or *stage gate* or *kill point*, of the deliverables and the project's performance to detect and correct errors and also to determine if the project should continue. A *stakeholder* is an individual or organization involved in the project, who may be positively or negatively affected by the execution or completion of the project, and who may exerting influence on the project. Phases are related through their deliverables where the deliverables from prior phases are used by subsequence phases. Furthermore, each phase, which may be further divided into sub-phases and so forth, may be considered a project. The distinction between a project lifecycle and a product lifecycle is essential. A *product lifecycle* involves phases through which a product evolves. A *project lifecycle* forms a single phase of a product life cycle. Thus, a project life cycle is but a single part of a product life cycle.

Project Management

Project management involves the application of best practices to meet project objectives. A *project manager* is the stakeholder who is responsible for managing a project. The PMBOK Guide describes an integrative project management framework, which provides a basic structure for understanding project management practices, composed of processes organized into process groups and knowledge areas. Process groups are more formally known as *project management process groups*, knowledge areas are more formally known as *project management knowledge areas*, and each knowledge area is referenced as *X management* where X is the name of the knowledge area and represents a subset of project management.

Project management, projects, and the framework are integrative in that they require focusing on goals or objectives, balancing competing demands, and leveraging feedback to ensure the goals or objectives are achieved given the competing demands. The *project triple constraint*, which includes project scope, time, and cost parameters, provides a framework for evaluating competing demands, integrating processes and their interactions, and ultimately achieving project objectives and goals. The project triple constraint is often shown as a triangle where either each side or corner represents one of the parameters being managed such that when any side or corner changes, the other sides or corners also change to maintain the shape of the triangle.

Figure 1, from "Epoch: Strategy for Success" (http://home.earthlink.net/~salhir), shows a conceptual view of the process groups and knowledge areas. The process groups are shown organized at the corners within the delimiting solid-outline rectangle and include Initiating, Planning, Executing, Controlling, and Closing. The knowledge areas are shown organized within the solid-outline rectangle around the project triple constraint triangle and include Integration, Scope, Time, Cost, Quality, Human Resource, Communications, Risk, and Procurement.

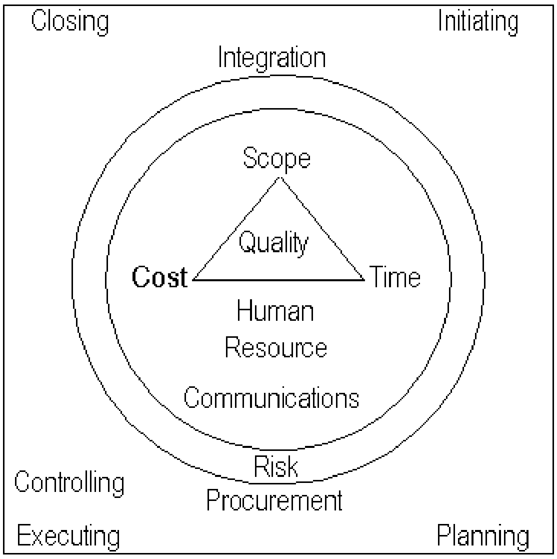


Figure 1: Process Groups and Knowledge Areas

Processes

A *process* is a collection of actions focused on producing a result. Similar to the distinction between project scope versus product scope and project lifecycle versus product lifecycle, the distinction between project management processes and product-oriented processes is essential. *Project management processes* focus on the work of a project. *Product-oriented processes* focus on a project's result, product or service. Thus, project management processes are generally accepted while product-oriented processes are application area specific, but both sets of processes are interdependent and interact throughout a project.

Processes are linked by their inputs and outputs and described by their inputs, tools and techniques, and outputs. Inputs are items that are acted upon by tools and techniques. Outputs are items that result from applying tools and techniques. Tools and techniques are mechanisms applied to inputs to produce outputs. Furthermore, processes are *iterative* in nature and ongoing such that they may be iterated several times during one phase of a project. Core processes are processes that are performed in generally the same order due to their dependencies while facilitating processes are process that are performed intermittently as needed. Furthermore, not all of the processes are needed for every project and not all of their interactions apply to every project. Core processes are bolded in the figures herein and general ordering dependencies are shown using solid-line paths from the independent process groups or knowledge areas to the dependent process groups or knowledge areas.

The quintessential approach to understanding the PMBOK Guide includes understanding (1) how processes from different knowledge areas are grouped into process groups and the relationships among process groups, thus providing a broad view of project management, and (2) how processes from different process groups are grouped into knowledge areas and the relationships among knowledge areas, thus providing a detailed view of project management.

Process Groups

A process group organizes processes based on when they occur in a project or phase. Process groups are linked by the results they produce where the output from prior process groups become input to subsequence process groups. Process groups are not discrete or one-time events but are overlapping activities that are *iterative* in nature such that they may be iterated several times and occur at varying levels of intensity across a project, within and across phases of a project, and are commensurate with the scope of the project and the value the process group adds to the specific project.

Figure 2 shows the relationships among process groups for a project or phase.

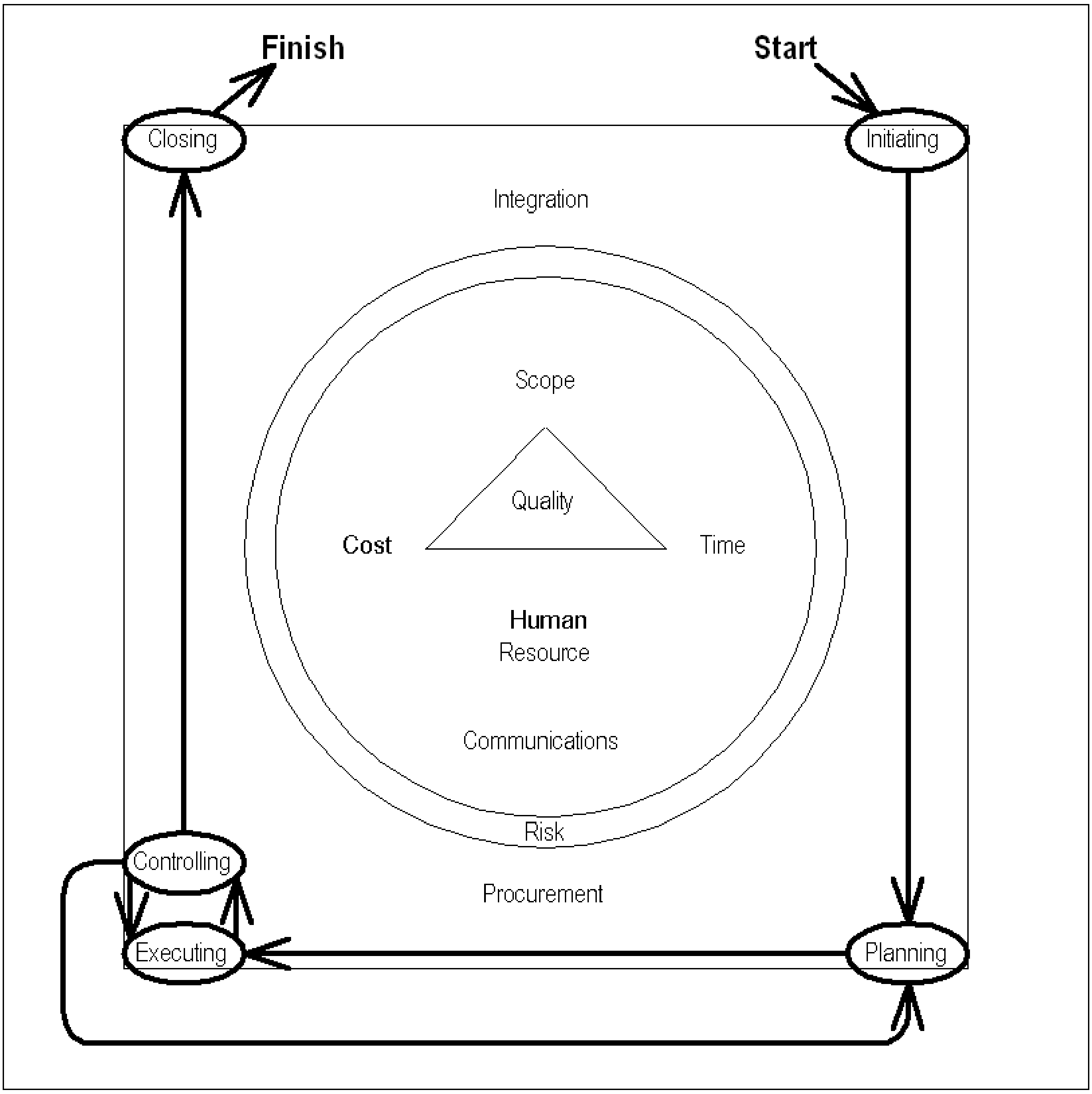


Figure 2: Process Groups

Initiating

The Initiating process group focuses on authorizing a project or phase. This process group generally occurs at the beginning of a project or phase. The results from this process group become input into the Planning process group. Figure 3 highlights the Initiating processes organized by knowledge areas. Notice that the overall theme is scope.

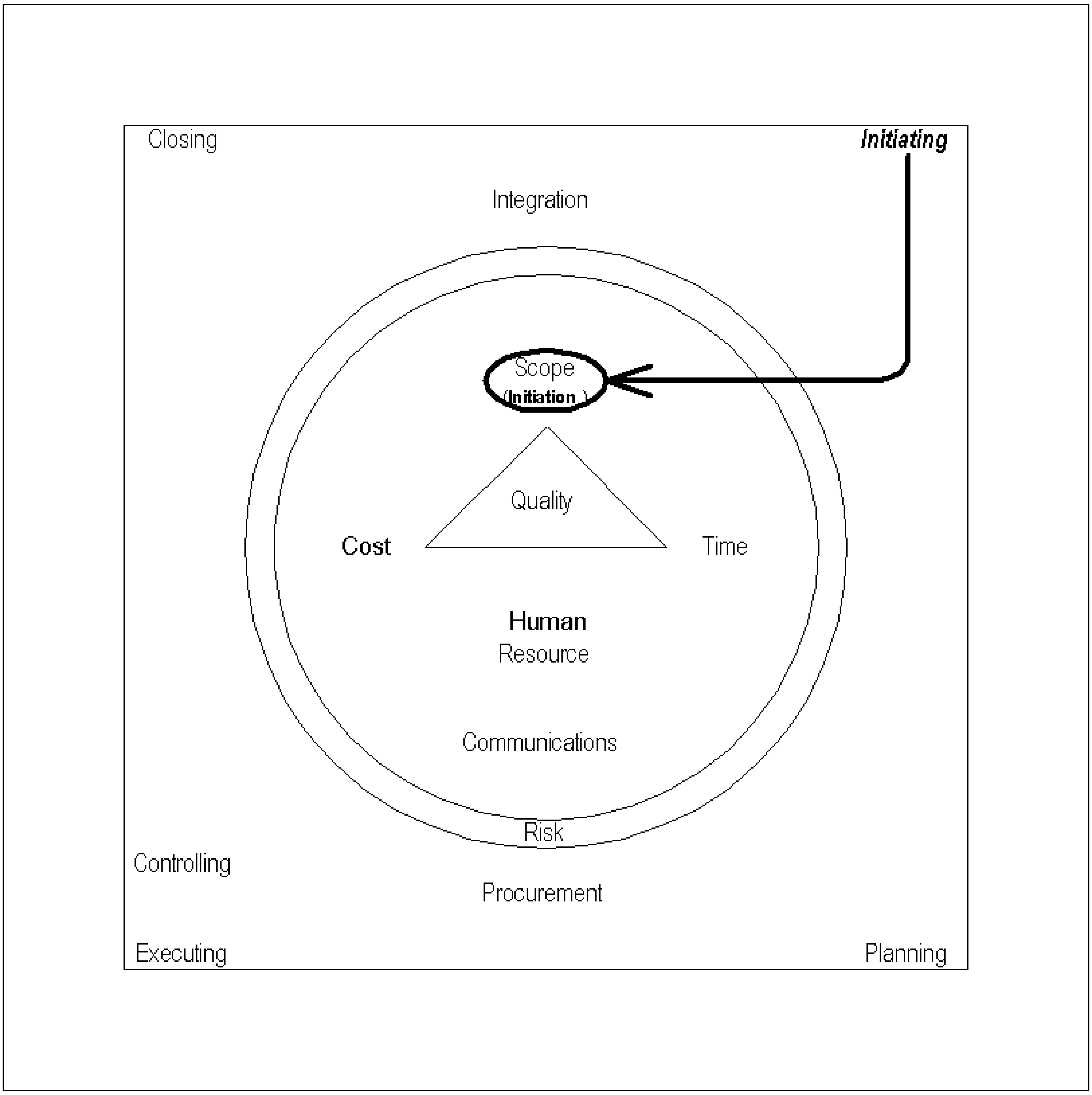


Figure 3: Initiating Processes and Knowledge Areas

Planning

The Planning process group focuses on defining project or phase objectives and determining the best alternative for attaining the objectives. This process group generally peeks early in a project or phase and steadily decreases. The results from this process group become input into the Executing process group. This process group coincides with the Plan part of the common Plan-Do-Check-Act quality cycle. Figure 4 highlights the Planning processes organized by knowledge areas with their general ordering. Notice that the overall theme involves time and cost based on scope and risk.

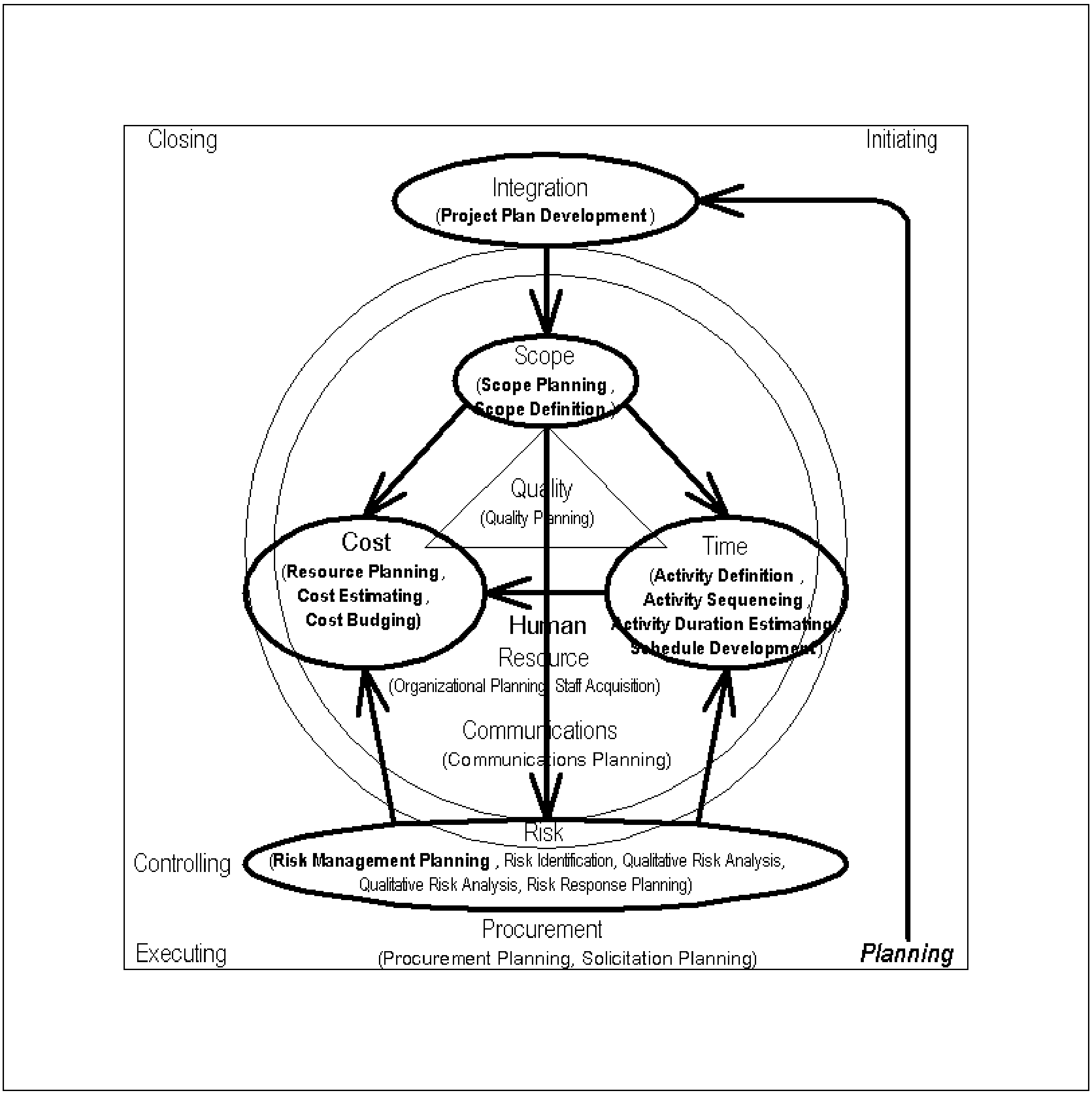

Closing
Initiating
Integration
(Project Plan Development)
Scope
(Scope Planning, Scope Definition)
Quality
(Quality Planning)
Cost
(Resource Planning, Cost Estimating, Cost Budging)
Time
(Activity Definition, Activity Sequencing, Activity Duration Estimating, Schedule Development)
Human Resource
(Organizational Planning, Staff Acquisition)
Communications
(Communications Planning)
Risk
(Risk Management Planning, Risk Identification, Qualitative Risk Analysis, Qualitative Risk Analysis, Risk Response Planning)
Controlling
Procurement
(Procurement Planning, Solicitation Planning)
Executing
Planning

Figure 4: Planning Processes and Knowledge Areas

Executing

The Executing process group focuses on coordinating resources to perform the best alternative for attaining project or phase objectives. This process group generally steadily increases and peeks later in a project or phase. The results from this process group become input into the Controlling process group. This process group coincides with the Do part of the common Plan-Do-Check-Act quality cycle. Figure 5 highlights the Executing processes organized by knowledge areas. Notice that the overall theme is quality.

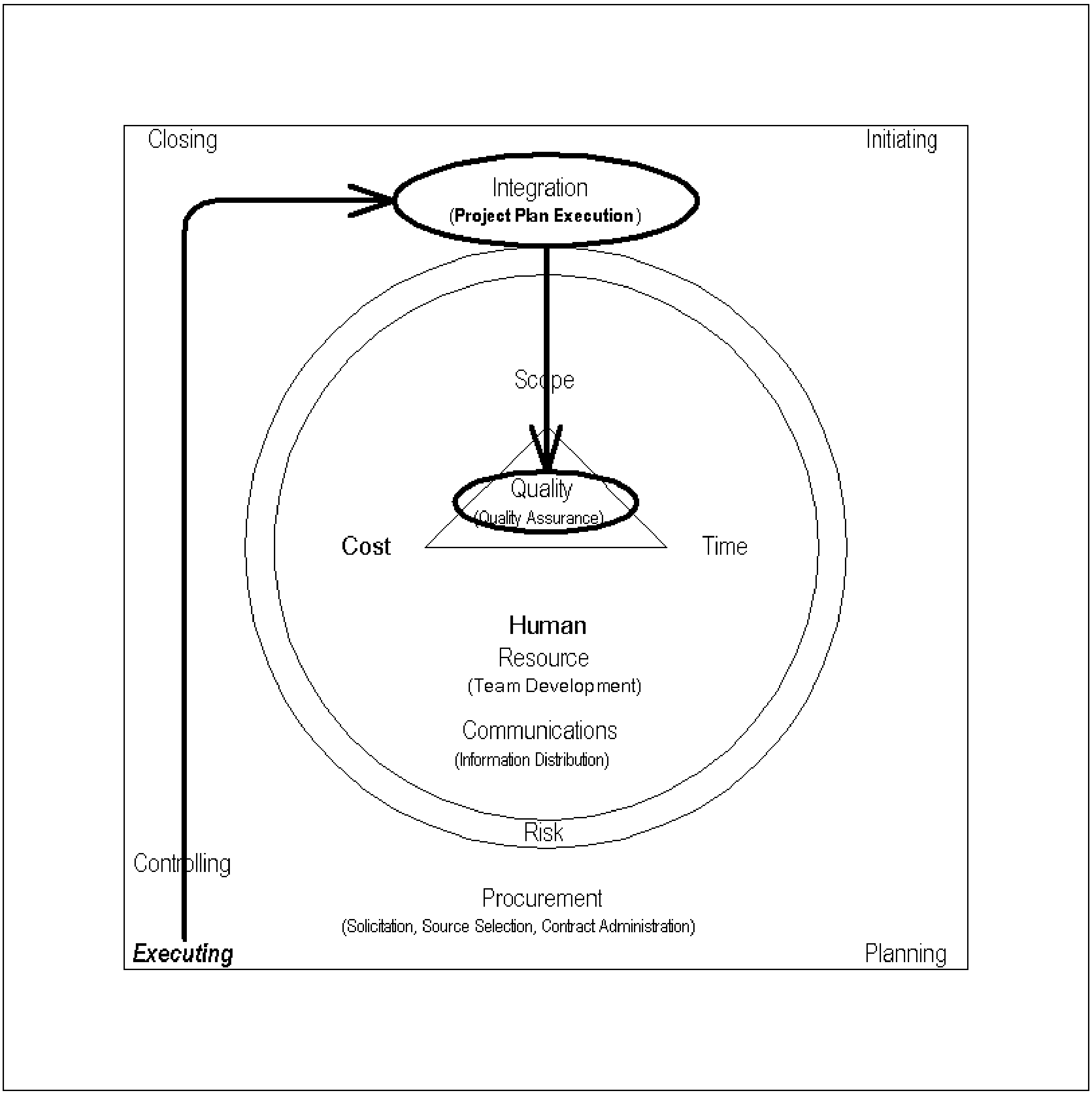


Figure 5: Executing Processes and Knowledge Areas

Controlling

The Controlling process group focuses on monitoring and measuring progress to identify variances from the best alternative for attaining project or phase objectives and take corrective actions to address the variances and ensure that the objectives are met. This process group generally steadily increases and peeks in the middle of a project or phase and steadily decreases. The results from this process group become input into the Planning, Executing, and Closing process groups. This process group coincides with the Check and Act parts of the common Plan-Do-Check-Act quality cycle. Figure 6 highlights the Controlling processes organized by knowledge areas with their general ordering. Notice that the overall theme involves time and cost based on quality and risk to address scope.

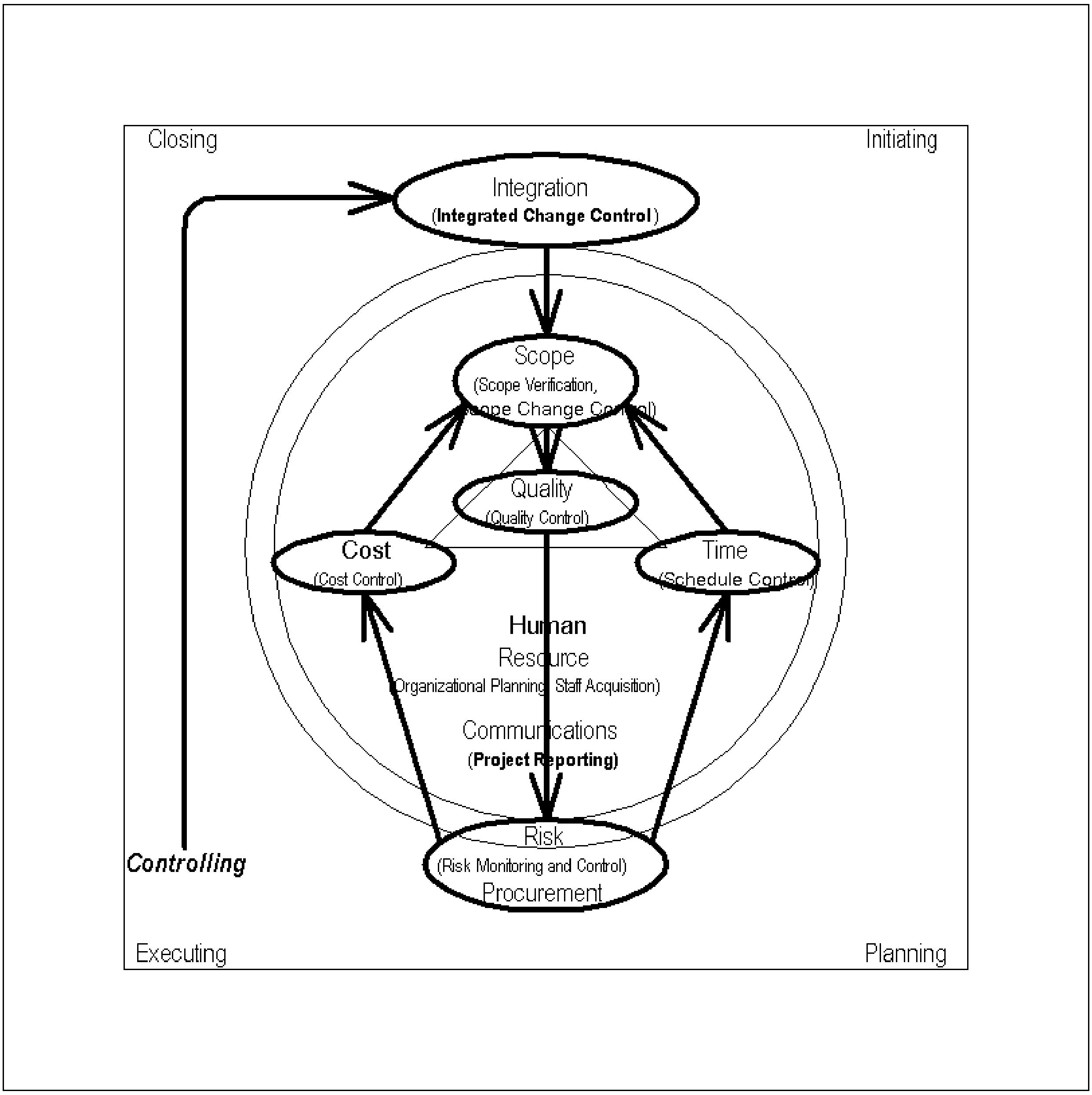


Figure 6: Controlling Processes and Knowledge Areas

Closing

The Closing process group focus on formalizing acceptance of a project or phase and ends the project or phase. This process group generally occurs at the end of a project or phase. The results from this process group become input into the Initiating process group in the next project or phase. Figure 7 highlights the Closing processes organized by knowledge areas. Notice that the overall theme is communication.

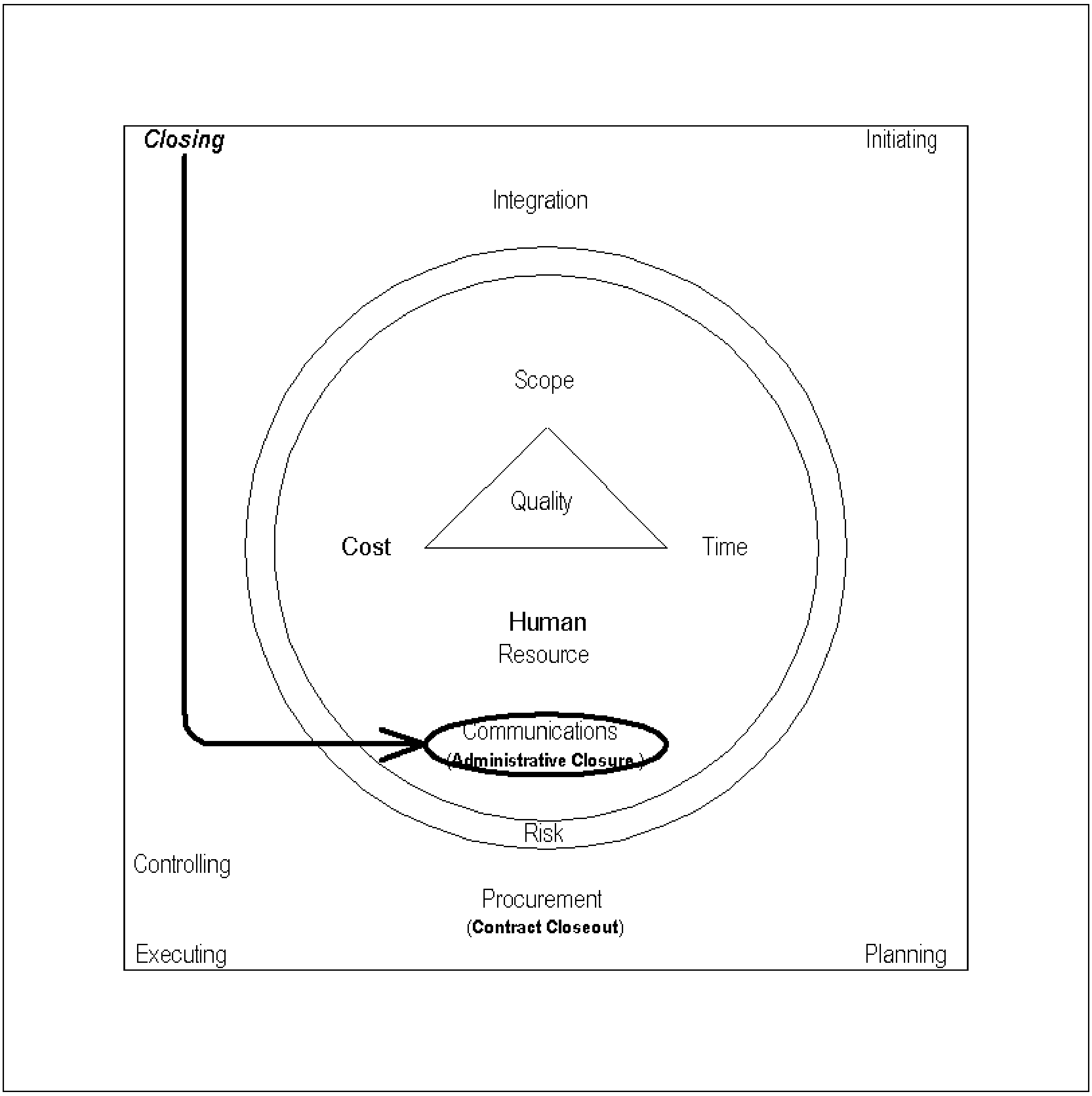


Figure 7: Closing Processes and Knowledge Areas

Knowledge Areas

A knowledge area organizes processes based on their subject or area of concern or focus. Knowledge areas are linked by the results they produce where the output from one knowledge area becomes input to other knowledge areas. Knowledge areas are not discrete or one-time events but are overlapping activities that are *iterative* in nature such that they may be iterated several times and occur at varying levels of intensity across a project, within and across phases of a project, and are commensurate with the scope of the project and the value the knowledge area adds to the specific project.

Integration

The Integration knowledge area focuses on integrating and coordinating the various elements of a project. The Project Plan Development core process focuses on developing a project plan. A *project plan* captures planning assumptions, decisions, approved baselines (scope, schedule, and cost), and facilitates communication and also guides project execution and project control. A project plan commonly includes the project charter, project management approach, scope statement, work breakdown structures, performance baseline (schedule, cost estimates, and budget), milestones, other plans, open issues, and pending decisions. The progressive elaboration of a project plan is known as *rolling wave planning* to emphasize that planning is an *iterative* and ongoing process. The Project Plan Execution core process focuses on performing the activities in the project plan. The Integrated Change Control core process focuses on addressing changes to the project.

Scope

The Scope knowledge area focuses on a project's deliverables.

The Initiation core process focuses on developing a project charter. A *project charter* authorizes a project and provides a project manager with the authority to manage the project. A project charter commonly includes a description of the business needs that the project results are intended to address and a description of the results, a product or service description.

The Scope Planning core process focuses on developing a scope statement. A *scope statement* captures a common understanding of the project scope and project objectives, which are used to make future decisions. A scope statement commonly includes project justification in terms of the business needs summarized from the project charter, project product in terms of the results description summarized from the project charter, project deliverables summarized and known exclusions identified, and project objectives. *Project objectives* are quantifiable success criteria for time, cost, and quality and include attributes, metrics, and target absolute or relative values. Any un-quantified objectives such as "customer satisfaction" entail high risk.

The Scope Definition core process focuses on developing a work breakdown structure (WBS). A *work breakdown structure* captures the work scope of a project as a hierarchical deliverable-oriented grouping of project elements where the lowest level elements are known as *work packages* and all of the elements are described in a *WBS dictionary*.

The Scope Verification facilitating process focuses on formalizing acceptance of the project scope. The Scope Change Control facilitating process focuses on addressing changes to the project scope.

Time

The Time knowledge area focuses on a project's schedule. The Activity Definition core process focuses on identifying activities in an activity list to produce the project deliverables. An *activity list* is an extension to the work breakdown structure. The Activity Sequencing core process focuses on identifying dependencies among activities. The Activity Duration Estimating core process focuses on estimating the time required for completing activities. The Schedule Development core process focuses on developing a project schedule. A *project schedule* captures the planned dates for activities and milestones. The Schedule Control facilitating process focuses on addressing changes to the project schedule.

Cost

The Cost knowledge area focuses on a project's budget. The Resource Planning core process focuses on determining the resources and their quantities required for performing project activities. The Cost Estimating core process focuses on determining an estimate of the cost of the resources. The Cost Budgeting core process focuses on determining the cost of project activities and establishing a cost baseline. A *cost baseline* captures the planned cost for a project or phase. The Cost Control facilitating process focuses on addressing changes to the project budget.

Quality

The Quality knowledge area focuses on the quality of a project's performance and results. The Quality Planning facilitating process focuses on identifying quality standards. The Quality Assurance facilitating process focuses on evaluating project performance. The Quality Control facilitating process focuses on evaluating project results.

Human Resource

The Human Resource knowledge area focuses on the individual people involved with a project. The Organizational Planning facilitating process focuses on identifying roles and responsibilities. The Staff Acquisition facilitating process focuses on acquiring human resources. The Team Development facilitating process focuses on enhancing individual and group competencies.

Even though the preceding figures do not explicitly highlight this knowledge area, it is essential, vital, and fundamental to a project's success, and could be highlighted on the preceding figures.

Communications

The Communications knowledge area focuses on communicating appropriate project information. The Communications Planning facilitating process focuses on determining the communication needs of the stakeholders (who needs what, when, how it will be communicated, and perhaps where and why). The Information Distribution facilitating process focuses on distributing information. The Performance Reporting core process focuses on capturing and distributing performance information. The Administrative Closure core process focuses on distributing information bringing closure to a project or phase.

Even though the preceding figures do not explicitly highlight this knowledge area, it is essential, vital, and fundamental to a project's success, and could be highlighted on the preceding figures.

Risk

The Risk knowledge area focuses on confronting risks to a project. A *risk* is an uncertain event or condition that may have a positive or negative/adverse effect on project objectives if actualized. The Risk Management Planning core process focuses on deterring how to approach risk management. The Risk Identification facilitating process focuses on identifying risks and triggers. A *trigger*, also known as a *symptom* or *warning sign*, is an indicator that a risk has occurred or is about to occur. The Qualitative Risk Analysis facilitating process focuses on prioritizing risks based on their impact on project objectives. The Quantitative Risk Analysis facilitating process focuses on determining the probability of risks occurring and their effects on project objectives. The Risk Response Planning facilitating process focuses on determine how to enhance opportunities and reduce threats to project objectives. The Risk Monitoring and Control facilitating process focuses on monitoring risks, identifying new risks, responding to risks, and evaluating the effects of responses.

Procurement

The Procurement knowledge area focuses on acquiring goods and services external to the organization performing the project. The Procurement Planning facilitating process focuses on determining what to procure. The Solicitation Planning facilitating process focuses on determining procurement requirements and identifying potential sources. The Solicitation facilitating process focuses on obtaining responses from potential sources. The Source Selection facilitating process focuses on selecting a source. The Contract Administration facilitating process focuses on managing the relationship with the source. The Contracts Closeout core process focuses on bringing closure to the procurement.

Conclusion

Unequivocally, people are and will remain the “original ingredient” necessary for success. However, with the project management discipline as a profession and the PMBOK Guide, project managers and teams are further empowered not only to simply address change and complexity, but leverage change and complexity for a competitive advantage by delivering successful projects. Furthermore, it is experience, experimentation, and application of the PMBOK Guide that will enable us to realize its benefits.

Regulatory Compliance and IT – Converging Challenges

Patrick McElrea, pmcelrea@mks.com
MKS Inc., www.mks.com

How IT Process Methodologies and Change Management Solutions Can Help Your IT Organization be Audit Ready

In the past decade, two important trends have converged to create a formidable challenge for organizations worldwide. First, regulatory environments have become more complex and demanding as governments strive to protect the public and strengthen oversight. Second, information technology (IT) is playing a bigger role than ever before in helping companies deliver their products and services to their customers. The pervasiveness of IT in today's business environment, therefore, points to its potentially critical role in regulatory compliance.

This paper argues that regulatory compliance can be made easier through improved IT processes and auditing. In the following sections, we look at three popular frameworks or methodologies for bringing control to IT processes so that organizations are audit ready. Secondly, it looks at the role that software configuration management and process/workflow management solutions can play in helping implement these critical processes while providing reliable data for an IT audit.

Without a plan or methodology, however, SCM and process/workflow management solutions are useless. The three frameworks or methodologies discussed below are a good starting point for bringing discipline to software development and IT processes in general. ITIL is a library of best practices for the provision of quality IT services. CobiT is an IT governance framework that can be applied to the entire IT realm and its processes in general, and the Capability Maturity Model (CMM) represents a more detailed and granular approach to controlling individual processes within the IT realm. While these are guidelines, there are no "one-size-fits-all" frameworks.

ITIL

Originally created by a UK government agency, ITIL (IT Infrastructure Library) claims to be the most widely accepted approach to IT Service Management in the world. While it is used extensively in Europe and Asia it has only recently gained wider visibility in North America.

ITIL consists of a series of books giving guidance on the provision of quality IT services. It provides a cohesive set of best practices, drawn from the public and private sectors internationally and is supported by a comprehensive qualification scheme, accredited training organizations, and implementation and assessment tools.

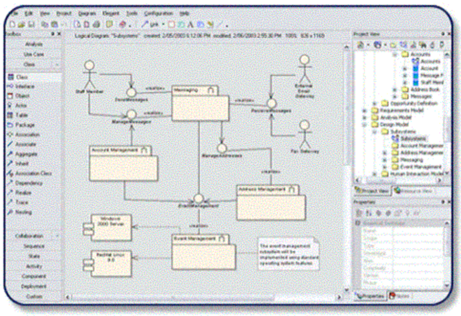
ITIL is organized into a series of sets, which themselves are divided into two main areas – service support and service delivery:

- Service Support is the practice of those disciplines that enable IT Services to be provided effectively.
- Service Delivery covers the management of the IT services themselves. It involves a number of management practices to ensure that IT services are actually provided as agreed between the Service Provider and the Customer.

Each of these two areas contain a number of disciplines, which themselves stipulate the ITIL practices/requirements. For the purposes of this paper, Services Support is the more relevant because it deals with, among other areas, Configuration Management and Change Management. According to ITIL, Configuration Management is the implementation of a database (Configuration Management Database – CMDB) that contains details of all the elements used in the provision and management of an organization’s IT services. This is more than just a list of assets, as it contains information that relates to the maintenance, movement, and problems experienced with the Configuration Items. The CMDB also holds a much wider range of information about items that the organization’s IT Services are dependant upon. This range of information includes hardware, software, documentation, and personnel.

Change Management is the practice of ensuring all changes to Configuration Items are carried out in a planned and authorized manner. This includes ensuring that there is a business reason behind each change, identifying the specific Configuration Items and IT services affected by the change, planning the change, testing the change, and having a rollback plan should the change have an adverse affect on the Configuration Item.

Implied in all this is the use of high quality software configuration management and process/workflow management solutions for providing the organization with the ability to control the management of each item (including changes and who is authorized to make them); the tracking of the status of all Configuration Items and the maintenance of this information; and the ability to verify or audit to ensure that information contained in the database is accurate. All these activities (control, tracking, and verification) are critical to the audit process when proving compliance with regulations.



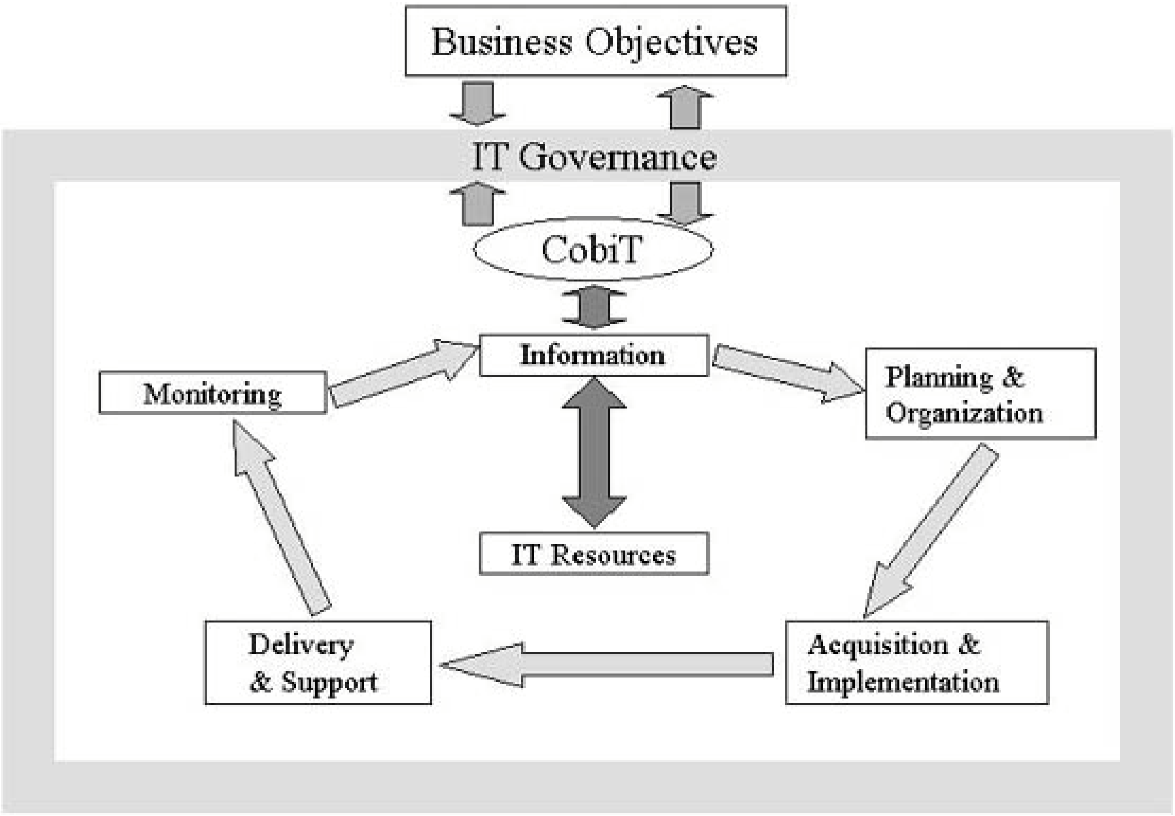
COBIT

CobiT (Control Objectives for Information and Related Technology) was developed by the IT Governance Institute as a generally applicable and accepted standard for good Information Technology (IT) security and control practices that provides a reference framework for management, users, and IS audit, control and security practitioners. The institute was founded in 1998 by the Information Systems Audit and Control Association (ISACA) as a not-for-profit organization dedicated to sharing better practices for IT governance,.

According to CobiT, IT governance is a structure of relationships and processes to direct and control the enterprise in order to achieve the enterprise's goals by adding value while balancing risk versus return over IT and its processes. It provides the structure that links IT processes, IT resources and information to enterprise strategies and objectives.

CobiT's framework for IT governance identifies 34 key, naturally grouped IT *Control Objectives*, which fall under one of four broad domains: planning and organization (11), acquisition and implementation (6), delivery and support (13), and monitoring (4). Each control objective can be regarded as a separate process to which CobiT's *Management Guidelines* are applied. The management guidelines are governed by a generic maturity model that allows managers to map where the organization is today, where it stands in relation to the best-in-class in its industry and to international standards and where the organization wants to be. The following section discusses maturity models, and in particular the Software Capability Maturity Model, as a means for controlling software development processes.

As it relates to regulatory compliance, CobiT represents an excellent reference point for assessing current internal process controls and implementing new and improved ones. An IT governance model such as this is a worthy goal to aspire to in the longer term, but to comply with regulatory mandates, more immediate and short term actions can and should be taken.



Source: ISACA

The Capability Maturity Model (CMM)

Like any business process, IT processes lend themselves to auditing activities that track their effectiveness in achieving business goals. Key to this measurement is the use of maturity models for self-assessment and benchmarking. Maturity models are effective tools for determining the current status of the organization’s processes and how they should evolve. They provide both the goals to strive for and the means of measuring the attainment of those goals. If you are planning to audit your IT processes, they provide one of the readiest tools for preparing for it. There are five levels that make up the generic maturity model (see graphic).

To understand how maturity models are applied in the real world, consider the example of software development processes. For a number of years, software development organizations have used the Capability Maturity Model for Software (SW-CMM) as the de facto standard for assessing and improving software processes. It was developed by the software community under the stewardship of the Software Engineering Institute (SEI) at Carnegie Mellon.

The CMM is composed of five maturity levels. With the exception of Level 1, each maturity level is composed of several key process areas (KPA). Each KPA is organized into five sections called Common Features. The Common Features are attributes that indicate whether the implementation and institutionalization of a key process area are effective, repeatable, and lasting. The Features sections are: Commitment to Perform, Ability to Perform, Activities Performed, Measurement and Analysis, and Verifying Implementation.

Level 1, **Initial** - Software processes are ad hoc. There are few defined processes and success depends on individual effort.

Level 2, **Repeatable** - Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.

Level 2 key process areas

- Requirements Management
- Software Project Planning
- Software Project Tracking and Oversight
- Software Subcontract Management
- Software Quality Assurance
- Software Configuration Management.

Level 3, **Defined** - The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software.

Level 3 key process areas

- Organization process focus
- Organization process definition
- Training program
- Integrated software management
- Software product engineering
- Intergroup coordination
- Peer reviews

Level 4, **Managed** - Detailed measurements of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.

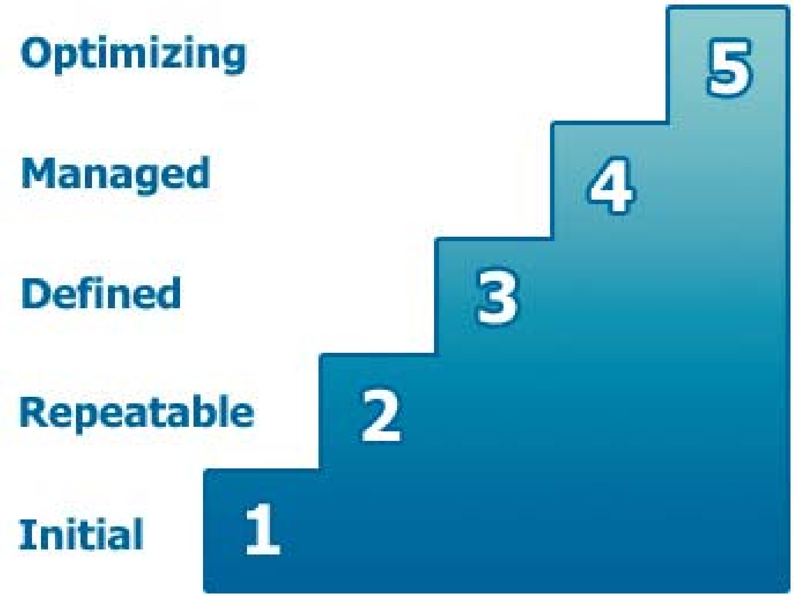
Level 4 key process areas

- Quantitative process management
- Software quality management

Level 5, **Optimizing** - Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.

Level 5 key process areas

- Defect prevention
- Technology change management
- Process change management



Generic maturity model levels – Source: Software Engineering Institute

What About ISO 9000?

ISO 9000 is a well known generic management system standard, which means it is concerned with the way an organization goes about its work, and not directly the result of this work. This standard can be applied to any organization, large or small, whatever its product or service in any sector of activity, including business, public administration, or government. If you consider software applications or hardware configurations as internal end products, then ISO standards can be helpful for achieving a high level of quality and will assist any audit, but they do not specifically address IT processes. For that, the frameworks mentioned above are specially designed for these purposes and should be consulted for more specific direction and development of processes.

SCM and Process & Workflow Management Solutions

Software Configuration Management (SCM) and process/workflow management solutions provide control over software and IT processes to make them more verifiable and audit ready. They also offer an effective way of controlling IT processes at a modest price. SCM was designed to provide assurance that a company’s mission critical software applications are not exposed to potential failure due to human error, staff turnover or sabotage. As SCM has become better understood and applied in different areas, however, a secondary but important role has emerged. SCM, in conjunction with a flexible process and workflow management solution, provides the ability to capture, track, version and report on changes to any process or system in an IT setting.

Process & Workflow Management

A flexible process and workflow management solution allows you to implement workflows that are customized for any IT process or system. It records every change and/or action made by every person involved in a given process, providing valuable details about “who’s done what”. It is critical that workflows are completely enforceable, meaning that a process cannot be subverted by an overzealous or malicious employee who wishes to skip steps in the process.

Go/no-go gates are one of the mechanisms that provide managers with the ability to enforce workflows and decide when the process can proceed and when it must remain stopped until another person in the process completes his action. And while not absolutely necessary, it is helpful if the process and workflow management solution has a graphical workflow modeler to make for easy graphical editing of workflows while providing a clear overall picture of the people and actions involved.

SCM

Software Configuration Management plays a central role in software development with its ability to version any type of file, guarantee the reproducibility of an application, and provide audit trails for illustrating migrations throughout the software development process.

In the wider IT environment, however, SCM solutions provide value for regulatory compliance through their versioning capabilities and integration with a process and workflow management tool. In a typical company, processes and workflows are defined and documented, and implemented, in that order. Process and workflow management allows you to implement and enforce those processes, while SCM performs the versioning of the process documents. This is not a trivial task. As processes improve and evolve, process documents will undergo almost constant revision. In an audit situation, a separation of duties and clear audit trail must be evident to illustrate that IT processes are up to date and in synchronization with what is being practiced by staff. An audit trail of approvals is also critical for demonstrating that internal controls are working properly.

Conclusion

A disciplined approach to internal process controls and good IT governance are key to regulatory compliance. Software Configuration Management and process/workflow management solutions help implementing these processes, but it must be remembered that the solutions are only as good as the processes and methodologies that govern their use. By adopting proven methodologies such as ITIL, CobiT, and/or CMM you will improve your organizations IT processes, making them audit-ready while ultimately contributing to your organization’s regulatory compliance.

Helpful Online Resources

Official ITIL Web Pages - http://www.ogc.gov.uk/index.asp?id=2261

The Information Systems Audit and Control Association & Foundation (ISACA) - http://www.isaca.org

Carnegie Mellon SEI (SW-CMM) - http://www.sei.cmu.edu/cmm/cmm.html

IT Governance Portal - http://www.itgovernance.org/

References

Comrie, George R., Software Development is Risky Business – Is it Audit Ready?, 2001, ISACA InfoByte - ISACA Website.

International Organization for Standardization, Demystifying ISO 9000 and ISO 14000, ISO Website - http://www.iso.ch/iso/en/iso9000-14000/tour/magical.html

Thomke, Stefan, R&D Comes to Services: Bank of America's Pathbreaking Experiments, Harvard Business Review, April, 2003.

IT Governance Institute, COBIT 3rd Edition – Executive Summary, July, 2000.

IT Governance Institute, COBIT 3rd Edition – Management Guidelines, July, 2000.

ITIL Toolkit, Website for ITIL documents and publications - http://www.itil-toolkit.com/itil-audit.htm

KPMG, Sarbanes-Oxley Section 404: Management Assessment of Internal Control and the Proposed Auditing Standards, March 2003.

MacSweeney, Greg, Governance Falls Into CIO's Lap, Wall Street & Technology Online, May 29, 2003.

Office of Government Commerce, ITIL: The Key to Managing IT Services, ITIL Website - http://www.ogc.gov.uk/index.asp?id=2261

Worthen, Ben, Playing By New Rules – Sarbanes-Oxley: Your Risks and Responsibilities, CIO Magazine, May 15, 2003.

Will Pair Programming Really Improve Your Project?
(A critical look at the book *Pair Programming Illuminated*)

Matt Stephens and Doug Rosenberg
Software Reality, www.softwarereality.com

This article is an excerpt from Chapter 6 of the book **Extreme Programming Refactored: The Case Against XP [1]**, by Matt Stephens and Doug Rosenberg. The book provides an entertaining look at some of the flaws behind Extreme Programming (XP), whilst suggesting some alternative strategies and practical techniques to achieve XP's agile goals in a more rigorous way. For this article we concentrate on pair programming - and in particular the book *Pair Programming Illuminated* [2] by Laurie Williams and Robert Kessler.

Pair Programming Illuminated (we refer to it as PPI for the rest of this article), not surprisingly given its title, pitches the case for pair programming. However, it also discusses quite openly some of the problems typically encountered by pair programming teams. In this article, we focus on some of those problems and examine how they may affect an XP project as a whole.

Problems with Pairing Different Categories of Programmer

PPI divides programmers into different categories and then discusses the effects of the various combinations thereof. The programmer categories are novice, average, expert, introvert, and extrovert. The pairing combinations discussed in PPI, with a chapter dedicated to each, are as follows:

- Expert-expert
- Expert-average
- Expert-novice
- Novice-novice
- Extrovert-extrovert
- Extrovert-introvert
- Introvert-introvert

Because pairs are meant to rotate frequently, these various combinations will resurface often in a team of mixed abilities. Thus, in small teams (which is likely, given an XP project), it would be difficult to keep “problem pairs” apart.

“Go Make Me a Cup of Tea” Syndrome

What happens if you pair up a newbie programmer with an expert? This is described in PPI as “expert-novice pairing.” The intention of such a pairing would be to “get the easier job done well, while training a novice programmer.” The challenge of such a pairing is primarily that the expert must take on a tutoring role and must maintain extreme patience throughout. If the expert coder slips, then the result is a “watch while I type” session (sometimes called “go make me a cup of tea while I finish this program” syndrome [1]), in which the novice remains passive throughout and the expert is effectively solo-coding. Despite this, there are distinct advantages to expert-novice pairing. In fact, it’s probably the one pairing combination that’s worth mandating, as long as the novice is willing and able to learn and the expert is prepared to give up a portion of her day to teach rather than code in full-flow. This combination is certainly better than novice-novice pairing, which even XP evangelist Ron Jeffries thinks is a bad idea [2].

Laurel and Hardy Take Up Pair Programming

The intent of a novice-novice pairing combination is described in PPI as follows:

"To produce production code in a relatively noncomplex area of the project, giving valuable experience to both programmers in the process." [2]

If you're considering such a pairing, it's important to ask yourself which part of your project is unimportant enough that you can afford to unleash two complete novices, unsupervised, on it. "Unsupervised" is actually the key. Two novices, unsupervised, would likely produce code that isn't exactly production quality. Luckily, PPI has the answer:

"There must be a coach, instructor, or mentor available to answer questions and also to help guide the pair. . . .We feel very strongly about the need for a coach. If you are unwilling to assign the mentoring task to some expert, then you need to understand the limitations of the asset being produced by the pair." [2]

In XP, this responsibility would fall into the lap of the person (or people) performing the coach role.

As with the other pairing combinations, pairs rotate so frequently that in a team of mixed abilities, the novice-novice pairing could happen quite often. Therefore, novice-novice pairing isn't something that can easily be controlled: It just happens, almost by accident, several times a week. The coach must be fully aware of the fact that two novices are currently pairing at any time, and the coach must be available to guide them and correct their mistakes. In practice, to combat the proverbial blind leading the blind, there's a risk that the coach may become fully occupied with mentoring one particular pair anytime two novices pair up.

Carrying Your Pair

Similar but less extreme problems occur with expert-average pairing. PPI describes three situations where the authors feel that expert-average pairing is a problem. The first is that the average programmer truly is average (i.e., the average programmer is likely to stay that way and will never really progress). The second is when the average programmer doesn't interact enough with the expert. The third is when the average programmer doesn't seem to "get it" and keeps asking the same question over and over:

"This can leave the expert frustrated and can reduce the ability of the pair to complete the task." [2]

And the Winner Is . . .

Aside from the longer-term learning benefits, it seems that the most beneficial form of pairing is with two programmers of roughly the same ability. It's more likely that the pair will be on the same wavelength and will spend less time disagreeing over things that probably don't matter that much. Unfortunately, when you consider that 50% of all programmers are below average, it becomes obvious that mixed-ability pairing is likely to be the norm. This highlights the problem that teams of mixed abilities are almost unavoidable. Pair programming makes the issue unavoidable by forcing these people to code together on the same program.

In a non–pair-programming project, the problem is handled effectively through other more natural practices, such as team leading, code and design reviews, occasional (voluntary) pair programming, mentoring, design documents, and so on. With almost all of the problems described in this article, it's up to the coach to catch and deal with them as promptly as possible. This places a lot of responsibility on the coach (almost as much as the on-site customer!)

Design Documents Reduce Reliance on Pair Programming

Design documents provide a record of design decisions. This makes them particularly helpful for novice programmers to explore the thinking behind the design, as described by the more experienced senior programmers. If the team is becoming lost in a sea of changed minds and refactorings, the design document often helps to remind the team members of why they originally decided to do something in a particular way. There's usually a pretty good reason.

We discuss the role of documentation in software projects (and how it can lessen the need for pair programming) in *Extreme Programming Refactored* Chapter 7 [1].

And More Problems

Chapter 7 of PPI (titled "Problems, Problems") discusses several problems with pair programming. We briefly discuss some of these problems here. Although the authors of PPI do offer some practical advice to overcome or help prevent these problems, the proposed solutions either result in high maintenance or rely idealistically on the programmers being constantly aware of all the problems (with advice such as "Just proceed a bit more cautiously"). One problem is that of rushing. Because pairs rotate often, they might rush to finish a task before it's time to separate. The advice given in Chapter 7 of PPI is as follows:

"If a task must roll over to another pairing session, the task must roll over to another pairing session! Slow down, and do it right together." [2]

The coach would need to be particularly vigilant to spot this problem recurring, because pairs rotate so often. If the problem happens a lot, it may be because the tasks are too big (another direct consequence—evidence of the circle of snakes unraveling. To counter this, the team may need to spend more time planning or designing, or change its process for estimating stories or tasks). Another problem, which we suspect would particularly manifest in teams that publicly laud themselves as "the best team on the face of the Earth," (such as the original XP team that worked on the infamous C3 project) is that of overconfidence:

"There may be a feeling that a pair can do no wrong. If you're working together, you might convince yourself that whatever you do together must be right. Remain cautious and careful!" [2]

The problem of overconfidence would need to be watched for carefully by the coach, who should be aware of this type of problem. She would then need to be able to watch out for the telltale signs and be prepared to act on them when she catches pairs reassuring each other into writing bad code. "Well, I suppose it will do for now—we can refactor it later!" is the typical start of a slippery slope. Another problem is that it's human nature for people to want to be in control, at least of their immediate surroundings:

"New folks should specifically be paired with mentoring types, lest they feel unwelcome or frustrated in the hands of a partner who wants to make only personal progress. This mentor must also give up control and allow the less skilled team member to drive most of the time.

When the mentor is directing most of the activity, it's better for the trainee to be typing and not just listening. The student might not be assertive enough to ask for the keyboard." [2]

This is, of course, an idealistic approach. As we discover in Rich Camden's "Voice of eXPerience" account [1], being in control of the keyboard is the preferred option for most people. Rich wasn't offered the keyboard once in 5 months (that's not to say that he didn't get to type, but no one actually offered to relinquish control). If the other person doesn't speak up, he's not going to be offered the keyboard. As the previous quote suggests, this is particularly a problem with inexperienced programmers being allocated an experienced partner. Everybody likes to be the driver, to be in control.

Watch Out, There's a Snake Under the Desk!

The problems we just described must all be watched for and quickly fixed before they lead to other problems. This is a lot of problems associated with one XP practice, all waiting to slip and catch the unwary coach, who must be especially vigilant.

Pair programming by itself can be a beneficial practice, but in an XP project its problems are much more acute because (as we discuss in Chapter 3 of *Extreme Programming Refactored*) so much else in XP relies so heavily on its correct and consistent execution throughout the project.

References:

[1] Matt Stephens and Doug Rosenberg, "Extreme Programming Refactored: The Case Against XP", Berkeley, CA: Apress, 2003

[2] Laurie Williams and Robert Kessler, "Pair Programming Illuminated", New York, NY: Addison-Wesley, 2002

ASP Tools for Software Development

Franco Martinig, Editor of Methods & Tools,
www.methodsandtools.com

What is an Applications Services Provider?

Once upon a time, when computers where so expensive that not all companies could afford them, most of the processing was performed by specialised firms that will rent computer time and services. Some of these organisations have a long history of providing such services like for instance ADP. The development of Internet and Web-based applications has offered these firms a new infrastructure to offer theirs services and these companies are known now as Applications Services Provider (ASP). ASPs provide mainly software applications for the basic business functions of a company: human resources, accounting or customer relations, but there are also some organisations targeting the software development domain.

The ASP model was presented at the end of last century as a major trend in the software market. This situation is confirmed when you research on the Net about ASPs. You will find a lot of material dated from the years 2000 and 2001 with impressive growth rate predictions from the traditional research firms. In 1999, Dataquest predicted that ASP revenues would top $22 billion by 2003. In 2001, the Gartner Group estimated ASP revenues at $2.1 billion and predicted that this number would climb to $7.3 billion in 2003. These were the days where you have to be in B2B segment, before the burst of the Internet bubble reduced the anticipated growth rate. Many companies operating in this area have been considered cautiously as their survival capability was questioned.

Even if there have been some costly failures, like for instance Pandesic (a joint venture between Intel and SAP), this market is still alive. According to IDC, companies spent more than $2.3 billion in 2002 for software delivered as services. Selling software as services is still an interesting idea for vendors that prefer a continuous revenue stream to variable licence deals. Software vendors like Oracle or SAP and systems integrators like EDS or IBM are also playing a larger role in this market. Siebel recently bought hosted CRM vendor UpShot for $70 million, even if it has already its own service, Siebel CRM OnDemand, in association with IBM. Peoplesoft added last month to its hosted software offerings the applications acquired through JD Edwards buyout. The company announced having about 60 customers using its hosted services.

In a worldwide study of the ASP Industry Consortium conducted in 2001, only 8% of the respondents were using an ASP application. Other surveys performed in 2000 and 2001 showed that between 26% and 31% of US businesses were using an ASP. The applications mostly used in 2001 were communication (e-mail, groupware) 34%, finance 25%, e-commerce (21%) and CRM (19%). In 2000, 90% of ASP users said that the service level agreement provided by their suppliers meets their needs.

In the business functions market, ASPs can offer third-party "branded" solutions like Oracle, Peoplesoft or SAP products, but in the software development segment, companies offer mainly their own solution. Some tools are provided only on a web-hosted basis. Other companies have developed web-based tools and offer hosting. Hosting is either managed by the tool vendor or by another firm specialised in web hosting. There are commercial services offered on a monthly/project/user fee, but you will also find some services available for free, either completely or on a limited functionality basis. In this article we will mainly focus on companies that offer externally hosted software development tools on a commercial basis.

ASP tools for software development target currently primarily functions that request collaboration or centralisation of data, like project management and bug tracking for instance. We can however expect that in near future more external services could be available for activities like code validation, code refactoring or language translation. These are the kind of activities than could use a tool on a rental basis.

The current market trend

There are many different views on the current trend for the ASP market. Here are how some vendors that are or have been active in the ASP market see it, sorted in ascending order of positive view.

"Hosted solutions will continue to struggle because they are hard to sell and administer, "
Simon Galbraith, Red Gate Software

"Most companies prefer to run the software themselves as there are security concerns that they are worried about. A big problem we have is that we are not an established brand, and some people have problems with trusting the intellectual property with us. Therefore the uptake on hosted solutions is a lot less."
Andrew Thomas, Inobyte Ltd

"It seems to be the case that there are legitimate reasons to want either solution. Given that the different needs are driven more by overall IT strategy (more control with in-house solutions, more flexibility and less maintenance with hosted ones) rather than by specific technologies, it seems likely this separation will continue. Non-browser-based applications, by comparison, must either come to use the Internet in the same way that browser-based applications do or die out because of their disadvantages in terms of collaboration and availability.
Luc Reid, Vermont Software Testing Group

"For a variety of reasons, some companies still prefer to implement in-house solutions. Over the past few years, we have seen a steady increase in new customers who are looking for a short-term hosted solution which can be implemented while they continue their evaluation of in-house enterprise-wide solutions"
Krishen Kota, AdminiTrack.com

"On a wide scale, hosted software is continually gaining traction. However, there are companies for which it makes sense for them to host their own software. It is generally a balance between the business case and the comfort level of the IT department. Very often, a happy medium can be worked out. "
Mark Phillips, Standpipe Studios, L.L.C..

"There has been an interesting uptick in client signups with the service over the past 3 quarters. I am currently attributing this spike in activity to the many technical people who have been out of work for 6-12 months. Instead of finding new jobs with established companies, I believe that they are now starting up their own software companies. This effect has been especially noticeable in the United States, where the economy has been depressed for the past 2-3 years. As new start-ups, these small companies can ill afford to spend the time and money necessary to maintain their own source code control system. SourceHosting.Net fits their budget, so I believe we will help a large number of these early-stage start-ups during the initial product development phase. Even after that initial phase is complete, our clients typically stay with us because of the

cost savings compared to hosting a source code repository internally and dedicating time and resources to maintaining it."
Greg Larkin, SourceHosting.Net

"At first, we expected only smaller companies and start-ups to use the ExtraView hosted solution rather that the in-house solution. We assumed that teams with smaller budgets would be inclined to use a pay-as-you-go plan in order to preserve cash flow. However, we found that it was the larger companies that preferred to cut server, database and maintenance costs and go with a secure hosted solution. Larger companies understood that their data is more secure on our collocation servers and were comfortable with our hosted security, reliability, and availability."
Michael Stebbins, Sesame.com

"At the present time IT community is shifting dramatically toward hosted solutions. As Internet infrastructure will grow further, this process will accelerate."
Val Karmazin, eBUGtrack.com

As you can see the answers differ sometimes considerably. This difference can also be attributed to the type of services offered by the ASP. I think that two current trends can favourably influence the ASP market for software development tools:

- The growing importance of geographically dispersed teams and collaboration between different organisations. As Internet becomes the place to share information and tools, hosted solutions provide the advantage of neutrality.
- The benefit seen by management in ready-to-use short-term solutions. ASP solutions could be readily usable and their monthly fees allow a precise management of direct costs.

Working with an ASP

Working with an ASP could gives specific advantages, but also carries some risks. Here some of the differences between externally hosted tools and an internally managed solution

Specific advantages of ASP

- Faster evolution of the product: instead of having to distribute patches or upgrades, the supplier can work directly on the centralised product.
- Easier cost identification: all direct costs related to the tool are on your ASP bill.
- Unique channel for all application support issues: your vendor cannot argue that the tool doesn't work because you have installed it with an OS or a database version that is not supported.
- Formally negotiated service level agreement: you can set application availability levels and support response-time objectives... and discuss penalties in cases these objectives are not achieved.
- Smaller initial investment: ASP products are often billed with a monthly fee payment scheme. You don't have to make an important up-front purchase investment.
- Neutral solution for multi-organisation projects: if you collaborate with external organisations, an ASP provides a neutral ground to work with. It will also eliminate the problem of having outsiders accessing your information system to share one of your tools. Finally it can also be abandoned after the project completion if it is not needed.
- Trial period for new software: if the vendor offer to sell or host its solutions, using the hosted solution initially can allow to seriously test the software and the vendor support before buying it and installing it internally.

Risks associated to ASP

- External storage of data: your data is not on your hardware and you have less control on its management and who can access it.
- Greater dependability on the supplier: if there is a serious problem with your ASP company (or the company your tool supplier use to host its services), you will not be in the situation where you have an application without support like internally managed apps. Access to your application and your data can be reduced or suppressed.
- Sharing resources: processing power, bandwidth and support have to be shared between each ASP's customers that could have their own priorities, needs and agenda. If the number of customers increase, performance could suffer. Technical problems could also arise from many instance of the same application running on the same machine.
- Internet related availability risks: even if things work usually fine, accessing net servers is always subject to specific risks like viruses or denial of services attacks.

Some question that you can ask when you choose an ASP

- Is the hosting managed internally or performed by a third-party hosting company?
- What are the security features of the application?
- How is your data handled and what are the means you have for accessing it?
- Are special measures (availability of source code for instance) provided in case of difficulties of the supplier?
- What are the metrics for measuring the service level?
- How much can I customise the application to satisfy my particular needs?
- Would it be possible to bring the application internally again? Is there another exit strategy?
- What are the staffing levels and the technical expertise of the supplier's staff?

The data problem

Security of the data is the most important potential issue for companies using an ASP and vendors have put in place different measures to answer these fears:

- Extended security administration
- Encrypted data communication
- Regular back-up
- Functions to export data
- Availability levels guaranteed in SLA (Service Level Agreements)

The overall message from ASP to prospective customers is that their data will be handled as good or better than in their internal data centre, because they are specialised in managing web-hosted applications. Many ASP offer a trial period, it could be wise to try before you buy.

What can you get?

You will find on Internet four major areas where web-hosted software development tools are available:

- Bug and defect tracking
- Project management/collaboration
- Performance/monitoring/testing/checking of web applications
- Source hosting/configuration management

Bug and defect tracking

Bug and defect tracking is one of the main areas for software development tools ASP. This can be explained by the development of many web-based tools in this area in recent years. External hosting of these tools was a natural extension for the development companies. This is also a function closely related to customer care (CRM) applications and external (Web) access could be part of the requirements. There is also a tendency to have multiple teams working on the same project, either to minimise costs or to find external competencies. Finally, the open source movement has completely integrated these activities, the main tool being provided by Bugzilla.org.

Available services:

- Track components and versions, incidents, bugs or issues
- Configurable workflow-based, alternate, multiple, and client approval policies
- Time-based escalation of defect priority
- Automatic defect assignment routing
- User-defined entry forms, fields, alerts, and notifications
- Configurable email notification schemes
- Attach files notes, documents, and links to items
- Defect change history and audit trail

Project management and collaboration

Collaboration is a main issue of distributed teams since the Lotus Notes era of specific collaboration software. The widespread diffusion of the Internet has changed the vision of the market, but the needs remain the same. Many ASPs offer services related to distributed software project management.

Available services:

- Planning: Gantt, WBS Charts, critical path analysis
- Calendars
- Tasks
- Issue management
- Risk management
- Resource management
- Time, billing, cost and rate management
- Discussions forum management
- Files and document management
- Security
- MS Project integration
- Office software integration
- Email notifications and alerts

- Project dashboard reports
- Project analysis

Testing web applications

If you develop a web-based application, it is not always easy to test internally your web site. There are so many web standards and particularities (different browsers for instance) that it can be helpful to be assisted by dedicated external test teams or software. After production release, external monitoring can help to monitor how a possible customer in Berlin can access your web site in LA as the database gets loaded with data.

Available services:

- Download and verify pages
- Error diagnostics
- Performance tracking
- Screenshots of errors
- Contact escalation levels
- Unlimited mobile alerts
- Interactive reports
- Performance of Web transactions
- Transaction availability
- Changes in Web site content
- Secure page performance
- Redirection sequences
- CGI queries
- Authentication

Source hosting and configuration management

This is a strange area of the web-hosted software tools market. This kind of application is now widely adopted by all open source projects, but the market for commercial projects seems more underground and it could be difficult to find suppliers in this area. There is still reticence from companies to let intellectual property like software code in the hands of third party. Some providers in this area are simply providing a hosted version and support of the open source CVS tool (http://www.cvshome.org/)

Available services:

- Importing and exporting files
- Merge of concurrent modifications on same files with conflict warnings
- Versioning
- User management

Conclusion

In these days, the availability of software through the Web is a considered as natural, even if this trend is still relatively weak in the area of software management and development tools. Security and data availability are still major issues when deciding to use a web-hosted tool versus an internally managed solution. As confidence will increase that hosting companies can provide adequate answers to these concerns and a good level of support, I think that web-hosted software development tools will become a naturally accepted alternative solution to internally managed tools. The tendency to charge internal IT infrastructure costs on specific projects will also improve the comparison between internal management and externally hosting of applications.

For some companies, the most important trend is toward web-based applications. Suppliers develop products to adhere to this architecture, so you can choose to install them internally or use them as externally hosted solutions. For the vendors, this is a nice way to play on both markets. For the customers, this could be a safer way to buy ASP services, because it could provide an exit strategy with internalisation.

References

Specific software development ASP sites

dmoz.org/Computers/Internet/Web_Design_and_Development/Hosted_Components_and_Services/Site_Management/Monitoring	Listing of web testing and monitoring ASP
dmoz.org/Computers/Software/Configuration_Management/Bug_Tracking/Hosted	Listing of bug tracking ASP
www.web-based-software.com	Directory of software development related ASP

General ASP sites

directory.google.com/Top/Computers/Software/Business/E-Commerce/Business-to-Business/Application-Service-Providers	Google reference for ASP info
www.allaboutasp.org	Industry group. CompTIA's Software Services Group Formerly known as the ASP Industry Consortium (ASPIC), CompTIA's Software Services Group is a global advocacy group
www.applicationserviceproviders.com	ASP portal
www.asperado.com	ASP portal
www.aspnews.com	ASP portal
www.aspstreet.com	ASP portal
www.hostingtech.com	Hostingtech magazine
www.theaspexchange.com	ASP portal
www.webharbor.com	ASP portal
www.nwfusion.com/research/asp.html	ASP news and research

Acknowledgements

Although I assume complete responsibility of the above text, I would like to thank the following persons for their contribution to the development of this article

Person	Company	Web site
Krishen Kota	AdminiTrack.com	www.adminitrack.com
Andrew Thomas	Inobyte Ltd	www.inobyte.com
Ian Harrison	MJI Consulting	www.mjiteamworks.com
Simon Galbraith	Red Gate Software	www.red-gate.com
Michael Stebbins	Sesame Technology	www.sesame.com
Val Karmazin	SkyeyTech, Inc	www.eBUGtrack.com
Greg Larkin	SourceHosting.Net, LLC	www.sourcehosting.net
Mark Phillips	Standpipe Studios, L.L.C..	www.vertabase.com
Luc Reid	Vermont Software Testing Group	www.software-testers.com/dragonfly

METHODS & TOOLS is published by **Martinig & Associates,** Rue des Marronniers 25,
CH-1800 Vevey, Switzerland Tel. +41 21 922 13 00 Fax +41 21 921 23 53 www.martinig.ch
Editor: Franco Martinig; e-mail franco@martinig.ch ISSN 1023-4918
Free subscription: http://www.methodsandtools.com/forms/submt.php