61

IS/IT Project Management: The Quest for Flexibility and Agility

61.1 Introduction

Providing technological capabilities to an organization in support of ongoing or new business initiatives is a core responsibility of the information systems (IS) function. IS/IT project management is therefore a key capability since the delivery of technological capabilities—from understanding user requirements to acquiring a technological solution to implementing the system—often falls to project teams led by an IS/IT project manager. IS/IT project management involves understanding the requirements of a project, developing a plan to deliver a product or service, and implementing the plan. IS/IT project managers are tasked with delivering a particular technological capability (e.g., an information system) to satisfy scope and quality requirements within specific time, budget, and resource constraints.

The context of managing IS/IT projects has changed dramatically over time. When computers were first introduced into organizations, it was common to develop custom information systems in-house. For example, in the United States, the first administrative information system was a payroll system developed by General Electric. In 1954, General Electric purchased a Univac I computer to use in its new Major Appliance Division plant in Louisville, Kentucky. GE Corporate Accounting Services took primary responsibility for managing this project, which involved designing and programming the first payroll information system to be used initially by the Washer and Dryer Department.* As part of its plan to automate its production facilities and make Louisville a showcase location, GE decided to use the computer not only to process payrolls and other accounting applications but also for manufacturing control and planning.

* See http://www.softwarehistory.org for more information about the history of the software industry and Burton Grad’s first person description of his experiences in programming the first commercial information system at GE (http://www.softwarehistory.org/pdf/x-1stCommCompAppGE.pdf).
From the 1950s well into the 1970s and beyond, sourcing options for information systems were few, and most systems were built by internal IS staff. As the GE example illustrates, information systems were typically designed to meet specific needs, usually for one particular functional area and typically focused on an operational aspect of the business. If a development methodology was used, it was most likely the waterfall (Systems Development Life Cycle) approach. The waterfall process, based on functional decomposition, consumed considerable time and effort and produced a wealth of documentation, including design documents, sign-offs, and responsibility charts, all intended to hold IS professionals accountable for producing a well-designed, well-constructed, and maintainable system. Consistent with the waterfall approach to development, most project managers and developers commonly assumed that information requirements were known, could be elicited, and would be stable. Thus, the waterfall process provided IS/IT project managers with a template for planning and execution, with its series of steps performed in a sequential manner.

The context of computing started to change rapidly, however, beginning in the 1980s. The prevalence of the personal computer and end-user computing, followed by the rise of the Internet and web-based computing, provided firms with many more technical capabilities and options. The business environment was also changing. Firms found themselves in an increasingly global, dynamic, and, in some industries, a highly regulated environment. Global competition, customers, and suppliers became the norm rather than the exception for most firms. Successful firms learned to adapt and evolve in order to respond to ever-changing consumer demands and competitor actions in the marketplace, as well as shifting resource availability and costs along the global supply chain.

The changes in the business environment converged with the rapidly changing technology such that more and more firms today expect information systems to allow for rapid response to both anticipated and unanticipated change (Lee and Xia 2005, 2010; Lyytinen and Rose 2006; Maruping et al. 2009). The complexity, volatility, and uncertainty inherent in the business environment demand flexibility and agility in the design and delivery of information systems and, correspondingly, the management of IS/IT projects. Flexibility is defined as the ability to adapt and change (Elfatatry 2007, p. 36). Agility, in this context, refers to “the ability to sense and respond swiftly to technical changes and new business opportunities” (Lyytinen and Rose 2006, p. 183).

Satisfying a firm’s need for flexibility and agility in its computing environment encompasses various aspects related to information systems acquisition and delivery. Not only must information systems themselves deliver flexibility, but the development methodologies and project management approaches must privilege adaptability and nimble responses to change. Further, firms desire flexibility and agility when they are looking to acquire a new system, and no longer view internal custom development as their only option for sourcing an information system. The waterfall approach as a template for project management becomes less relevant in an environment characterized by change and uncertainty.

This chapter explores IS/IT project management in contemporary firms. We start by considering the underlying principles of IS/IT project management. This includes an overview of modern project management as well as a detailed examination of the context of IS/IT projects, with a particular focus on flexibility and agility. Next, we discuss approaches for managing highly uncertain and volatile projects as IS project managers and teams strive to deliver flexibility and agility. We then explore research opportunities in IS/IT project management. We end this chapter by offering a few concluding remarks.

### 61.2 Underlying Principles of IS/IT Project Management

Though individuals have been managing projects for centuries, project management, as a profession, is relatively young; it was in 1969 that the Project Management Institute (PMI), a nonprofit association, was established with a mission to foster the profession of project management. By 1990, there were 7500 members of the PMI; today, there are over 600,000 members in more than 185 countries.* The PMI

---

* See [http://www.pmi.org](http://www.pmi.org) for more details about the PMI.
IS/IT Project Management offers various certifications to practicing project managers, including the Certified Associate in Project Management (CAPM) and the Professional Project Manager (PMP) certifications.

As part of its mission, the PMI has identified and codified the project management body of knowledge (PMBOK), the ten knowledge areas that all project managers must master to achieve CAPM or PMP certification. The knowledge areas are scope, time, cost, quality, human resources, communication, risk, procurement, stakeholder, and integration management. Project managers make decisions and take actions related to each knowledge area in an overlapping fashion across project management “process groups”: project initiation, planning, execution, monitoring and controlling, and project closing. The PMBOK maps activities associated with each knowledge area into the process groups.

The efforts of the PMI have had a profound impact on professionalizing project management in IS/IT as well as in other disciplines. However, the practice of IS/IT project management has also been greatly influenced by the methodologies used to develop information systems. Methodologies provide templates for managing projects since they articulate activities and processes associated with developing and delivering information systems. As methodologies have evolved, so has IS/IT project management. Therefore, we next examine the information systems development (ISD) process and highlight the relationship between ISD approaches and IS/IT project management.

ISD is the process by which computer software is specified, written, tested, and implemented. The waterfall process, a conventional approach to ISD, involves planning, analysis, design, coding, testing, implementation, and maintenance (George 1999). However, for today’s firms, the types of systems being developed, as well as the way in which they are developed, have evolved considerably since the 1950s. Initially, firms generally built single-purpose, stand-alone systems for a particular department or functional unit. Today, we are more likely to find one of two scenarios. The first is the development of smaller systems, often tightly integrated with others. One example is adding a web-based front end to an existing legacy system to improve its usability. The second scenario is the development of very large, complex systems that are often global or inter-organizational in scope. In contemporary firms, the choice of methodology is also changing. The conventional waterfall approach to development is just one of many that may be used; alternatives include the spiral model and agile methods. Over time, project management has also evolved so that today it tends to be more formalized and less ad hoc.

Table 61.1 compares the characteristics of traditional systems development and project management with emerging trends found in contemporary firms.

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Traditional Practices</th>
<th>Contemporary and Emerging Trends</th>
</tr>
</thead>
<tbody>
<tr>
<td>System characteristics</td>
<td>Single purpose and stand alone</td>
<td>Smaller and tightly integrated or larger and spanning units</td>
</tr>
<tr>
<td>Design philosophy</td>
<td>Design a system for long-term use</td>
<td>Design a system for adaptability</td>
</tr>
<tr>
<td>Common development method</td>
<td>SDLC or waterfall</td>
<td>Spiral or agile methods</td>
</tr>
<tr>
<td>Methodology approach</td>
<td>Functional decomposition</td>
<td>Component-based design</td>
</tr>
<tr>
<td>Methodology assumptions</td>
<td>Predictability and longevity</td>
<td>Uncertainty and flexibility</td>
</tr>
<tr>
<td>Methodology principles</td>
<td>Heavyweight</td>
<td>Lightweight</td>
</tr>
<tr>
<td>Project management approaches</td>
<td>Ad hoc, individual processes</td>
<td>More formalized approaches</td>
</tr>
<tr>
<td>Role of user</td>
<td>Minimal or token user participation</td>
<td>Partner in deployment of IS</td>
</tr>
<tr>
<td>Project teams</td>
<td>Colocated</td>
<td>Geographically distributed</td>
</tr>
<tr>
<td>Managing project teams</td>
<td>Simplistic understanding and use of control and coordination mechanisms</td>
<td>More nuanced understanding and use of control and coordination mechanisms</td>
</tr>
</tbody>
</table>

© 2014 by Taylor & Francis Group, LLC
In today’s quest for flexible and agile systems, three aspects of managing IS/IT projects deserve particular attention: requirements determination, design considerations, and sourcing options. Successfully determining requirements is a crucial step in an IS/IT project, but it is notoriously difficult, particularly when requirements are ambiguous or fluctuating. In addition, developing systems that are adaptable and flexible can heighten the challenge of determining requirements. Similarly, the principles underlying the design of a technical solution have implications for the nature of the solution itself. Whether a system is adaptable over time is, in part, a result of how the system was initially designed. Finally, the way in which a system is sourced—whether it is custom developed or acquired—has implications in terms of flexibility and agility. The method chosen to develop a technical solution varies in its ability to accommodate change: some methods are constraining while others are better suited to handle changes and produce systems that are flexible and agile. Moreover, increasingly organizations are turning away from custom development and considering alternate sourcing options. Accordingly, IS/IT project managers need to understand the advantages and disadvantages of a variety of sourcing options in order to choose the approach that best fits the needs of the system and the organization. In the following sections, we explore requirements determination, design considerations, and sourcing options in some depth, with a focus on flexibility and agility.

61.2.1 Requirements Definition

Research over the last 25 years has repeatedly highlighted the many challenges associated with requirements determination, including ambiguous requirements, communication obstacles, difficulty in reaching consensus on global needs, and the thin spread of domain knowledge across stakeholders (Alvarez 2002; Curtis et al. 1988; Urquhart 2001; Walz et al. 1993). The challenges are compounded in contemporary firms where the business needs are highly volatile and requirements change as systems are developed in response to emerging business needs or increased understanding of a complex business environment (Kirsch and Haney 2006; Maruping et al. 2009). Moreover, needs and requirements may be largely unknown at the start of a project, and may instead emerge as the project unfolds. Dynamic and emergent requirements represent a considerable departure from prior years, when requirements were regarded as specifiable and stable.

IS/IT project managers may increasingly be called upon to develop systems with emergent requirements. For example, there is growing interest and investment in large-scale, multiparty information systems that span disciplines and institutions and are geared toward supporting collaboration among many stakeholders (Berman 2008; Levina 2005). An example of this type of project is a cyberinfrastructure project (or “cyber project”). Cyberinfrastructure refers to integrated information technologies (i.e., hardware, software, digital sensors, middleware, networks, and data components) that support scientific research activities (Berman 2008; Bietz et al. 2010; Edwards et al. 2009). A specific objective of cyberinfrastructure is to foster innovative research and discovery through the use of technologies that support distributed collaborative work among geographically dispersed researchers (de la Flor et al. 2010; Ribes and Lee 2010). An example of cyberinfrastructure is the George E. Brown, Jr. Network for Earthquake Engineering Simulation (or NEES), which was created to better understand earthquakes, their causes, and effects.* NEES provides researchers, scholars, and industry access to state-of-the-art technologies and resources, augmenting their ability to study complex questions about earthquakes, to predict when earthquakes will occur, and to provide guidance to governments and emergency responders.

A “cyber project” refers to the information technology (IT) development activities that design, build, integrate, test, and implement a specific cyberinfrastructure. Cyber projects tend to be large and complex: they involve stakeholders from industry, academia, and government with varying goals and requirements, and they often cost millions of dollars and require years to develop (Finholt and Birnholz 2006). Because cyberinfrastructure is meant to promote experimentation and discovery, the requirements are emergent and unpredictable. In fact, the project outcomes are often unknowable and

* For more information, see http://www.nees.org
unpredictable: as a cyber project unfolds over time, the vision and goals associated with the project will themselves evolve in response to the ongoing experimentation and discovery.

Though cyber projects may be an extreme example in terms of fluctuating and emergent requirements, they nevertheless are illustrative of the challenges associated with managing volatile requirements. IS/IT project managers must recognize volatility and learn to adapt the way in which these projects are managed (Maruping et al. 2009). With fluctuating requirements, there is increased need for IS/IT project managers to promote ongoing mutual learning between providers of solutions and customers. Some methods are designed to accommodate changing requirements. Prototyping, spiral development methodologies, and rapid application development (RAD) assume that requirements are difficult to elicit and encourage iteration between developer and customer. For example, consider a cyber project called Global Environment for Network Innovation (GENI). The goals of GENI are to develop a virtual laboratory at the frontiers of network science and engineering for exploring future internets at scale (see http://www.geni.net). Given these goals, the project faces a considerable amount of requirements uncertainty. To address this uncertainty, the project is managed using spiral development. Each spiral lasts 1 year and consists of projects that develop different networking technologies. After each spiral, the project manager evaluates the success and failure of the technologies and decides on the technology goals for the next spiral. Spiral development affords the opportunity to explore cutting-edge technologies and provides flexibility to discontinue less promising technologies while affording the ability to explore new technology opportunities. From a project management perspective, however, the spiral approach—particularly as it is implemented in the GENI cyber project—introduces considerable complexity into the management process as there are multiple moving parts to coordinate and oversee.

Newer agile methods also generally recognize the need for intensive user–developer interaction. Using eXtreme programming (XP), for example, customers write brief “user stories” explaining what a system needs to do, and developers estimate the time needed to implement that functionality. Later in the process, the developer receives more details from the customer in order to implement the story.* In the context of emergent requirements, it is also important for IS/IT project managers to recognize that customers will not be able to fully specify requirements at the start of a project (Curtis et al. 1988; Maruping et al. 2009; Urquhart 2001). A number of empirical studies have highlighted the role of learning and intense analyst–user interactions especially in the context of poorly understood requirements and ill-structured problems (Alvarez 2002; Boland 1978; Kirsch and Haney 2006; Lyytinen and Robey 1999).

### 61.2.2 Design Approaches

In the past, systems were built for long-term use. The need for flexible, adaptable systems was not paramount as business environments were relatively stable. Contemporary firms, though, face a completely different environment, one in which competitors, products, and regulations are changing quickly and often in unpredictable ways. IS/IT project managers must understand this changing environment and adapt accordingly.

One response to the need for flexibility and agility is a modular approach to designing and implementing systems, based on a specific plan or architecture. An architecture incorporates a high-level understanding of the requirements, the solution, the modules, subsystems, and interface, as well as a broad plan for deployment (Goodhue et al. 1992; McBride 2007). When systems are designed without an architecture, they often evolve haphazardly, are difficult to change, and ultimately fail to meet the needs of the business (McBride 2007). In contrast, when an architecture is in place and followed, systems are designed for change because they are built in modules using standardized parts, which facilitates future adaptations. Thus, an architecture and modular design provide firms with flexibility and agility to deal with highly dynamic requirements.

* See http://www.extremeprogramming.org for additional information.
Recent studies have provided insights into the role of modular architectures in system adaptability. In a qualitative study, Collins and Kirsch (1999) observed the use of modular, incremental design and deployment in response to a dynamic and turbulent business environment. For example, in one firm, a complex global system was designed in relatively small increments (typically 6 months or less project duration) for geographically dispersed business units to implement and adapt to local requirements as needed. Moreover, researchers have begun to document the positive impacts of modular design. For example, in a field study of system enhancement activity over a 20 year time period, Barry, Kemerer, and Slaughter (2006) found that greater use of a standard architecture in a system localizes changes to specific components. The findings demonstrate that a standard architecture increases the design integrity and stability of the system while at the same time accommodating change.

Model-driven architecture (MDA) is a relatively recent software design approach that can be adopted for the development of information systems. MDA provides guidelines for specifications, expressed as models, and was launched by the Object Management Group (OMG, http://www.omg.org/mda). It involves producing software code from abstract modeling diagrams (Frankel 2003). MDA can introduce flexibility and agility in the software development process by separating design from architecture. This separation allows the designer to address the functional (use case) requirements while the architecture provides the infrastructure through which nonfunctional requirements like scalability, reliability, and performance are addressed. Separating design from architecture allows developers to take advantage of the latest techniques and technologies for each, while providing an architecture that is robust to changes.

Another design approach that affords considerable flexibility and agility in software projects is component-based development (CBD). An individual software component is a package, service, or module that encapsulates a set of related functions (or data). CBD is a reuse-based approach to defining, implementing, and composing loosely coupled independent components into systems (Heineman and Councill 2001). This approach allows the ability to “plug and play” components into software architectures, and components play a central role in service-oriented architectures (SOA). A recent field study of CBD by Subramanyam, Ramasubbu, and Krishnan (2012) found that CBD approaches are associated with what the authors call “efficient flexibility” in the development of business information systems.

### 61.2.3 Sourcing Options

Achieving flexibility and agility goes beyond requirements determination and design considerations. It is also important that IS/IT project managers consider sourcing options. We first consider in-house development methodologies and then turn our attention to a variety of external sourcing options.

Custom in-house development is the traditional approach to acquiring an information system in which an organization’s IT staff develop and install a system designed to meet the unique needs of the organization. Custom in-house ISD allows for meeting highly specialized requirements, facilitates changes to firms’ business processes, and helps to build up personnel skills in the organization. It also provides firms with a certain kind of flexibility—that is, the flexibility to creatively solve business problems, and to develop a solution that fits the particular nuances of a firm’s business environment. Custom in-house development enables firms to continually adapt and enhance their information systems to meet unique needs.

However, custom in-house development can also be constraining in that it generally takes a long time to develop a software solution internally. Moreover, custom in-house ISD can tax an organization’s resources and may incur significant risk. It is not uncommon for ISD projects to be over budget, run over schedule, and not deliver the required functionality (Barki et al. 2001; Keil 1995). Some projects are abandoned, without developing any useful parts of a system. One recent example is the Virtual Case File (VCF). The VCF project was supposed to automate the FBI’s paper-based work environment and allow agents and intelligence analysts to share investigative information. After years of effort, the VCF’s contractor, Science Applications International Corp. (SAIC), in San Diego, delivered 700,000 lines...
of code; however, the software was largely unusable, and the FBI had to scrap the entire $170 million project and start over from scratch (Goldstein 2005).

Because the conventional waterfall approach is often seen as too cumbersome and time-consuming to be effective when flexibility, speed, and agility are paramount (Lyytinen and Rose 2006; Slaughter et al. 2006), firms are increasingly turning to agile methods, a set of practices intended to minimize unnecessary bureaucracy and maximize adaptability and responsiveness. The Agile Manifesto, composed by a group of “independent thinkers about software development” who advocate lightweight, agile methods for development, highlights four distinct value propositions:

1. Individuals and interactions over processes and tools
2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan

While XP and Scrum are the most widely known and used agile methods, there are a number of others, including the Rational Unified Process (RUP) and Dynamic Systems Development Method (DSDM) (Boehm and Turner 2003).†

Despite the advances in development methodologies, custom design and development often are risky, lengthy, and prone to failure (Barki et al. 2001). Thus, organizations often turn to other sourcing strategies. Today, there are many viable alternatives for acquiring IS that a firm and its IS/IT management can consider. These include packaged software, outsourcing, and offshoring. Each is explored as follows.

**Packaged software** is code written by a vendor for purchase by organizations and use “off the shelf.” The package provides common functionality and may allow for customization of certain features. Packaged software is available in a wide variety of systems, sizes, and prices. For example, Enterprise Resource Planning (ERP) systems are very large software packages that enable organizations to integrate business processes across functional areas. ERP packages can be contrasted with smaller, more special purpose software packages such as for payroll and office productivity. Compared with custom development, software packages offer numerous advantages. First and foremost, the software is already written and functioning, and this may save time and reduce the risk of unusable code. The software may be very costly to “customize” a package, even if the vendor allows it (Davis 1988). A larger issue is that the organization may have to significantly change how it does business in order to use the software (Elbanna 2010). Research on implementing software packages, such as ERP systems, suggests that the greater the “misfits” between the package and the organization’s needs, the greater the cost and effort of implementation and the higher the risk of failure (Soh et al. 2000). In terms of flexibility and agility, packaged software provides agility in acquisition, as the time to acquire the software is short (compared to in-house custom development), although implementation can still be complex and lengthy. However, the flexibility to change the software is less than for in-house developed systems as the organization must negotiate with the vendor to make the changes.

**Outsourcing** involves hiring an external vendor, contractor, or service provider to create or support an information system. Information systems outsourcing has been a viable sourcing strategy since the early days of computing. Organizations can outsource their entire IT function (total outsourcing) or can outsource some systems or processes (selective outsourcing). An increasing percentage of organizations are currently outsourcing or considering it.

*See [http://www.agilemanifesto.org](http://www.agilemanifesto.org) for additional information.
†The degree of agility embodied in any method can vary. See, for example, [http://www.ibm.com/developerworks/rational/library/edge/08/feb08/lines_barnes_holmes_ambler](http://www.ibm.com/developerworks/rational/library/edge/08/feb08/lines_barnes_holmes_ambler) for a discussion of how RUP can support an agile approach to development.
Outsourcing offers numerous advantages as a sourcing strategy. By outsourcing, organizations can tap into the potentially broader and deeper IT experience and skill base of a vendor (Lacity and Willcocks 1998). The vendor may be more cost-efficient in providing IT services due to economies of scale in servicing multiple firms (Levina and Ross 2003). Outsourcing also can afford an organization greater agility (Lacity et al. 1995); for example, in the early 1990s Xerox outsourced software maintenance so that Xerox’s IT staff could focus on new systems development. However, there are some important disadvantages of outsourcing (for a review, see Lacity et al. 2009). Generally, by hiring an external party to develop the information system, the client can lose control over the development process as well as expertise and can actually have less flexibility in making adaptations to the system. There is also a risk that confidential or strategic information can be compromised. In addition, it is important for the client organization to have capable staff in-house to actively manage the outsourcing relationship with the vendor.

Offshoring is a recent sourcing strategy that can be used by organizations for in-house software development or, more commonly, in outsourcing arrangements. It is a methodology by which an organization exports ISD to another country via offshore development centers (e.g., Target has established an offshore IT development center in Bangalore, India, to complement its IT unit in Minneapolis, Minnesota), captive units (i.e., a business unit of a firm that functions as an independent entity offshore but still maintains close ties with the parent company), outsourcing to locally based vendor (such as Infosys, Tata, or Wipro in India), or outsourcing to a global IT service provider (such as Accenture) that has delivery centers overseas. There are both “near” shore destinations (Canada and Mexico would be near shore offshore destinations for U.S. organizations) and “far” shore destinations (India, Russia, China, and the Philippines would be far shore offshore destinations for U.S. organizations).

Offshore systems development has distinct advantages. By conducting systems development in multiple locations around the globe, organizations can achieve faster development time since they can leverage a 24 × 7 development cycle (Carmel 1999). This can increase an organization’s flexibility (because of increased productivity in developing and adapting information systems) and an organization’s agility (e.g., by offshoring some projects to one location, IT staff at another location may become available for other projects). Offshoring also allows an organization to diversify its IT talent pool and reduce the risks involved in relying on a single market. Offshoring can facilitate localization of product software and provide proximity to target markets. Finally, offshoring may result in significant cost savings on IT development staff as IT professionals in some parts of the world receive significantly lower salaries. For example, an Indian IT professional costs 1/6 to 1/10 as much as an IT professional in the United States.* On the other hand, the costs of coordinating and communicating can significantly increase when systems development staff are distributed globally (Espinosa, Slaughter, Kraut, and Herbsleb 2007). Other potential issues include language problems, legal, accounting and cultural barriers, loss of control, confidentiality, security, liability, intellectual property, and increased vulnerability to political events (Carmel 1999).

61.3 Implications for Practice: Managing Projects for Flexibility and Agility

In general, strong project management and risk analysis, combined with effective communication and information gathering, help deal with complexity and uncertainty in systems (McBride 2007; Rai et al. 2009). In this section, we offer a number of specific recommendations for managing IS projects with an eye toward flexibility and agility.

* For more information about international salaries, see http://www.sourcingline.com/country-data/salaries-software-engineer-web-developer and http://www.payscale.com/research/IN/Job=Computer_Programmer/Salary#by_Years_Experience
It is important for project managers to recognize and consider the various sourcing options. If custom development is called for, managers must decide whether to use the traditional systems development life cycle, or an agile method. A well-understood business need and relatively stable requirements call for using the systems development life cycle. So too do mission-critical systems, or systems being designed for the long term. On the other hand, volatile, emerging, or uncertain requirements suggest the use of an agile method. If time is of the essence, an agile approach is more likely to deliver a system more quickly.

There are many viable alternatives to custom in-house ISD. An important and challenging decision facing managers is which sourcing strategy to select, given a particular project. Although sometimes the sourcing strategy is determined by political or other organizational factors, it may be helpful to compare the strategies along several dimensions in making the best choice. Table 61.2 shows the major IT sourcing alternatives arranged along dimensions including the nature of the business need, the extent of in-house experience and capabilities, and the time frame of the project (Dennis et al. 2005).

<table>
<thead>
<tr>
<th>Business need</th>
<th>Custom Develop with Waterfall Approach If:</th>
<th>Custom Develop with Agile Approach If:</th>
<th>Buy Package If:</th>
<th>Outsource If:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique or core/complex, and relatively stable requirements</td>
<td>Unique or core/complex, and relatively stable requirements</td>
<td>Common</td>
<td>Not core or complex</td>
<td></td>
</tr>
<tr>
<td>In-house experience</td>
<td>Have both functional and technical experience</td>
<td>Have functional experience but limited technical experience</td>
<td>No functional or technical experience</td>
<td></td>
</tr>
<tr>
<td>Project development skills</td>
<td>Want to build development skills in-house</td>
<td>Development skills are not strategic. Have support and integration skills in-house</td>
<td>Development and support skills are not strategic</td>
<td></td>
</tr>
<tr>
<td>Project management skills</td>
<td>Skilled manager of IS/IT development projects. Proven methods available</td>
<td>Manager skilled at coordinating with vendor</td>
<td>Skilled manager can negotiate with and manage outsourcer</td>
<td></td>
</tr>
<tr>
<td>Time frame to implement</td>
<td>Flexible</td>
<td>Short</td>
<td>Short or flexible</td>
<td></td>
</tr>
</tbody>
</table>

Project managers must proactively plan for ways to deal with volatility found in an evolving problem domain, customer expectations, and perhaps the technology itself. In the presence of volatile requirements, it is important to promote ongoing mutual learning between developers and customers. Some methods, such as prototyping or agile approaches, encourage learning and intense customer–developer interaction (Lee and Xia 2010; Maruping et al. 2009). Similarly, Beyer and Holtzblatt (1995) propose an “apprenticeship model” of development in which the analyst and client engage in intense communication, discovery, and learning. Another option is to use participatory design, which recognizes the key role of the user in systems development, and in which both stakeholders together develop an understanding of the problem and solution possibilities (Kyng 1991; Simonsen 2007). Intense and frequent interaction promotes effective requirements gathering and interface definition; this in turn helps stakeholders understand and learn to deal with fluctuating requirements (McBride 2007).

Another means of dealing with volatility is to plan to deliver software solutions iteratively, relying on frequent releases and intense customer interaction and feedback (Harris et al. 2007; McBride 2007). Frequent releases ensure customers have access to the new features as they are developed. As the customers use features, project managers and developers can note their reactions and receive their feedback, learning how to adapt the system as customer needs emerge, evolve, and crystallize. Customers too can develop a better understanding of what is needed in a system.

As noted earlier, a high-level architecture can also help ensure that systems are designed and implemented for flexibility and agility. An architecture provides a high-level understanding of the requirements, solution, modules, subsystems, interfaces, and deployment plan. This helps the project manager and development team deal with system complexity and volatility, and it provides the basis for future adaptability, as it encourages modularization and layering of components based on standardized parts (Barry et al. 2006; McBride 2007). It can be a challenge, however, to ensure that systems are deployed in accordance with an architecture. One tool project managers have is the set of control mechanisms they can put in place to enforce compliance with an architecture (Kirsch 1996, 1997). Such mechanisms include the articulation of specific policies and procedures, and monitoring adherence to them, as well as fostering consensus about the importance of an architecture. Well-designed control mechanisms can help align interests of relevant stakeholders and ensure compliance with policies and standard practices. Another approach to ensure compliance is to add a software architect to projects (McBride 2007). The software architect, as the solution expert, works collaboratively with the business manager and project manager to ensure an initial understanding of the problem domain, elicit high-level requirements, identify and validate unstated expectations (McBride 2007).

Managing projects for flexibility and agility suggests an increased reliance on informal approaches to control and coordination to supplement and augment formal mechanisms (Maruping et al. 2009). Liberal use of ad hoc or casual meetings, hallway conversations, and social events are important means for controlling project teams and coordinating project tasks (Kirsch et al. 2010). Though it is important to rely more on informal and emerging mechanisms, it is equally important to recognize that existing structures, relationships, and methods cannot be ignored (Kirsch 1997; Madsen et al. 2006) as they provide the foundation for an emergent process that is in part predictable. Managers must have or develop an ability to adapt when technological, business, or environmental conditions change (Madsen et al. 2006). For example, systems development methodologies should be adapted for the situation at hand, rather than followed in the same manner for different situations (Fitzgerald 1997; Kirsch 1997; Madsen et al. 2006; Slaughter et al. 2006).

Finally, managers can take advantage of distributed geographic location and expertise by utilizing global distributed teams. However, it is important to foster trusting relationships and take steps to overcome cultural, language, or time barriers (Collins and Kirsch 1999; Rai et al. 2009). Providing access to dispersed team members through the use of tools such as collaborative technologies and videoconferencing will encourage knowledge sharing and joint problem solving (Majchrzak et al. 2005).
61.4 IS/IT Project Management: Research Opportunities

The contemporary environment for IS/IT project management is complex, dynamic, and uncertain (Elbanna 2010; Lee and Xia 2010; Maruping et al. 2009). This calls for an approach for developing and managing information systems projects that is flexible and agile, but our understanding of how to provide flexibility and agility is limited. In this section, we highlight three areas of future research. Each area represents a key role for IS/IT project managers. The first concerns development and design issues. The second concerns managing a distributed project team. The third concerns controlling projects.

61.4.1 Design and Development Issues

Requirements determination is a lynchpin in the development process. Project managers must ensure that a process is in place that not only elicits requirements but is sensitive to changing requirements. Given the increased volatility and uncertainty of the requirements determination task, it is clear that developers need the ability “to effectively and efficiently respond to business and technology changes” (Lee and Xia 2005, p. 77). Research suggests that such flexibility may be hard to find, however. In their study of over 500 ISD projects, Lee and Xia (2005) found that the project teams responded more extensively to business changes than technology changes, but they were more efficient addressing the technology changes. Additional research is needed to better understand how to achieve client–analyst learning in an effective and efficient manner, how to balance the need for customer responsiveness against the need for a disciplined development process, and how to train developers and analysts to thrive in an environment that demands agility and flexibility.

Another means of providing flexibility and agility is an architecture that promotes a modular design via a component-based or service-based approach to design. These approaches are similar in that they can be classified as “requirements adaptation” approaches as opposed to “requirements anticipation” approaches (Elfatatry 2007). However, they differ in that the component-based approach is to assemble existing prefabricated modules into a system during design (before system run-time). A SOA allows system designers to offer services through published or known interfaces. This kind of approach relies on run-time binding of components: when services are invoked, providers are chosen. Thus, a service-based approach provides potentially even more flexibility than a component-based approach. However, a service-based approach is not without its limitations, notably, implementation inefficiency and execution overhead (Elfatatry 2007). Additional research is needed to understand the capabilities and limitations of service and component approaches for achieving flexibility and agility. For example, under what circumstances does it make sense for an organization to adopt either approach?

Finally, there is a need for much more research on the use and effectiveness of agile methods to guide IS/IT project managers in choice of methodologies, and in better understanding the role of IS/IT project managers in agile contexts (Ågerfalk, Fitzgerald and Slaughter 2009). Initial research is promising. For example, in their case study at Intel, Shannon et al. (2006) found that the use of agile practices resulted in reductions in code defect density by a factor of 7, and that longer-term projects (i.e., 6 month and 1 year duration) were delivered ahead of schedule. Unlike some scholars, Fitzgerald et al. (2006) demonstrate that agile methods “are not anti-method, and require an equally disciplined approach, and as much tailoring as any traditional method” (p. 212). A special issue of the Journal of Database Management on Agile Information Systems Development (Erickson et al. 2005) contains a number of conceptual and review articles and case studies of agile systems development approaches. For project management, agile methods provide an interesting challenge: it is not clear whether and how formal project management practices (such as earned value analysis, work breakdown structures, and critical path analysis) that are predicated on waterfall development approaches can be applied or adapted to an agile development project. Additional field-based or experimental studies on the adoption and efficacy of agile methods for developing information systems and for managing agile projects would complement this research and provide valuable insights.
61.4.2 Managing Distributed Teams

Historically, project teams were generally collocated, and the management of the effort was typically vested in a single IS stakeholder. However, globalization is the new reality of contemporary firms, which suggests that project stakeholders—managers, users, project team members—are likely to be globally distributed (Herbsleb and Mockus 2003). The use of global teams can provide organizations with flexibility and agility. As previously noted, firms can adopt a 24 × 7 development cycle, leverage distributed knowledge, and respond quickly to local needs (Carmel 1999; Collins and Kirsch 1999).

Managing dispersed teams presents additional challenges for IS/IT project managers, including challenges related to differences in language, time zone, customs, and government regulations (Carmel 1999; Kotlarsky and Oshri 2005; Tractinsky and Jarvenpaa 1995). Moreover, distributed teams can incur higher costs of coordinating, controlling, and communicating: numerous researchers have documented the difficulties and costs associated with geographically dispersed teams (Espinosa et al. 2007; Kirsch 2004; Rai et al. 2009). Rapport and knowledge sharing are essential for successful collaborations but can be difficult to achieve in globally distributed teams (Carmel 1999; Kirsch and Haney 2006). Though technology can help address some of these challenges—e.g., high-speed connections and collaborative tools such as e-mail and instant messaging—they do not necessarily address social needs (Kotlarsky and Oshri 2005), nor do they necessarily compensate for geographic distance (Collins and Kirsch 1999).

Researchers have identified a number of ways to mitigate problems associated with distributed teams. For example, trust among distributed team members can improve personal relationships and productivity (Rai et al. 2009). When team members have electronic access to each other, knowledge sharing and collaboration are encouraged (Majchrzak et al. 2005). Transactive memory, the knowledge possessed by group members along with individual awareness of who knows what in the group (Wegner 1987), has been found to improve project effectiveness because knowledge seekers can quickly find expertise in the group (Faraj and Sproull 2000). Further, social ties and knowledge sharing have been found to improve collaboration among distributed team members (Kotlarsky and Oshri 2005). Additional research is needed to better understand the challenges of managing dispersed project teams, and the effectiveness of various tools and techniques to overcome those challenges.

61.4.3 Exercising Control and Coordination

Successfully controlling and coordinating projects is essential for IS/IT project managers to deliver the technological capabilities sought by their organizations. Control refers to all attempts to motivate individuals to work in accordance with specific objectives (Kirsch 1996). Coordination integrates and links different parts or activities to accomplish a task (Van de Ven et al. 1976; Zmud 1980). Control and coordination, though distinct concepts, share a focus on motivating and integrating people and processes to accomplish particular goals. Mechanisms to do so include formal approaches—rules, policies, methodologies, and standards—as well as informal approaches—ad hoc communication, peer pressure, and common values and norms (Kirsch 1997; Zmud 1980).

In the past, it was not uncommon for IS/IT project managers to rely on a simplistic understanding of control and coordination processes and therefore to use one-size-fits-all mechanisms. That is, control was often viewed as a static and generic cybernetic process in which goals or standards could be precisely specified, progress could be accurately assessed against those objectives, and corrective action could be identified and prescribed (Kirsch 1997). Consistent with this view, managers also may have assumed that a development methodology, precisely specified, could be faithfully followed and produce the desired system. However, it is now recognized that development methodologies are templates or guides for action, rather than a prescription for precise behaviors (Fitzgerald 1998; Fitzgerald et al. 2006; Kirsch 1997; Madsen et al. 2006). Thus, a methodology is best viewed as a “method in action” (Fitzgerald 1997).

In contemporary settings, there is a need to adapt control strategies to particular types or phases of projects or to adopt control approaches appropriate for nonroutine work (Kirsch 2004). Adaptation is
particularly important in an environment of uncertainty and volatility. Cyber projects such as NEES and GENI, in which needs and goals are evolving and precise IS-related activities or behaviors can be difficult to articulate, exemplify this environment and present a rich context for studying the complexity of IS/IT project management.

Dealing with increased complexity, uncertainty, and ambiguity also calls for increased reliance on informal communication, coordination, and control (Herbsleb and Moitra 2001; Kirsch et al. 2010; Kotlarsky and Oshri 2005; Kraut and Streeter 1995). For example, the use of informal social or clan control mechanisms early in the project can promote learning and consensus among stakeholders (Kirsch 2004) by focusing attention on common values among project team members and negotiating a consensus on project goals. Research suggests a need for flexibility in control and coordination strategies, adapting them as a project progresses in response to emerging understanding, performance problems, or changes in the project environment (Choudhury and Sabherwal 2003; Kirsch 2004).

Moreover, it is apparent that multiple stakeholders control a project, thus suggesting that no one person completely controls what happens (Kirsch et al. 2010; Madsen et al. 2006). As control strategies evolves over the life of the project, the controller, i.e., the person exercising control, also changes (Kirsch 2004). At any one point in time, any particular manager may be both “in control” and “out of control” in the sense that he or she must grapple both with certainty (e.g., working in a stable environment with regular work, meetings, and patterns) and with uncertainty (e.g., an unpredictable environment with emergent action, conflict, and diversity), what has been termed the “paradox of control” (Madsen et al. 2006). However, it is not clear how managers grapple with a paradox of control. Additional research that examines the ebb and flow of control across project stakeholders over time would be extremely useful. In addition, there is a need for a more nuanced understanding of the types of control and coordination needed to achieve agility and flexibility, specifically the trade-off between too much and too little formal and informal control and coordination as a project unfolds over time.

61.5 Conclusions

Information systems are indispensable to firms, enabling their operational and strategic business processes. In recent years, dramatic changes in the business landscape, coupled with the expanding range and capability of available IT, brought the need for flexibility and agility in corporate computing to the fore. Today’s IS/IT project management must facilitate a firm’s ability to rapidly respond to competitive strikes and to quickly deploy their own strategic initiatives.

This chapter explored current and emerging trends in IS/IT project management. We discussed the meaning of flexibility and agility in the context of systems development and implementation projects, and highlighted the implications for managing IS projects. Flexibility and agility start with a plan, an architecture, or a blueprint to guide the development and deployment of systems. The architecture or blueprint provides the basis for adaptation by facilitating incremental, modularized development with standardized parts. Flexibility and agility can also be enhanced by using principles from agile development approaches that favor a lightweight, adaptable approach. The numerous sourcing options available to firms provide a great deal of flexibility and agility in system acquisition, as they allow firms the freedom to select an approach that best fits its needs. Firms are no longer constrained to the often high-risk, time-consuming method of developing systems internally. On the other hand, when the situation calls for it, internal development is feasible and can provide firms with flexibility in solving business problems or enabling strategic initiatives.

This chapter discussed the need for flexibility and agility in project management, with a particular focus on design decisions, the management of distributed teams, and on control and coordination of projects teams. Achieving flexibility and agility do not require eliminating the use of standards, rules, and procedures, but it does require judicious use of formal structures so that action is not unnecessarily constrained. Using informal control and coordination mechanisms to supplement formal mechanisms
provides a project manager with flexibility. The use of geographically distributed teams can also provide flexibility and agility, but it is not without its challenges.

It is clear that achieving flexibility and agility requires a supple and nimble mindset, as well as a number of specific principles and approaches. Additional research is needed to develop a better understanding of ways to achieve flexibility and agility, as well as ways to overcome barriers preventing developers, managers, and others from embracing flexible and agile approaches.

Acknowledgments

This chapter is based upon work supported by the National Science Foundation under Grant No. 0909611 and 0909833. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References


Boland, R.J., Jr. The process and product of system design, Management Science 24(9), 1978, 887–898.


Davis, G. To buy, build, or customize? Accounting Horizons, March 1988, 101–103.


Herbsleb, J.D. and Mockus, A. An empirical study of speed and communication in globally distributed software development, IEEE Transactions on Software Engineering 29(6), 2003, 1–14.


Keil, M. Pulling the plug: Software project management and the problem of project escalation, MIS Quarterly 19(4), December 1995, 421–447.


