

STRATEGIES FOR IMPROVING SYSTEMS DEVELOPMENT PROJECT SUCCESS

Christopher G. Jones, California State University, christopher.jones@csun.edu

Glen L. Gray, California State University, glen.gray@csun.edu

Anna H. Gold, Erasmus University, agold@rsm.nl

David W. Miller, California State University, david.miller@csun.edu

ABSTRACT

Sixteen years after the publication of the Standish Group's first CHAOS report in 1994, there is little cause for celebration. True, system development project (SDP) success rates have improved to 32% from the benchmark low of 16.2%; however, when 68% of projects are either cancelled or seriously challenged with regard to budget, schedule, or project scope, there is considerable room for improvement in the SDP process. This research examines the critical risk factors responsible for system development failure with an eye toward the role internal auditors could take in increasing the likelihood of SDP success. In this paper we provide an overview of our efforts to identify a relevant set of critical factors by synthesizing the voluminous practitioner and academic literature. From the hundreds of potential factors identified, we conclude with a preliminary list of 16 strategies for improving SDP success that are the subject of an ongoing investigation.

Keywords: System Development Project, Critical Success Factor (CSF), Critical Failure Factor (CFF), System Development Risk, Role of Internal Auditing

INTRODUCTION

The publication of the Standish Group's first biennial survey of IT project performance in 1994 revealed a staggering 31.1% of all U.S. system development projects (SDPs) ended in failure. Almost fifty-three percent (52.7%) were seriously challenged, either through budget overruns, missed deadlines, or feature sets that did not meet user requirements. Only 16.2% of SDPs were considered successful, coming in on budget and on time [32].

Fourteen years later the statistics were somewhat better but still cause for concern. In the 2008 CHAOS report, the Standish Group reported that SDP success rates had improved to 32% from the benchmark low of 16.2%. Outright failures declined to 24% and "challenged" projects fell to 44% [30]. This is an encouraging trend. However when 68% of SDPs are either cancelled or seriously over-budget, behind schedule, or short some requirements, there is

considerable room for improvement in the system development process. Not everyone accepts Standish Group's oft-quoted statistics on project resolution. Recently some academics have been challenging the Standish Group's research methodology [13, 19]. Even so, SDP success (defined as on-time, within budget, and including the promised feature set) is rare.

In 2002, U.S. Congress attempted to legislate better financial systems. Tucked within the language of the 2002 Sarbanes-Oxley Act (SOX), Congress mandated a stronger connection between financial information systems and internal controls [14]. Although SOX only applies directly to U.S. public companies, many privately held, non-for-profit, and foreign companies have voluntarily implemented SOX-like provisions. The impact on IT has been "significantly greater levels of auditing on process controls within IT governance" [21]. As a result, the role of internal auditors as members of the corporate governance team has changed radically, elevating their organizational stature from mere application control experts to a meta-control role over the SDP process itself [15]. This is an emerging role that is being shaped by the strengths internal auditors bring to the systems development process. These strengths include a holistic, organizational perspective; a business-value orientation; and a rich history of post-implementation reviews from which to draw best practices for process improvement.

This research explores strategies for improving the success rate of SDPs. Research methods included a literature review, a series of focus groups with internal auditors, and a survey of members of The Institute of Internal Auditors (IIA). The research is ongoing. In the interest of brevity, this paper focuses solely on one segment of the early stages of our research – a literature review of the critical factors responsible for SDP outcome.

FACTORS IN SYSTEMS DEVELOPMENT

The academic and practitioner IT literature is replete with case histories, analyses, and editorials regarding SDP failures. There is a separate, although less voluminous, literature on project successes. Over

time the literature has moved from a focus on project failure to identification and mitigation of system risks. We begin this literature review with a discussion of early research into critical factors. From there we examine later attempts by practitioners and academics to isolate critical SDP factors. We conclude this section with an analysis of the CHAOS report – perhaps the most widely-cited research on SDP outcomes. In the next section we explore academic research attempts to synthesize the critical factors into a complete set.

Early Research on Project Failure

As early as 1973, Morgan and Soden examined determinants of failed information systems projects. After studying ten unsuccessful projects, Morgan and Soden concluded that most failures were due (not surprisingly) to management’s inability to manage – that is plan, organize, and control [28].

Using a case study of a state planning agency, Schmitt and Kozar analyzed the events and problems leading to development of a land-use management information systems by an outside consultant that was completed but never used. According to Schmitt and Kozar’s 1978 paper, the land-use MIS was beset by a series of risk factors that created a degenerative error network that eventual lead to project failure. Risk factors included: (a) lack of systems analysis, (b) immaturity of the client’s decision making process, (c) excess trust placed in the outside consultant, (d) an ill-defined contract between agency and consulting firm, (e) IS developed outside the user organization, (f) no user involvement in data selection, (f) a single system approach rather than integration with existing systems, (g) non-aggregated data collected at too low a level of granularity, and (h) weak planning products and documentation [28].

According to Alter and Ginzberg’s 1978 article on managing uncertainty in MIS implementations, the top risks, identified through structured interviews with designers and users, were: (a) “designer lacking experience with similar systems, (b) nonexistent or unwilling users, (c) multiple users or designers, (d) turnover among users, designers or maintainers, (e) lack of support for system, (f) inability to specify the purpose or usage patterns in advance, (g) inability to predict and cushion impact on all parties, and (h) technical problems, cost-effectiveness issues” [1, p. 27].

Gordon Davis’ 1982 paper on requirements determination strategies listed three risks: (a) “existence and stability of a usable set of

requirements, (b) user’s ability to specify requirements, and (c) ability of analysts to elicit requirements and evaluate their correctness and completeness” [7, p. 20].

McFarlan’s 1982 portfolio approach to information systems also named three key risks: (a) “size in cost, time, staffing level, or number of affected parties, (b) familiarity of the project team and the IS organization with the target technologies, and (c) how well structured is the project task” [26, p. 250].

According to Boehm’s 1991 oft-cited article on software risk management, the top ten risk items according to a survey of experienced project managers were: “(1) personnel shortfalls, (2) unrealistic schedules and budgets, (3) developing the wrong functions and properties, (4) developing the wrong user interface, (5) gold-plating (i.e., unneeded features), (6) continuing stream of requirements changes, (7) shortfalls in externally furnished components, (8) shortfalls in externally performed tasks, (9) real-time performance shortfalls, and (10) straining computer-science capabilities” [3, p. 35].

Barki *et al.* identified five general risk factor categories: (a) “newness of the technology, (b) application size, (c) lack of expertise, (d) application complexity, and (e) organizational environment” [10, pp. 40 – 43].

Ropponen and Lyytinen examined risk-management practices of Finnish software project managers by analyzing 83 projects across a variety of organizations. Six risk categories were identified: (a) “scheduling and timing, (b) system functionality, (c) subcontracting, (d) requirement management, (e) resource usage and performance, and (f) personnel management” [10, pp. 41- 43].

Project Abandonment

Ewusi-Mensah and Przasnyski [11] analyzed failed systems development efforts to identify factors responsible for project abandonment. Senior IS executives at Fortune 500 companies were asked to complete a lengthy questionnaire; 82 (5.6%) responded. A factor analysis identified 12 dimensions across three categories—economic, technological and organizational (listed in order of importance)—(1) “escalating project costs and completion schedules, (2) lack of appropriate technical infrastructure and expertise, (3) actual project expenditures and duration below estimates, (4) technological inadequacies and shortcomings, (5) loss of critical personnel and management changes, (6) end-user acquiescence, (7)

management commitment and perceptions, (8) end-user conflicts and technical disagreements, (9) satisfy existing or emergent technology, (10) lack of funds, (11) discouraged end-user participation, and (12) consequence of merger/acquisition by another company” [11, p. 193].

In later (popularized) discussions of his research, Ewusi-Mensah [9] reduced the number of project abandonment factors to seven: (a) “unrealistic project goals and objectives, (b) inappropriate project-team composition, (c) project management and control problems, (d) inadequate technical know-how, (e) problematic technology base/infrastructure, (f) lack of executive support and commitment, and (g) cost overruns and schedule delays” [10, p. 43]. Ewusi-Mensah has since revised the list of abandonment factors for a book length examination of system development failures to include two new items: (a) “changing requirements and, (b) insufficient user commitment and involvement” [10, p. 43].

Updated Risk List

Using a modified Delphi survey approach of 41 project managers from three countries, Keil, Cule, Lyytinen and Schmidt [20], identified and ranked critical risk factors for software projects. The resulting “universal set of risk factors” (ordered by relative importance) included: (1) “lack of top management commitment to the project, (2) failure to gain user commitment, (3) misunderstanding the requirements, (4) lack of adequate user involvement, (5) failure to manage end user expectations, (6) changing scope/objectives, (7) lack of required knowledge/skills in the project personnel, (8) lack of frozen requirements, (9) introduction of new technology, (10) insufficient/inappropriate staffing, and (11) conflict between user departments” [20, p. 78].

For the most part, rankings were similar from country to country with the exception of “conflict between user departments” which was more important in Hong Kong and Finland than in the USA. When compared to Boehm’s [3] 1991 top-10 risk list discussed earlier, Keil *et al.*’s [20] list is broader and more up-to-date with less focus on execution factors under the project manager’s control. Boehm’s list was derived from his experience in the defense industry in the 1980s before the advent of distributed computing and off-shoring.

Outsourcing Risk

Based on a case study of the failure of the Bezeq-AMS Billing System, Natovich [27] discussed the additional systems development risk introduced by outsourcing. According to the study, vendor risks were unique to outsourced IT projects and by virtue of their contractual nature not applicable to in-house projects. Natovich defined vendor risk as “the risks that the client bears when contracting the project to an external vendor rather than conducting it in-house” [27, p. 410]. Such supplementary risks included: (a) adversarial relationships resulting in loss of trust between client and vendor, (b) de-escalation of vendor commitment when circumstances change, and (c) difficulty in severing contractual engagements in response to the threat of litigation.

Natovich faults Keil *et al.*’s [20] IT risk classification framework as too narrow, claiming that it only takes into consideration in-house development. He particularly finds problematic the case where the IT project has been outsourced and the vendor becomes the project manager, since the framework ignores the risks attendant in the vendor-client relationship.

An SEI View of Failure

Citing data from the 2000 Chaos Report, the Software Engineering Institute’s (SEI) Watts Humphrey [17] made the case that system development failure is project size dependent. Projects exceeding \$10 million in projected cost have no chance for success, whereas projects under \$750,000 have a 55% success rate. According to Humphrey, the historical reason for large-scale system development failure was due to lack of planning. The introduction of sound project management practices has improved project success rates but has not completely eliminated the unique challenges facing large projects.

CHAOS Report

The Standish Group, source for the project resolution statistics cited in the introduction to this paper, conducts biennial (once every two years) research on IT project performance using a combination of focus groups, surveys, and executive interviews. From 1994 through 2006 it had examined over 50,000 completed IT projects [16]. In the 1994 CHAOS report, the Standish Group identified ten key risk factors responsible for project failure. In order of ranking, with the most cited factor listed first, project impairment factors included: “(1) incomplete

requirements, (2) lack of user involvement, (3) lack of resources, (4) unrealistic expectations, (5) lack of executive support, (6) changing requirements and specifications, (7) lack of planning, (8) didn't need it any longer, (9) lack of IT management, and (10) technology illiteracy" [32, p. 6].

In *CHAOS 1994*, the Standish Group also identified a top ten list of critical factors responsible for project success (Table 1). Many of these factors are the semantic inversion of the failure factors. For example, "user involvement," the Standish Group's top ranked CSF, is linguistic reversal of "lack of user input" and "lack of user involvement." According to the 1994 CHAOS report, the key CSFs for project

success (listed in order of importance) were: "(1) user involvement, (2) executive management support, (3) clear statement of requirements, (4) proper planning, (5) realistic expectations, (6) smaller project milestones, (7) competent staff, (8) ownership, (9) clear vision and objectives, and (10) hard-working, focused staff" [32, p. 5] .

Except for some minor rewording, the top three CSFs have remained the same since the original 1994 report. Table 1 summarizes CHAOS critical success factors (CSFs) by survey year for information available through publicly available sources, as the CHAOS Report is proprietary.

Table 1 CHAOS Report Success Factors by Rank

Factor	1994	1998	2000	2002	2006	2008
User Involvement	1	1	2	1	1	1
Executive Management Support	2	2	1	2	2	2
Clear Statement of Requirements	3	6	7			
Proper Planning	4	8				
Realistic Expectations	5					
Smaller Project Milestones	6	5				
Competent Staff	7	7		10	8	8
Ownership	8	9				
Clear Vision and Objectives	9	3	4	4	3	3
Hard-Working, Focused Staff	10					
Project Management		4	3	3	6	7
Minimized Scope			5	5		
Standard Tools and Infrastructure			6	7	10	10
Formal Methodology			8	8	9	
Reliable Estimates			9	9		
Agile Requirements Process				6	5	6
Optimizing Scope/Optimization					4	5
Financial Management					7	
Emotional Maturity						4
Execution						9
Other		10	10			

CHAOS Challenged

Although the CHAOS report is the most oft-quoted source for statistics on project resolution (success, challenged, failure), lately some academics have challenged the Standish Group's research methodology [8]. Robert Glass, editor emeritus of Elsevier's *Journal of Systems and Software*, the publisher of the *Software Practitioner* newsletter, and respected contributor to the *Communications of the ACM*, recently questioned the extent of the project failure rates reported by the Standish Group and whether the data supported a real "software crisis" as portrayed in the CHAOS reports [12, 13].

Glass [13, p. 16] suggested that the Standish research is "biased towards reports of failure" because it tends

to focus primarily on organizations with failed or failing projects. According to Glass "software projects succeed far more often than they fail" because the era of the Computing Age "wouldn't be possible if we didn't have astoundingly successful software to make all those computers do the wonderful things they do" [12, p. 110]. The real failure rate, Glass [12] speculated, is closer to 10 or 15%, but he only has anecdotal support for his figures.

Jørgensen and Moløkken examined the cost overrun statistics reported over the years in the CHAOS report [19]. They specifically questioned the 1994 CHAOS statistic of an average 189% cost overrun on challenged projects. Using data from three academic studies on cost overruns, Jørgensen and Moløkken

estimated the real average is closer to 33%. Interestingly, the most recent cost overrun figures (54%) reported by the Standish Group [30] are closer to Jørgensen and Moløkken findings than the original 1994 CHAOS report. Jim Johnson, founder and chairman of the Standish Group, has responded to the criticisms of his firm's data with assurances that the research is conducted appropriately but that specifics are proprietary [18].

CRITICAL FACTOR FRAMEWORKS

A fundamental purpose of this research was to identify the factors that contribute to the success of SDPs. A review of the professional and academic literature revealed an abundance of factors cited as contributing to SDP success and failure. Reported factors varied across project context, system size, development methodology, role of survey participant or field informant, and a host of other parameters. The number of critical factors was so great (literally in the hundreds), that complete analysis of the factor set was not practically feasible. As a consequence, in order to proceed with the research, the sizeable number of factors described in the literature had to be synthesized into a manageable taxonomy of items deemed critical to the success of SDPs. To do this, we investigated existing taxonomies for grouping critical factors.

Risk Management Approach

Lyytinen, Mathiassen, and Ropponen [26] explored risk management as a promising approach to reduce system development failure. In their study, Lyytinen *et al.* adopted Harold Leavitt's socio-technical model as a framework for analyzing risk management and risk resolution. Under Leavitt's model, organizational change is governed by a multivariate system of four interacting components – task, structure, actor, and technology. As applied to system development, Lyytinen *et al.* characterized *tasks* as system goals; *structure* as project organization; *actors* as users, managers, developers; and, *technology* as the technical platform and development tools. Unaddressed risk was seen as a disequilibrium (i.e., *variation* in socio-technical theory) that was to be managed until balance was regained.

The relationship between the four components is as important as the individual component itself. For example, risks associated with actor-technology interdependencies may be created by the mismatch of people and technology. Using the modified socio-technical model, Lyytinen *et al.* analyzed four classical approaches to software risk management

and reduction. The result was a list of risk items and related risk resolution techniques classified into each of the four socio-technical components or the four component-component interactions. Of particular interest were the risk item lists from the four studies mentioned in the previous section (Boehm; Davis; Alter and Ginzberg; and McFarlan).

Risk Categorization Taxonomy-Part I

As part of Keil *et al.*'s [20] research, discussed earlier, in which 11 critical risk factors were identified, Kiel and his colleagues constructed a multi-dimensional framework for categorizing software risks. In this typology, risks fell into four quadrants along two dimensions: (a) perceived level of project manager control and, (b) perceived relative importance of risk. Quadrant 1 "Customer Mandate," for example, included risks with high relative importance over which the project manager has little control. Examples include *lack of top management commitment* or *inadequate user involvement*. Risk mitigation for this quadrant requires "relationship management, trust-building, and political skills" [20, p. 80]. Keil *et al.* argued that the framework simplified risk management by clustering CFFs into a reduced set of classes subject to mitigation through broad strategic initiatives.

Risk Categorization Taxonomy-Part II

Cule, Schmidt, Lyytinen and Keil [6] extended their previous research on risk framework by detailing further the risk categorization taxonomy and the behavioral model for risk mitigation. The four quadrants remained the same but the names were changed. Quadrant 1 Customer Mandate became *Client*. Quadrant 2 Scope and Requirements became *Self*. Quadrant 3 Execution morphed into *Task*. Quadrant 4 remained as *Environment* (see Table 2).

In addition, the quadrants were now grouped into two categories – inside risks and outside risks. For each risk category, there was a one-word label for the associated behavioral strategy. To mitigate Client risks, project managers should "relate" by managing the relationship with those who fund and those who will use the system. For Self, the strategic term is "assess." Here Cule *et al.* [6] suggested, once again, the importance of independent auditors in gauging the project manager's abilities, capabilities, and knowledge regarding IS development. Evaluation could be performed on a peer basis by another experienced project manager or through the exercise of conducting a process maturity assessment using

Table 2 Risk items grouped by type and category (Cule, Schmidt, Lyytinen, & Keil, 2000, p. 68)

INSIDE RISKS	
Self	Task
<ul style="list-style-type: none"> • Not Managing Change Properly • Lack of Effective Project Management Skills • Lack of Effective Project Management Methodology • Improper Definition of Roles and Responsibilities • Misunderstanding the Requirements • Poor or Non-Existent Control • Poor Risk Management • Choosing the Wrong Development Strategy • Lack of “People Skills” in Project Leadership • Project Not Based on Sound Business Case • No Planning or Inadequate Planning 	<ul style="list-style-type: none"> • Bad Estimation • Lack of Effective Development Process/Methodology • Trying New Development Method/Technology During Important Project • Lack of Required Knowledge/Skills In the Project Personnel • Poor Team Relationships: • Insufficient Staffing • Excessive Use of Outside Consultants • Lack of Available Skilled Personnel • Introduction of New Technology • Stability of Technical Architecture • Multi-Vendor Projects Complicate Dependencies
OUTSIDE RISKS	
Client	Environment
<ul style="list-style-type: none"> • Lack of Top Management Commitment to the Project • Failure to Gain User Commitment • Conflict Between User Departments • Failure to Get Project Plan Approval From all Parties • Failure to Manage End User Expectations • Lack of Adequate User Involvement • Lack of Cooperation from Users • Failure to Identify All Stakeholders • Growing Sophistication of Users Leads to Higher Expectations • Managing Multiple Relationships with Stakeholders • Lack of Appropriate experience of the User Representatives • Unclear/Misunderstood Scope/Objectives • Number of Organizational Units Involved • Lack of Frozen Requirements • New and/or Unfamiliar Subject Matter for Both Users and Developers • Under Funding of Development • Under Funding of Maintenance • “All or Nothing” • Artificial Deadlines 	<ul style="list-style-type: none"> • A Climate of Change in the Business and Organizational Environment that Create Instability in the Project • Mismatch Between Company Culture and Required business Process Changes Needed for New System • Project that Are Intended to Fail • Unstable Corporate Environment • Change in Ownership or Senior Management • Changing Scope/Objectives • “Preemption” of Project by higher Priority Project • Staffing Volatility • External Dependencies Not Met • Lack of Control Over Consultants, Vendors, and Sub-Contractors

Note: The 11 “universal set of risk factors” (Keil, Cule, Lyytinen, & Schmidt, 1998) are highlighted in bold.

externally available tools from the Software Engineering Institute. Another approach would involve benchmarking against other projects and organizations. Task risks could be “controlled” using tools provided in project management texts. Environment risks could be “monitored” in order to keep abreast of the infrequent but unpredictable changes that can derail development efforts.

Perhaps the biggest contribution of the article was a comprehensive list of the 53 risk items mentioned in Keil *et al.* [20] but never delineated in their entirety. Cule and his colleagues [6] categorized these 53 risks into the four major risk quadrants based on their own project management experiences. Table 2 displays the risks identified in the Keil *et al.*[20] Delphi study grouped by internal and external risk and categorized into the four risk quadrants. The 11 “universal set of risk factors” have been bolded. It is interesting to note that 7 (63%) of the top 11 CFFs were considered beyond the control of the project manager.

Risk Groups

In “Identifying Software Project Risks: An International Delphi Study,” Schmidt *et al.* [28] provided a complete exposition of the methodology behind the research leading to the comprehensive 53-item list of systems development risks (Table 2) listed above. As part of the study, Schmidt and his colleagues used three Delphi panels to validate a taxonomy based on the source of the risks. The result was a list of 14 risk groups. As part of their research, Schmidt *et al.* categorized the 53 risk items by risk group and provided a detailed description of each individual risk item.

One of the interesting findings from this cross-cultural study was that relative risk varies by country. While there seemed to be general agreement on some of the major risks, there were important differences regarding the lesser project risks based on cultural dimensions such as *individualism* or *uncertainty*

avoidance. The top system development risks for U.S. project managers were: (1) lack of top management commitment to the project, (2) misunderstanding the requirements, (3) not managing change properly, (4) failure to gain user commitment, (5) lack of effective project management skills, (6) lack of adequate user involvement, (7) failure to manage end-user expectations, (8) lack of effective project management methodology, (9) unclear/misunderstood scope/objectives, (10) changing scope/objectives, (11) lack of adequate user involvement, (12) introduction of new technology, (13) insufficient/inappropriate staffing, and (14) lack of frozen requirements [28, p. 21].

Failure Factors Categorization

In Yeo's [34] survey, conducted in 2000, of close to 100 respondents associated with a major project failure in Singapore, failure factors were grouped into three organizational categories based largely on Checkland and Holwell's [5] Processes for Organization Meanings (POM) model. Under this taxonomy, factors dealing with culture, leadership, and organizational issues are classified as "context-driven" and are shaped by corporate management and users. Factors related to technology and business process, the "what" and the "how," are labeled as "content-driven" and are the purview of IT professionals.

Matters related to strategic formulation and change management were categorized as "process-driven" and were seen to be largely under the influence of the project manager. The top five process-driven CFFs were: "(1) underestimate of timeline, (2) weak definition of requirements and scope, (3) inadequate project risk analysis, (4) incorrect assumptions regarding risk analysis, and (5) ambiguous business needs and unclear vision" [34, p. 245].

Top context-driven issues were: (1) lack of user involvement and inputs from the onset, (2) top down management style, (3) poor internal communication, (4) absence of an influential champion and change agent, and (5) reactive and not pro-active in dealing with problems" [34, p. 245]. Finally, for content-driven issues, the top CFFs were: (1) consultant/vendor underestimated the project scope and complexity, (2) incomplete specifications when project started, (3) inappropriate choice of software, (4) changes in design specifications late in [the] project, and (5) involve[s] high degree of customization [sic] in application" [34, p. 245].

Beyond Risk Checklists

Building on prior research on software risk factors, Wallace, Keil, and Rai [33] performed a cluster analysis on survey results from a convenience sample of project managers ($n = 507$) who were members of the Information Systems Special Interest Group (ISSIG) of the Project Management Institute (PMI). Respondents were asked to rate 44 risk variables and 9 performance measures using a seven-point Likert scale. Cluster analysis revealed six risk dimensions: (a) team, (b) organizational environment, (c) requirements, (d) planning and control, (e) user, and (f) complexity. The performance measures were clustered into two groups: (a) product, and (b) process. *Product* performance measured user satisfaction and product quality; *process* measured budget and schedule constraints. A complete list of risk items categorized by risk dimension is found in the study.

Wallace et al.'s risk categorization overlaps to some extent earlier cluster analyses such as Barki et al.'s [2] research involving 35 risk variables and 5 dimensions discussed earlier. Barki's work focused more on technological and scale risks and lumps users in with developers under the heading "expertise"; Wallace separated users from team members and broke down development risk into two categories: (a) requirements, and (b) planning and control. Both studies share two general risk categories: (a) organizational environment and (b) complexity.

As part of the Wallace [33] study, the six risk dimensions were analyzed across project risk levels to determine if risks were differentiated by project type. Low risk projects were found to have high complexity risk. High risk projects had high requirements, planning, control, and organization risks. The study also examined influence of project scope, systems sourcing practices, and strategic orientation on the six risk dimensions. Sourcing arrangements were found to affect team risk and planning and control risk; strategic orientation of the project was found to impact project complexity risk. Project scope influenced all six risk dimensions.

Approaches to Software Risk Management

The Software Engineering Institute [4] has developed a comprehensive risk mitigation strategy based on an exhaustive study of risk factors. Carr *et al.*[4] found three major categories of systems development risk: (a) product engineering, (b) development environment, and (c) program constraints. Product

engineering included such factors as requirements, design, code, unit and integration test, and engineering specialties. Development environment consisted of considerations for development process and system, management process and methods, and work environment. Program constraints referred to resources, contract, and program interfaces. The SEI risk taxonomy has a decidedly computer science feel to it.

SYNTHESIS OF LITERATURE REVIEW

A survey of the taxonomy literature associated with critical factors revealed a variety of approaches for categorization depending on the contextual perspective (e.g., process, product, culture, or organization). For this research, our context was organizational/cultural in nature, based on an “outsider’s view from the inside,” specifically, that of the internal auditor. To supplement the literature review, we conducted focus groups with internal auditors to identify SDP success factors over which internal audit had influence. In all, over 38 individual factors emerged from the focus group discussions. Two factors, however, stood out as particularly germane to this research (Systems Development Process Monitoring; System Interoperability), owing

to what focus group participants felt was the internal auditor’s holistic perspective on how projects evolve during the SDP life cycle. A full discussion of the focus groups is beyond the scope of this paper but will be included in the final research monograph.

A summary of the various factors and taxonomies derived from the literature review and focus groups served as a starting point for an initial attempt at deriving a reduced factor set. A naturalistic inquiry approach [23] was used. Through many hours of group and individual analysis, duplicate removal, and factor consolidation, the preliminary inventory of items was revised into a final list of factors that were to be used in the survey phase of our research. As part of the refinement process, a taxonomy co-evolved that met our categorization scheme criteria of drawing factors from both the literature and the focus groups without creating category overlaps or type-subtype dependencies. Five broad categories emerged from the factor classification analysis: (1) People, (2) Organization, (3) Project, (4) Project Management, and (5) Externalities. The final list of key factors (see Table 3) is presented below, grouped by taxonomic category, and described using the language that was employed in the questionnaire.

Table 3 SDP Critical Success Factors by Category

Category	Factor	Definition
<i>People</i>	Executive support	Key executives providing alignment with business strategy, as well as financial, schedule, emotional, and conflict resolution support.
	Project personnel	Acquiring, retaining, and managing skilled project personnel in the face of turnover and other personnel hurdles.
	Project management expertise	Project leaders possessing basic project management skills and practices.
	Conflict management	Influencing the emotions and actions of project stakeholders to minimize the impact of ambition, arrogance, ignorance, passive-aggressiveness, fear of change, and deceit.
<i>Organization</i>	User involvement	Involving business and IT users with key consensus-building, decision-making, and information-gathering processes.
	Business alignment	Ensuring stakeholders understand the core value of the project and how it aligns with business strategy.
<i>Project</i>	System requirements	Defining system objectives and scope. Capturing user requirements and incorporating them into the system specification.
	System interoperability	Designing the system to work with other systems and functional areas.
<i>Project Management</i>	System development methodology	Defining a set of process-based techniques that provide a road map on when, how, and what events should occur in what order.
	Tools and infrastructure	Providing project infrastructure tools that enable management of tasks, resources, requirements, change, risks, vendors, user acceptance, and quality management.
	Agile optimization	Using iterative development and optimization processes to avoid unnecessary features and ensure critical features are included.
	Change management	Monitoring and controlling modifications to system requirements.
	Monitoring of system development process	Methodically reviewing project milestones for schedule, scope, and budget targets.
	Quality assurance	Governing project quality through definitive acceptance criteria, timely testing, issue identification, and resolution.
<i>Externalities</i>	Financial management	Managing financial resources, accounting for project budget/costs, and demonstrating the value of the project.
	Vendor relationship management	Actively monitoring and controlling contracts with vendors/consultants.

CONCLUSION, LIMITATIONS, AND FUTURE RESEARCH

A review of the academic and professional literature found a seemingly boundless number of factors that authors claim are associated with successful (or failed) SDPs. However, on closer inspection, it was clear there were many duplicates and near duplicates (i.e., different phrasing but with same meaning). In an effort to reduce the factor set to a manageable level, we developed a taxonomy based on the literature review and the unique perspective of our research. During the factor identification-consolidation-categorization process we were mindful that synthesizing success and failure factors into a single factor could be problematic. We were cognizant that *success* was not necessarily the opposite of *failure*. Inadequate resource allocation, for instance, could almost certainly doom an SDP, but adequate resources would not necessarily assure success. For this reason, we carefully considered the unique contribution each factor made, regardless of its possible linkage to failure or success. Nevertheless, there is a chance, that during this factor consolidation process, critical factors may have been eliminated or marginalizing through aggregation with related concepts. Our future research will compare the internal audit rankings for critical success factor importance with the general IT community. We do not anticipate the content of the factor list to vary from what was synthesized from the literature and focus groups. We do, however, expect CSF rankings to reflect the unique perspective of IA practitioners.

ACKNOWLEDGMENTS

This paper is part of a larger project sponsored by the Institute of Internal Auditors Research Foundation (IIARF). Research involved three lines of inquiry: (1) factors in SDP success and failure (the subject of this paper), (2) measures for gauging SDP success, and (3) the evolving role of the internal auditor in monitoring SDPs. The literature review phase was followed by a series of focus groups with internal auditors, and a survey of The IIA membership to determine what role internal auditors can play in increasing the success rate of SDPs. A full discussion of the remaining two dimensions of the literature review, the focus groups, and survey results is planned for a forthcoming IIARF research monograph. Opinions expressed in this paper are those of the authors and not necessarily those of the IIARF or The Institute of Internal Auditors.

REFERENCES

1. Alter, S. & Ginzberg, M. (1978). Managing uncertainty in MIS implementation. *Sloan Management Review*, 20 (1), 23-31.
2. Barki, H., Rivard, S., & Talbot, J. (1993). Toward an assessment of software development risk. *Journal of Management Information Systems*, 10 (2), 203-225.
3. Boehm, B. W. (1991). Software risk management: Principles and practices. *IEEE Software*, 8 (1), 32-41.
4. Carr, M. J., Konda, S. L., Monarch, I., Ulrich, F. C., & Walker, C. F. (1993). *Taxonomy-Based Risk Identification*. Carnegie Mellon University. Pittsburgh, PA: Software Engineering Institute.
5. Checkland, P. & Holwell, S. (1998). *Information, Systems and Information Systems*. Chichester, UK: John Wiley.
6. Cule, P., Schmidt, R., Lyytinen, K., & Keil, M. (2000). Strategies for heading off IS project failure. *Information Systems Management*, 17 (2), 65-73.
7. Davis, G. B. (1982). Strategies for information requirements determination. *IBM Systems Journal*, 21 (1), 4-30.
8. Eveleens, J. L. & Verhoef, C. (2010). The rise and fall of the CHAOS Report figures. *IEEE Software*, 27 (1), 30-36.
9. Ewusi-Mensah, K. (1997). Critical issues in abandoned information systems development projects. *Communications of the ACM*, 40 (9), 74-80.
10. Ewusi-Mensah, K. (2003). *Software Development Failures: Anatomy of Abandoned Projects*. Cambridge, MA: The MIT Press.
11. Ewusi-Mensah, K. & Przasnyski, Z. H. (1994). Factors contributing to the abandonment of information systems development projects. *Journal of Information Technology*, 9 (3), 185-201.
12. Glass, R. L. (2005). IT failure rate - 70% or 10-15%? *IEEE Software*, 22 (3), 112, 110-111.
13. Glass, R. L. (2006). The Standish Report: Does it really describe a software crisis? *Communications of the ACM*, 49 (8), 15-16.
14. Gray, G. L. (2004). *Changing Internal Audit Practices in the New Paradigm: The SOX Environment*. Altamonte Springs, FL: The IIA Research Foundation.
15. Gray, G. L. (2008). *Then and Now: Expectations and Reality of Sarbanes-Oxley*. Altamonte Springs, FL: The IIA Research Foundation.

16. Hartmann, D. (2006, August 25). *Interview: Jim Johnson of the Standish Group*. Retrieved June 30, 2010, from InfoQ: <http://www.infoq.com/news/Interview-Johnson-Standish-CHAOS>
17. Humphrey, W. S. (2005, March). Why big software projects fail: The 12 key questions. *Cross Talk: The Journal of Defense Software Engineering*, 1-8.
18. Johnson, J. (2006, September 12). *Reader Comments - Re: Two questions to Jim Johnson of the Standish Group*. Retrieved June 30, 2008, from InfoQ: <http://www.infoq.com/news/Interview-Johnson-Standish-CHAOS>
19. Jørgensen, M. & Moløkken-Østfold, K. (2006). How large are software cost overruns? A review of the 1994 CHAOS report. *Information and Software Technology*, 48 (4), 297-301.
20. Keil, M., Cule, P. E., Lyytinen, K., & Schmidt, R. C. (1998). A framework for identifying software project risks. *Communications of the ACM*, 41 (11), 76-83.
21. Leih, M. J. (2006). The impact of the Sarbanes-Oxley Act on IT. *Journal of Information Technology Theory and Application*, 8 (3), 13-30.
22. Levinson, M. (2009, June 18). *Recession causes rising IT project failure rates*. Retrieved June 30, 2010, from CIO.com: http://www.cio.com/article/495306/Recession_Causes_Rising_IT_Project_Failure_Rates_
23. Lincoln, Y. S. & Guba, E. G. (1985). *Naturalistic Inquiry*. Beverly Hills: SAGE Publications.
24. Lindstrom, D. R. (1993). Five ways to destroy a development project. *IEEE Software*, 10 (5), 55-58.
25. Lyytinen, K. & Hirschheim, R. (1988). Information failure—A survey and classification of the empirical literature. In P. Zorkoczy (Ed.), *Oxford Surveys in Information Technology* (pp. 257-309). New York, NY, USA: Oxford University Press, Inc.
26. Lyytinen, K., Mathiassen, L., & Ropponen, J. (1998). Attention shaping and software risk—A categorical analysis of four classical risk management approaches. *Information Systems Research*, 9 (3), 233-255.
27. Natovich, J. (2003). Vendor related risks in IT development: A chronology of an outsourced project failure. *Technology Analysis & Strategic Management*, 15 (4), 409-418.
28. Schmidt, R., Lyytinen, K., Keil, M., & Cule, P. (2001). Identifying software project risks: An international Delphi study. *Journal of Management Information Systems*, 17 (4), 5-36.
29. Schmitt, J. W. & Kozar, K. A. (1978). Management's role in information systems development failures: A case study. *MIS Quarterly*, 2 (2), 7-16.
30. The Standish Group. (2009). *CHAOS Summary 2009: The 10 Laws of Chaos*. Boston, MA: The Standish Group International, Inc.
31. The Standish Group. (1998). *CHAOS: A Recipe for Success*. Boston, MA: The Standish Group International, Inc.
32. The Standish Group. (1995). *The CHAOS Report*. Boston, MA: The Standish Group International, Inc.
33. Wallace, L., Keil, M., & Rai, A. (2004). Understanding software project risk: A cluster analysis. *Information & Management*, 42, 115-125.
34. Yeo, K. (2002). Critical failure factors in information system projects. *International Journal of Project Management*, 20, 241-246.