THE BLACKWELL ENCYCLOPEDIA OF MANAGEMENT

SECOND EDITION

MANAGEMENT INFORMATION SYSTEMS

Edited by
Gordon B. Davis
Carlson School of Management,
University of Minnesota

THE BLACKWELL ENCYCLOPEDIA OF MANAGEMENT

SECOND EDITION

Encyclopedia Editor: Cary L. Cooper
Advisory Editors: Chris Argyris and William H. Starbuck

Volume I: *Accounting*
Edited by Colin Clubb (and A. Rashad Abdel-Khalik)

Volume II: *Business Ethics*
Edited by Patricia H. Werhane and R. Edward Freeman

Volume III: *Entrepreneurship*
Edited by Michael A. Hitt and R. Duane Ireland

Volume IV: *Finance*
Edited by Ian Garrett (and Dean Paxson and Douglas Wood)

Volume V: *Human Resource Management*
Edited by Susan Cartwright (and Lawrence H. Peters, Charles R. Greer, and Stuart A. Youngblood)

Volume VI: *International Management*
Edited by Jeanne McNett, Henry W. Lane, Martha L. Maznevski, Mark E. Mendenhall, and John O'Connell

Volume VII: *Management Information Systems*
Edited by Gordon B. Davis

Volume VIII: *Managerial Economics*
Edited by Robert E. McAuliffe

Volume IX: *Marketing*
Edited by Dale Littler

Volume X: *Operations Management*
Edited by Nigel Slack and Michael Lewis

Volume XI: *Organizational Behavior*
Edited by Nigel Nicholson, Pino G. Audia, and Madan M. Pillutla

Volume XII: *Strategic Management*
Edited by John McGee (and Derek F. Channon)

Index

Similar access is available from regulatory agencies in other countries.

ACM

The Association for Computing Machinery is the largest broad-based international computer and information system society (*see* ASSOCIATIONS AND SOCIETIES FOR INFORMATION SYSTEMS PROFESSIONALS).

ADA

ADA is a general-purpose programming language sponsored by the US Department of Defense. It is especially suited for the programming of large, long-lived systems with a need for ongoing maintenance. It supports modern programming structured techniques and concurrent processing.

Adobe Acrobat

see PORTABLE DATA FORMAT

agency theory applied to information systems

Soon Ang

Agency theory examines the contracts between a party (the principal) who delegates work to another (the agent). Agency relations become problematic when the principal and agent have conflicting goals and when it is difficult or costly for the principal to monitor the performance of the agent. When goals are incongruent, the agent is assumed to have a different set of incentive structures from the principal; the agent will consume perquisites out of the principal's resources and make suboptimal decisions. These activities produce efficiency losses to the principal. To counter these losses, the principal designs contracts to align the goals at the lowest possible costs. Costs can arise from providing incentives and from monitoring to insure that the agent is acting for the principal's interests.

Agency theory can offer insights for information systems. First, principals can design information systems to monitor the actions of agents. Electronic communication systems, electronic feedback systems, and electronic monitoring systems are examples of monitoring devices that can be implemented to insure that agent behaviors are aligned with principal interests.

Secondly, information systems professionals themselves often enter into agency relationships with other stakeholders in organizations and agency problems can arise. Important examples of such agency relationships include systems development, outsourcing (*see* IT OUTSOURCING), and end-user computing.

Systems Development

As principals, users often engage information system (IS) professionals as agents to develop information systems on their behalf. Due to a lack of understanding and knowledge of each other's domain, goal conflict may arise between the two parties. To reduce agency costs, one or both parties must try to narrow goal differences. IS professionals can invite users to participate more actively throughout the development life cycle. This gives the users more opportunities to verify requirements and insure that the final system is aligned with user needs. Further, users may request that the information system produce information-rich documentation so that monitoring is made easier and more readily available to users.

Outsourcing

In any outsourcing arrangement, the client company (principal) is usually motivated to shift its IS operations to external vendors who can carry out the work at the lowest possible cost. The vendor, on the other hand, may be looking for high profit in the arrangement. There is thus an economic goal conflict. To protect its interests, the client will increase its monitoring of the vendor. This can be achieved by requesting regular operational performance measures from the vendor, frequent meetings with the vendor to review progress of outstanding projects, and independent auditors to review benchmarks and

END-USER COMPUTING

Agency theory can help explain the dynamics of end-user computing. End users develop information systems themselves with little IS involvement. End-user computing, interpreted in agency theoretic terms, is a mechanism for reducing agency problems by eliminating the agency relationship between the user and IS professional.

agile development

Manjari Mehta and Dennis A. Adams

Agile development (AD) combines accepted principles of programming and management into a new discipline for software development for rapidly changing environments. This development approach contains several different methodologies, one of which is the most popular "extreme programming." AD is a balance between highly structured and sometimes bureaucratic development processes and those processes that are very unstructured and ad hoc. For instance, although AD follows planning and modeling techniques such as data flow diagrams and UNIFIED MODELING LANGUAGE (UML), it avoids excessive reliance on them which can hinder the developer from meeting dynamic customer requirements. AD employs documentation only to the extent that it helps developers and customers understand the code. Instead, it relies heavily on face-to-face information sharing.

AD is based on the following tenets:

1 *Interaction among individuals* is preferred over strict adherence to formal plans, processes, and tools. The most efficient and effective way to share information is to use face-to-face communication. As a people-centric approach, the working environment must be conducive to the personality and working styles of those developing the system.
2 *Customer collaboration* is preferred over negotiating contracts. Because an organization be better that managers communicate with the developers on a daily basis rather than employ a detailed requirements document. This insures that customer satisfaction is the highest priority and incremental changes of the working software quickly reflect customer's needs.
3 *Working software* is successful software. It is preferable to have working software with less documentation over non-working software with comprehensive documentation. The key is to keep it as simple as possible. Working software must be delivered frequently, ranging from two weeks to two months.
4 *Responding to change* is better than following a plan. Change is welcomed and even embraced late in the development process. Because software is developed and delivered in quick iterations, agile development enables the design team to quickly respond to new customer requests.

Some developers would say that AD flies in the face of the decades-old traditions of structured design and waterfall methodologies and that the frequent changes lead to scope creep, never-ending projects, budget overruns, and the like. Extreme programming is an example of an agile development methodology. The very name "extreme programming" would seem to indicate some sort of radical, perhaps careless method of systems development. Extreme programming, however, has a set of guiding principles that must be followed, lending structure to the methodology. These principles are interconnected and support one another.

1 *Paired programming*: A pair of programmers work together on a single workstation. One developer interacts with the workstation and the other explores, thinks one step ahead, and analyzes the current logic. Each reviews and supports the other and they only work together on the current task. After it is complete, they disband and circulate among the other teams. This is done at least once a day.
2 *Refactoring*: Refactoring (Fowler, 1999) is the process of rewriting existing code with the goal of making it easier to understand and more robust. Refactoring is done to a small