American Journal of Software Engineering and Applications
2019; 8(1): 8-17
http://www.sciencepublishinggroup.com/j/ajsea
doi: 10.11648/j.ajsea.20190801.12
ISSN: 2327-2473 (Print); ISSN: 2327-249X (Online)

Completing Information Technology Projects as Scheduled and on Time

Charles William Butler[1], Gary Lowell Richardson[2]

[1]College of Business, Colorado State University, Fort Collins, United States of America
[2]College of Technology, University of Houston, Houston, United States of America

Email address:
Charles.butler@colostate.edu (C. W. Butler),grichardson@uh.edu (G. L. Richardson)

To cite this article:
Charles William Butler, Gary Lowell Richardson. Completing Information Technology Projects as Scheduled and on Time. *American Journal of Software Engineering and Applications.* Vol. 8, No. 1, 2019, pp. 8-17. doi: 10.11648/j.ajsea.20190801.12

Received: January 28, 2019; **Accepted**: April 8, 2019; **Published**: July 26, 2019

Abstract: Even though Information Technology (IT) software development projects exhibit a checkered history of poor scheduling and late delivery, emerging project traits are promoting successful implementation and reduced risk of failure. IT project management, requirements gathering and management, user involvement, organizational alignment, development methodology, quality assurance and testing process maturity are integrating together to create a culture of rising success and efficient project implementation. Organization alignment, project management, business and technology architectures, and organizational change management domains are integrated into a successful software development framework.

Keywords: Organizational Alignment, IT Project Management, Software Development Methodology, Artifact Repository, Controlled Processes

1. Introduction

When candy giant Hershey Foods former CEO and Chairman Kenneth L. Wolfe told Wall Street analysts during a conference call in September 1999 that the company was having problems with its new order-taking and distribution computer system—a $112 million combination of software from ERP maker SAP, CRM provider Siebel and supply chain software from Manugistics—he didn't offer any details. He did say, however, that the problems were going to keep Hershey from delivering $100 million worth of Kisses and Jolly Ranchers for Halloween that year. *—Hershey Foods VP and CIO George David in an Aug. 29, 2002, news release*

Is the Hershey story newsworthy? Absolutely, but hardly unusual! Missed deadlines, cost overruns, and/or failure to meet requirements are the rule rather than the exception for information technology (IT) project development efforts.

Computerworld detailed ten large project disasters totaling in the 100's of millions of dollars in lost revenues, cost over runs and so on. [1] One firm, FoxMyer Drugs, reportedly went into bankruptcy because of the implementation failure of its enterprise resource planning system. For both FoxMyer and Hershey, the culprits were a large third-party software package. History suggests that system risks further escalate when custom design and coding is added to the equation.

In 2001, the Standish Group, who sponsored a project measurement survey for IT development projects, reported: [6].

i. Average schedule overrun was 163%
ii. Average cost overrun was 145%
iii. Actual functionality compared to plan was 67%

Only 26% of projects surveyed were judged to be a success; the lost value from marginal and failed projects was estimated at $75 billion. These outcomes are a dismal report on the state of project management. The failures are rarely attributable to the underlying technology but rather to other factors. Recent research has identified a pattern of manageable factors that lead to both project success and failure. The bottom line is that effective management is usually the predominant missing element in systems development.

In 2014, the Standish Group, who sponsored another project measurement survey for software development projects, reported: [4]

i. $250B expenditure on IT application development of 175,000 projects
ii. Average project cost for:
iii. A large company of $2,322,000

iv. A medium company of $1,331,000
v. A small company of $434,000
vi. 31.1% were cancelled before they are completed
vii. 52.7% cost 189% of their original cost
viii. Failure to delivery originally proposed features and functions:
   a) For large companies, approximately 42% of proposed requirements were implemented
   b) For small companies, 78% of projects delivered 74% of proposed features and functions

The facts are that project execution from 2001 to 2014 has not come a long way. As shown in Table 1, failed projects decreased by 30% from 23% to 16.2%. As the same time, the percent of challenged projects increased by 7% from 49% to 52.7%. The percent of succeeded projects increased by 11% from 28% to 31.1%.

Table 1. *Distribution of Project Resolution.*

Resolution	2001	2014
Failed	23%	16.2%
Challenged	49%	52.7%
Succeeded	28%	31.1%
Total	100%	100%

To better understand some of the technical and managerial issues surrounding effective project management, it seems worthy to explore how the current state was realized and to examine a general roadmap for the future. With that in mind, the section below describes the evolutionary path that has established today's IT project culture (1960s to current). From this base strategies are introduced that can lead to improved project success.

2. Evolution of IT Project Management

Systems development during the 1960s and the 1970s was a simplified and informal process. Manually drawn flowcharts were often used to define user requirements, though even less rigorous approaches were common. Programmers, working with little formal management oversight, translated system requirements to code. Schedules and cost estimates were created using informal methods. The results, as measured in missed budgets, slipped schedules, and failed systems reflected a lack of rigorous management. Information technologists survived because enterprise management did not understand that the process could be better executed. In fact, a basic vision for a proven process did not exist. System building was viewed as a "black art" and IT professionals were usually excused as doing the best they could in difficult circumstances. Developers survived, not because of their efficiency or effectiveness, but because their products were important to the business and they were often the only source for IT-enabled solutions.

In the 1980s, islands of success began to emerge for structuring elements of the system development process. Tools and techniques evolved for eliciting requirements, estimating tasks, designing code and databases, and for testing modules and systems. Unfortunately, rapid changes in technology, particularly the migration of hardware platforms, first to minicomputers and then to desktops, lessened the impact of these advances. New architectural models coupled with changes in development tools and methodologies aided in creating a chaotic development environment. Development productivity tools were oversold to management, their impact on productivity largely unproven. The value of the resulting new systems, ones increasingly viewed as strategic necessities, continued to capture management's attention. Progressively, attention turned to development costs, cost overruns, and the associated business costs of systems that failed to meet objectives.

In the 1990s organizations continued down the path of piecemeal project management approaches, with some organizations now differentiating themselves by their ability to consistently develop successful systems. For most firms, the normal result of a system development project continued to be cost and schedule overruns and less than planned functionality. At this point the recognition of project failures had expanded to greater public visibility as exemplified by the Standish Group study. Also, the underlying technology platforms, both for development and production platforms, continued to churn. The first half of the decade was largely devoted to migrating to the newly fashionable client-server architectures. Emergence of the internet and Y2K conversions are the most noticeable activities in the latter half of the decade. Collectively, these events overshadowed the recognition of underlying project management issues. At this point some firms, burned by project disasters or losing confidence in their development units, turned to third party outsourcing as an easy, if sometimes shortsighted, solution. As the 21st century was entered, the stage was set to explore what can be called the contemporary view of information technology projects.

3. Contemporary Views

In the early 2000s, management became very sensitive to the failure rate of projects and demanded that IT produce solutions. IT often reacted by pushing new coding technologies and buying some vendors canned development methodology. Today, neither of these approaches has generated meaningful answers to the problem and recent project statistics show little improvement in success rates industry-wide. Still, some companies have made successful inroads and this paper focuses on these new critical success factors.

The first issue to deal with is a development process. It is true that every system evolves through the basic life cycle steps, but the mechanics of this evolution are not generally agreed upon. The variety of system types and dynamic technical nature of solutions suggests that more must be understood about how to modify the process to fit the particular development situation. However, that does not mean a lack of process control. Regardless of the process, there are some management issues that need to be enforced. The following short list represents the most common

shortcomings of the modern IT organization and is not just a duplicate historical failure list:

1) Prior to project initiation there needs to be a process dedicated to matching the project proposal to organizational objectives. This mapping is often called aligning IT with the business.
2) A more formal effort is needed to document project scope early in the process. Establishing project scope goes against those who believe this activity cannot be done other than by building prototypes or using agile techniques to drive the project.
3) System function validation is a process that needs to occur throughout the cycle and not as a testing activity at the end. More effort is needed to assess the match of requirements versus current state from logical design through coding and on into the implementation activity. The earlier in the cycle deviations can be defined and corrected the less that these will have to be dealt with later.
4) Test plans need to be produced early and these plans need to be matched to the requirements.
5) Stakeholders need to be actively involved in the project throughout the life cycle. System development is not just an IT activity. This item is a known, historic problem but still not being corrected in most organizations today.
6) New system design should be based on a defined process, which has been carefully thought out and communicated to the participants.
7) In parallel with the system development process recognize that the user communities will be significantly impacted by that system. Increase focus supplies appropriate resources for the activities required to deploy the new system into that environment. Change management is essential and the needed focus is on the user side of the equation rather than the system side.

One of the most noticeable contemporary characteristic of the modern system is that it has to deal more directly with the user self-service than was required in traditional development environments. Self-service user requirements raise a host of human ergonomic considerations that have not previously been a part of system design. Items such as screen color, layout of the user graphical interface, and user navigation were not traditional computer technical issues but are now a critical system success factor. Also, the components used in many new systems often needs to be duplicated or connected across many interconnected processors, which in turn increases testing and debugging system complexity. Twenty years ago all of the application generally ran in one mainframe with little code or vendor variety, and the network was dedicated to limited connectivity.

A second significant complicating management issue in the contemporary environment is third party coordination, often across broad geographical and cultural boundaries. Coordination greatly complicates the communication process compared to having a contiguous internal, collocated development team work. In this situation the internal communication that was already recognized as being difficult is now compounded not only by the geographical dispersion of the team but also by the language and culture. Where requirements definition was one of the top failures in the traditional project effort, it is now multiplied exponentially in the global model.

Earlier discussion highlighted the fact that management has a decreased tolerance for a runaway project. There is a need to produce more accurate estimates prior to formal approval and then execute according to that plan. Also, it is more important to provide better tracking of actual project status. This requirement increases the complexity of processes related to estimating, measurement and status reporting. The attempt to deal with measurement of the overall process has highlighted the deficiency of support tools and techniques for activities such as estimating, reporting, and communication.

So, there are valuable lessons in project management that are now beginning to be broadly recognized. Organizations, who understand these basic ideas, can be observed. There is a fairly clear pattern emerging as to what it takes to emulate success. One important conclusion is that project success is linked to a more controlled task process and active external user/stakeholder involvement rather than through the use of a particular technology or canned development methodology. New technologies can improve segments of the overall process but cannot overcome the lack of these essential success variables. Far too often today technical system builders continue to focus on technical tools rather than on effectively managing the development process. Careful predefinition and management of three essential project variables — functionality, cost, and schedule – are still not high on the agenda of project managers, who have often been selected because of their technical acumen rather than their project management skills. This fact also suggests a needed strategy modification. Universities and professional organizations such as PMI are working to move this knowledge into the work force. Several university educational sources are beginning to formulate project focused programs. As a sample, programs at Stanford University, George Washington University, and the author's program at the University of Houston offer high quality exposure to project management. Expansion of these offerings is yet another example that the topic has been recognized and organizations are working to improve the situation.

Dr. William Ibbs researched the effect of process maturity on project success and offers some compelling evidence that improved success is a definable goal. [3] As a result of his research, a correlation was shown between project maturity levels and ability to manage cost and schedule outcomes for projects. Using this model a project management maturity level can be quantified on a five point scale, with one being the lowest and five the highest. Within high maturity environments the average cost and schedule deviation from initial plan averaged less than 8% of original prediction as

compared to the typical 200% variances reported from the Standish studies.

Another well-known research source, the Software Engineering Institute at Carnegie-Mellon University, has similarly defined models to measure operational maturity in IT organizations [see www.sei.cmu.edu]. Its pioneering research also supports the conclusion that firms with higher levels of process maturity correlate with lower costs of development (e.g., higher development productivity).

Given the obvious need and the compelling, achievable benefits, why do many organizations refuse to fund and implement standard processes? Two common explanations are offered:

1) Increasing the project maturity level requires a significantly commitment to more formalization and documentation with greater front-end costs. Management is often unconvinced that these certain near-term investments will produce what they perceive as uncertain long-term benefits.
2) Technically oriented managers continue to search for the silver bullet technology or process model that will solve the development problem. Unfortunately the latter changes can include what many incorrectly view as "extraneous" steps such as documentation, planning, status reporting, or post implementation review.

Largely due to these two philosophical dilemmas, IT project management implementation remains adrift. If left unaddressed, senior management will become increasingly frustrated by the failures of its development organization. Eventually this dilemma can, and often has, contributed to the decision to outsource the development function, creating a new set of problems focused not on efficiency of development but the overall effectiveness of the IT investment.

To resolve common project performance failures, management must recognize the basic causes of failure and attack these issues in a rational and prioritized manner. Projects must be formulated to solve business problems and they must be actively tracked to ensure the desired objective is sustained through development and implementation. An IT project must be viewed as a business activity, not a technical one. Users and senior management must take ownership of this activity - even lead it in many cases. Technicians must recognize that a project is not a technology proving ground; rather they must work to predict economic as well as technical outcomes, meet user expectations, communicate status in a timely manner, and complete the activity as formally planned. These premises sound trivial but are not the normal project mode in many organizations. Consider the actual work activity and let's explore how its components affect success of project outcome.

4. System Development Framework

As shown in figure 1, at the aggregate level the system development life cycle consists of four interacting processes. These include 1) an organizational alignment process for ensuring IT project compatibility with the organization and its objectives, 2) a technical work process for carrying out the necessary technical work required to construct the new systems, 3) a project management process to oversee the management and administrative tasks required to ensure proper coordination through the various stages of the project, and 4) organizational change management for implementing the new system into the business model.

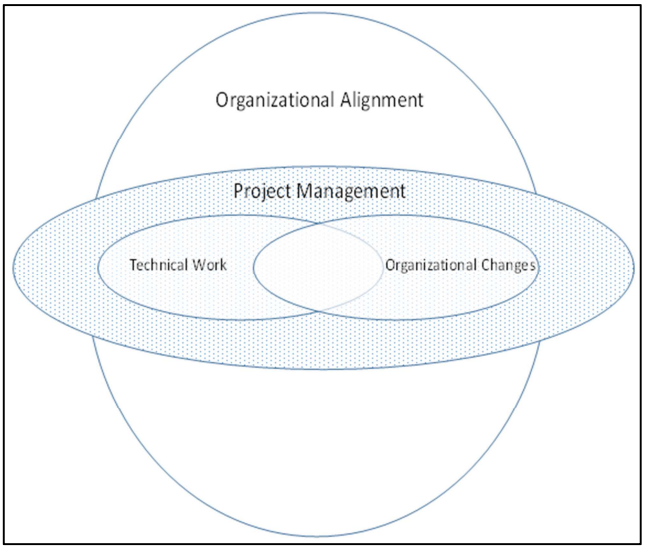

Figure 1. *Interlinking Aggregate Processes.*

Proper execution and coordination of each of these interlocking domains affect the ultimate success of a project. Project managers, if they are to succeed, must recognize the importance of each of these domains and be skilled in orchestrating the decision threads that weave through the intertwining process components. It is also important to recognize that these processes involve broad segments of the enterprise and there are varying success criteria that must be negotiated across the user community. After a quick examination of the first two processes to ensure that the reader gets the right perspective on the overall structure, the latter two processes will be digested and more thoroughly discussed.

4.1. Organizational Alignment Domain

It must be recognized that a flawed project vision will always, by definition, produce a flawed output. Obvious? Yes, but then why is this result often observed? It is seen when projects bubble up from lower levels in the organization and serve a local need that is not congruent with higher-level business goals. This origin can sometimes be major sources of innovation and, in itself, is not the problem. However, it is important to recognize that an organization operates with finite resources just like it has a limited amount of funds for capital investment. The appropriate process for approving a project is one which matches the project with the highest organizational payoff. For a more detailed overview of this subject refer to Benko and McFarland's excellent overview of this subject in *Connecting the Dots* [5].

The decision challenge of the allocation exercise is to establish a project slate that extends along a broad front encompassing both strategic and tactical objectives. Some projects are designed to keep the company moving in fighting trim; others target more strategic objectives, such as customer-facing applications or strategic alliances often related to external items such as customers or partners. The chosen mix for such efforts needs to be guided by the corporate objectives and with heavy involvement throughout the organization. Moreover, it is a dynamic process with both system costs and potential benefits estimates fluctuating as the project development process continues. Project alignment must be revisited as conditions change and as new information emerges. Project alignment falls outside this paper's scope, but it is important to recognize that a result of the alignment activity should be to define an appropriate project scope definition. *The optimum solution to the wrong problem is still the wrong solution!* The output goal is to seek and approve proper project targets prior to approving significant resources to produce an outcome.

4.2. Technical Work Domain

There are basically five birth to finish stages in an IT development project life cycle. A typical list of such stages includes: 1) requirements definition, 2) technical design, 3) construction, 4) testing and validation, and 5) implementation and closeout. Professionals who have been involved in actually identifying, structuring, and executing these activities for a complex IT project recognize that it is far easier to discuss stage development to actually manage and deliver a successful outcome. Human interactions within the organization, a dynamic technological environment for both development and system operation, and the complexity inherent in creating the desired functionality combine to make software development one of the most complex undertakings of humanity. Yet, the final solution, perhaps consisting of millions of lines of code, can with only a few keystrokes and in a few seconds be loaded into a computer with all of its complexity buried away. Unlike a large petro-chemical refinery, the complexity of a computer system is nearly invisible, its encoded intellectual logic and subsequent maintenance costs likely unrecognized by the user.

Viewed abstractly, systems development involves turning a set of user logic process requirements into a complex set of mechanical code and technical components. If the system is straightforward, the extraction process for this translation should be highly successful. As the complexity of the logic increases and the number of users expands, this process can become a world-class challenge in many dimensions. Though there are certain fundamental steps that must be accomplished, the most appropriate tools and processes for achieving these steps can vary depending upon several factors. Among the more obvious are the size of the project, the degree of risk, involvement of other parties, depth of understanding of the environment in which the system will operate, and acceptance by the intended user population. A contemporary development model needs not only to keep the required process functions in perspective but also to allow flexibility in the task sequence and means for producing the required functions. Publishing a lockstep development process will not produce quality results, even if you could enforce such a requirement. IT developers are particularly resistant to perceived inefficient bureaucratic activities.

The greatest current controversy among the technical development group involves techniques for collecting user requirements and validating those requirements. The traditional method of user requirement definition is to do that first and then start the system design and coding activities. There is an increasingly common view that users are not capable of a precise level of definition and therefore some level of system prototyping must be employed to show what the final system will be. Taken to the extreme this school of thought would suggest that this process be continued throughout the whole cycle and terminate when the user is satisfied. Likewise, there is a lack of agreement regarding the timing and approach to system validation and testing. Great merit is earned when test plans are completed prior to coding as opposed to after the fact activity. In reality, this activity is "validation" rather than "testing." The semantics are meant to focus on requirements matching rather than disjointed ad hoc testing late in the cycle. Regardless of individual opinions on these two topics, today's management requires that some outcome forecast be made prior to project approval. A forecast dictates a requirement for producing a level of early scope definition before significant resources are approved. More discussion on these issues can be found in *Radical Project Management* by Rob Thomsett or *Information Technology Project Management* by Kathy Schwalbe. Also, the quick build philosophy can be reviewed further from various sources discussing the Agile or Extreme development process. [7, 13].

The key to achieving an appropriate development process is to define a common, repeatable approach and train the staff on its merits. The process should have the following traits:

1) Management should be involved in the project portfolio decision process.
2) Formal external review events should occur at selected points in the development cycle with the decision being to continue or abort the effort.
3) Defined artifacts should result from the effort. Artifacts should include such items as a project charter, business case, technical documentation, user documentation, and other items as defined by the governance entity.
4) Status metrics related to the activity need to be captured and used in the review and improvement process.
5) Appropriate project status should be readily available to the stakeholders and management.

Failure to achieve these traits leaves a void in one aspect of the overall process. Technicians do not agree with the importance of these items as much as outside parties do. It is important to focus the project activity on satisfying customers at the approved price and not just playing with the technology. Visibility and communication are vital

responsibilities. For those wanting to look deeper into this subject a more detailed listing of "critical software practices can be found at the Integrated Computer Engineering web site. [8]

Earlier the basic development life cycle was identified. The basic life cycle phases offer reasonable check points for project technical and management review. There are two important support elements. A formal artifact repository and a defect tracking system should be implemented. These two repositories are added as communication integrators between the development stages and are vital to the technical work and management processes. One of the most vital components of the technical work process and the associated management process is the artifact repository. Basically, it is a formal data store of project information that can be used to supply appropriate stakeholders with technical or management information. It should contain the working data from the various project activities including the following:

1) System requirements
2) Technical design models
3) Test plans and scripts
4) Code libraries
5) Change management documents
6) Test data
7) Operational documents (user, technical, operations, etc.)
8) Project plans, status, and other related documents

Most projects teams file artifacts of this sort in various electronic document folders. It is recommended that this information be available to a broader group and be stored in a standard repository with access controls. As a project moves from the development to the maintenance mode, this repository should be accessible by the maintenance team for care and feeding over the remainder of the life cycle. Also, much of the development history is archived for review by future development teams or audit sources. A formal project repository strategy is a key strategy for long term process improvement given it contains the best source to examine lessons learned.

The second repository source contains data regarding defects identification and problem resolution during the development and operation of the system. The traditional concept of defect tracking is normally thought of as identifying detects in a working program. A defect can mean "a variation from requirement" which can occur at any point along the development process. For instance, in reviewing a narrative requirements statement or schematic design model, it is feasible to discover that the translation does not match its corresponding operational requirement. This contradiction should be captured at the point of identification and then managed to resolution. In some cases a defect may be left, while in others, it will be assigned to someone to review or repair. Tracking detects first ensures that they get resolved early (a good development strategy) and secondly provides insight into the ongoing quality of the process. Defects should converge to near zero as the project evolves and this data provides the analytical keys. Through defect tracking it is possible to identify confusing requirements and excessively complex code which are project quality objectives. The underlying philosophy for both of these repositories is that the project cannot be effectively manage unless it is measured or documented.

These two repository groups represent the primary raw data source describing project performance. Through these sources it is possible to create status reporting and a myriad other analysis activities with minimal effort. Long term, the development goal should be to automate relevant status data as a by-product of the activity itself, but that will not be accomplished initially. Nevertheless, having the data available in one central place is a major communication advantage. In each case, the filing structure for these repositories should be standardized and use of a content management system be implemented to facilitate archiving and retrieving.

What is not so obvious is that the technical development productivity tools are evolving in a positive way. The industry is slowly adopting modeling (blueprinting) standards that can be shared across the various stakeholder groups—similar to what was described for the construction industry. Much of the IT industry is following system documentation as outlined by the Unified Modeling Language (UML), originally produced by Rational. [10] Also, various development tools are now entering the marketplace that are capable of producing code from these models. Utopia would be that the whole system could be automatically produced from the original system requirements specifications. This capability is not on the near term horizon. IT organizations have searched for viable code generation utilities for years, but the continuing change in technology and changing favored languages has made the previous efforts only marginally successful. It appears that broader acceptance of UML and improved compatibility of coding languages is moving this goal closer to a reality. Today's vendors advertise that they can generate 80% of the code from design models. Envision the flow process of design models being stored in the artifact repository and then fetched into the code creation. Both humans and machine created code would complete the coding effort. The repository would hold a full working definition of requirements, design, code, and validation documentation. This level of maturity is not achievable for today's development projects. If it is achieved, future maintenance of the system will be accelerated by having a work flow from user logic to final code structure. In the current development culture, there is no traceable link and, as a result, the subsequent maintenance process is often labor intensive.

In a similar fashion, testing objects will be produced from graphic-generated requirements and design models. At both the design and code stages, there are several supporting technologies that are entering maturity stage and promise to aid in the streamlining future projects. The future challenge with these tools is to interconnect across the life cycle stages in order to integrate workflow. The current vision for tools is each will generate partial system definition into a common

data repository. Code generator tools will then use stored system modeling specifications (blueprints) to create code units. Validation tools will look at both requirements and system models and generate appropriate test scenarios. A general technical view of the future repository environment is encouraging. A key factor is the market availability of emerging products. In order for an organization to make productive use of an enabling tool suite, it should move to create consistent, repeatable processes and continue to evolve those processes as the technology roadmap becomes clearer.

An often misunderstood factor within the traditional development cycle, the coding process, causes effort to focus on the wrong targets. In many development projects, this is the only visible artifact – code. Surprisingly, resource consumption of system development is only 20% code writing. Even more surprisingly, the testing/validation activity consumes at least 20% and sometimes as much as 50% of the project's resources. Testing and validation requires so much work because there was very little other than code as a basis for building and executing tests. Sometimes, software developers and testers struggle to find documented requirements to ensure they are being covered. The creation of test data is equally haphazard and often involves using a large volume of replicated production data to generate code coverage and observe the actual results. Such approaches are inefficient, ineffective, and leave validation with a great potential for productivity gains. Improvement cannot occur without the earlier steps being more disciplined in their approach. Nevertheless, requirements validation is fertile ground for major productivity improvement.

If one examines the current resource consumption in the various development steps, it is possible to make rough estimates regarding productivity improvements from the implementation of integrated automated tools. A savings of 10-25% is estimated for the traditional development cycle just in the validation/testing phase. Certainly, this potential saving alone is enough to attract management attention. Similar improvement is anticipated in the other development tasks and savings could be further amplified as the concept of reusable code modules becomes more available. Organizations should be focused on these potential savings as they re-engineer their existing development processes. In order to move an organization towards the model environment outlined here, there must be high level recognition that the described factors are, in fact, accurate and relevant. There must be an ongoing improvement effort to choose tools that fit the model. An initial look at UML as a specification strategy and the associated tool suite is a reasonable starting point. [13] After any initial tool scoping review, examination of other vendor products should be completed and a phased implementation strategy formulate.

It is not intuitive why adding process structure and control to a process can save time and money. For example, managing scope creep, finding errors earlier in the development cycle, improving the cost of code development, maintaining artifacts in a structured repository, and decreasing the system validation/testing cost have significant productivity implications. There is research evidence compiled from the Software Engineering Institute Capability Maturity Model (CMM) program that showed orders of magnitude productivity improvement resulting from an organization going from an initial maturity (level one) to higher levels of process rigor (maturity levels three to five). [14] A project management model environment, outlined in previous sections, has the same characteristics and should yield the similar positive results when properly implemented.

What is most important from a management view is that an appropriate work process must be properly defined and used by the development project teams. An improved understanding of project management theory and an enterprise project management model will help the organization develop a culture that is appropriate for their specific needs. In this regard, various universities offer programs that will aid in this cultural migration. It is essential in the change management process to recognize that, in order to gain acceptance for any new approach to a highly technical and complex problem area, some positive and tangible motivation must be presented. Making that first move will be a management challenge. Technical teams love technical challenges but do not relish increased formalization of work process placed on them. The key is making this transition more transparent through both a theoretical education of research evidence in the field and an increase in use of automated tools for the various steps. If those strategies are productive and create positive benefits, then further acceptance will be embraced—a win-win strategy.

4.3. Project Management Domain

In figure 1, lying above the technical work and organizational processes is the project management activity. There are many documented prescriptions for effective project management, but the one that likely has the broadest following comes from the Project Management Institute (PMI) in the *Project Management Body of Knowledge*, more commonly called the PMBOK. This publication divides the basic activities of project management into ten knowledge areas: [11]

1) Integration: ensure that the various elements of the project are properly coordinated. This includes plan development, execution, and integrated change control activities.
2) Scope: identify what the product or service will be.
3) Schedule: ensure timely completion of the project.
4) Cost: determine the financial requirements of project and build an approved budget.
5) Quality: establish quality standards and output monitoring.
6) Resource: make the most effective use of people and materials.
7) Communication: complete timely generation, collection, dissemination and disposition of project information.
8) Risk: identify, analyze, and respond to project risk.

9) Procurement: acquire goods and services from outside the organization.
10) Stakeholder: identify stakeholders and manage their project involvement, requirements and expectations.

Taken together these ten knowledge areas constitute PMI's recommendation regarding processes and input/tool and techniques/outputs (ITO) that need management for successful project completion.

The PMI structure is good in that it points out critical issues with which to deal. However, the process is general in nature and it does not help so much with execution in an IT environment. In other words, it describes the "what," but not so much the "how" to carry out the needed tasks. It is important to point out that most IT projects do not give sufficient consideration to these activities, but often pay later for ignoring issues such as scope control, risk, quality and communications. [11]

Part of successfully managing a project is to gain insight into what can go wrong and anticipating it in a timely manner. In an earlier section, a list of shortcomings associated with project failure was provided. Let's look at additional reasons for failure. The following five reasons are often cited as contributing to a lack of project success:

1) Incomplete user requirements
2) Inadequate management involvement
3) Ineffective communication
4) Immature work processes
5) Technicians unwillingness to be constrained by formal standards

Surveys related to project failure identify many more items, but the root causes generally are linked to these five. To illustrate this point, functional scope creep is a typical item often identified as a cause for failure. However, scope creep spawns primarily from incomplete initial requirement statements by the system definers and lack of proper management control oversight during execution. Thus, it is a secondary tier event engendered from the root causes. Likewise, schedule and budget issues most often arise from similar underlying causes. So, the point is that the primary root causes must be effectively managed and from that the other identified failure factors will improve. From a management standpoint it is important to recognize that project scope control is probably management's singular most powerful technique for keeping a project on track. With project scope under control, management will face far less daunting issues. Proper scope definition requires a set of requirements that fit both the business and user needs. Unfortunately, there is little agreement among professionals regarding how those requirements should be captured and documented. An analogy for this disagreement can be drawn to building a house. Constructing a new house utilizes blueprints (models) that have been approved by the future owner and a defined work (construction) process that follows this design. In the construction industry, there are documented standards that are adhered to by the work force (and inspectors that confirm that status). Contrast the construction industry's accepted discipline to the information technology industry which still struggles with fundamental definitions for blueprinting standards, coding standards, and other activities involved in the formalization of a development process. IT would profit from following the lessons learned from the construction and the aerospace industries. Bridges and airplanes seldom fall down from poor work process, but computers systems do. This process maturity could be improved through adherence to improved technical work and management practices.

At least two factors have led to the IT industry's lack of process standardization. First, the rapid change of technology, coupled with the too common "magic bullet" mindset, clouds the implementation of a commonly accepted approach. Each year newly enhanced and more advanced development tools enter the market. So, it is hard to settle on a single one for an extended period of time. Second, the work force is by nature highly intelligent, creative, and technically literate. They do not like to be shackled with a formal process that is viewed as out of date by the time it is published. Instead, a dynamic software development tool set is preferred which undermines the acceptance of a standardized formal process. This tension might not abate soon, but management must force greater structure into the process in order to improve this situation. An appropriate management process is not significantly affected by technology changes. Technology can be an accelerator of a good process, but probably will not do much to improve a flawed process. Viable stable processes can be installed and technology enhancers added to these as they become mature products.

4.4. Implementation Domain

Extending the actual development effort is the process of migrating a finished system into a production environment. As systems grow in scope, the mechanics to successfully deploy a system into the production environment also grows. In many cases the deployment tasks will require changes to central servers, remote hardware, networks, data bases, and, often most critical, user business processes. In most cases the purpose of the new system is to implement new business processes and to modify existing ones. Project teams often lose sight of these impacts and are often disappointed when users are not excited to see the new system. In fact, users often respond apathetically to, or even resist, what the developers saw new opportunities. The mechanics related to the new system are sometimes not timely communicated to users. In order to be successful, these implementation issues must be dealt with in a similar manner in which internal project work tasks are analyzed and dealt. Invariably, resources are allocated to accomplish the business process change created by a new system.

The key message with implementation is the project plan must address the essential organizational, behavioral and process requirements as well as technical ones. It is important for higher level and project managers recognize that the knowledge and skills for organizational change are quite different from the technical skills. IT staff has often not been

trained in organizational change and personalities are not acclimated toward cultural and social change management processes. Changing business processes is often more difficult than it appears given the business users may not share a common vision, may want to learn something new, or may fear negative consequences. These potential stumbling blocks might be avoided by selecting a manager from the impacted user community as the project leader or project team liaison.

5. Costs and Benefits

A major hurdle for an organization moving toward development process improvement is allocating resources with low likelihood of short-term benefits. Two or more years may be required to see visible results. In the short term a rigorous project management model will add about 10% above the base costs of the technical work tasks. These economics might be a tough sale, particularly in a company culture driven by quarterly earnings reports. Attitudes towards project activities are now shifting away from the technical bias toward an increased management focus. Senior management no longer tolerates the historical levels of project failures and the corresponding lack of visibility in regard to ongoing project status. Project stakeholders are looking for visibility and predictability. When one reviews some of the meteoric cost and schedule overruns resulting from attempts to install ERP systems (e.g., SAP, Oracle, Lawson, etc.) over the past years, it is little wonder that the need for more control has become so widespread. If management does not care how much is being spent, when a project will be finished, and what the system will do, project management is not needed. Rationally, not many companies are so tolerable.

On the surface there are convenient excuses to not add the project management overhead. What is it worth to management to know the actual status of the project including schedule, budget and functionality? What is it worth to know your software development organization can reliably meet its responsibilities? Too often poorly organized projects operate long after the organizational value for the project has gone. Effective project management dictates corrective action be initiated in a appropriate time frame, not after damage is done. As organizations learn to effectively use formalized project management processes, the overhead level declines to the 5% range. [3] The real benefits derived from effective project management are the potential for lower project costs and future operational benefits. Some examples are improved estimating techniques, well-defined operational processes, early detection of missed requirements, and lessons learned. Four-fold productivity increases have been measured by organizations who have successfully accomplished improved level of task and management maturity. [12] The alternative is to leave the situation in limbo and to pay a far larger price in future cost overruns and failed projects.

There are two final items surrounding the core discussion regarding project management benefits. First, over the past few years, IT has become increasingly embedded into the business process fabric of the firm. Major business processes cease to operate when the associated system is not working. The airline reservation system is high profile example, but more and more companies are operating business architecture enabled by technology architecture. These highly interactive architectures have global scope and complex functionality requiring 24/7/365 availability. Such business functionality and technology cannot be thrown together necessitating more careful planning, design, and integration than the systems of just ten years ago.

Second, a system lives in a community of other systems each competing for scarce resources. In order for project management to be successful in an enterprise, there must be a rational allocation of resources scheme across the investment portfolio. In many organizations, the enterprise investment process is handled by an organization entity, usually the Project Management Office (PMO). This organization is critical to keep the overall organization moving in the right direction, not only from a process functional view but also from a technical and financial standpoint. Properly constituted, the PMO aids in implementing the concepts discussed in this paper and is the best current tactic for keeping the organization aligned with business goals and information systems.

6. Conclusion

The risks and costs associated with an ineffective software development projects are well documented. In this paper, the broad waterfront of traits required for project success in software development were explored. At this point in the evolution of project management knowledge, there is contemporary research and practice indicating that defined, formalized project management processes have a positive effect on the short and long term project execution. Moving an organization to a mature system development environment is a multiple-year challenge that requires tenacious senior executive leadership and commitment and, probably, the transfusion of a new project management culture. Successful implementation of a more mature project management state will herald measurable improvements in future deliverables and outcomes. Conversely, failure to change the status quo will likely exacerbate the current trends and eventually trigger credibility loss of in-house software development capability when compared to third party providers who achieved the goal. Hopefully, this discussion regarding project management and software development concepts, processes, tools, and repositories stimulated motivation to pursue organizational and cultural renaissance. The general roadmap and key outlined points should aid in formulating an organizational strategy to achieve successful development projects.

References

[1] "The Best and the Worst: Top 10 Corporate IT Failures in the 1990s", September 30, Computerworld, 2002, ttp://www.computerworld.com/news/2002/story/0,11280,74620,00.html.

[2] *Information Technology Project Management*, RMC Publications, Inc., Schwalbe, Kathy. Ninth Edition, 2018.

[3] "Searching for the $$$ Value of Project Management," Ibbs, William P. presented at the PMI Houston 2003 Conference, Houston, Texas, November 7, 2003.

[4] "Chaos," 2014 Project Smart. The Standish Group Report, 2014.

[5] "Connecting the Dots", Benko, C. and McFarlan, Warren F. HBS Press, 2003.

[6] "Extreme Chaos," The Standish Group Report. 2001.

[7] "Manifesto for Agile Software Development," www.agilemanifesto.org.

[8] Integrated Computer Engineering, www.ICEINCUSA.com.

[9] *UML Distilled: A Brief Guide to the Standard Object Modeling Language,* Edition 2, Fowler, Martin and Scott, Kendall. Addison-Wesley, 1999.

[10] "New to Rational," https://www.ibm.com/developerworks/rational/newto/.

[11] *A Guide to the Project Management Body of Knowledge PMBOK® Guide,* 2017 Sixth Edition, Project Management Institute, Newtown Square, PA.

[12] "How Mature is Your IT Organization?" Curtis, Bill. Presented at the Information Systems Research Council, University of Houston, Houston, Texas, January 16, 2003.

[13] *Radical Project Management*, Thomsett, Rob. Yourdon Press, 2002.

[14] Software Engineering Institute. The Software Engineering Institute (SEI) is a federally funded research and development center sponsored by the U.S. Department of Defense and Operated by Carnegie Mellon University (www.sei.cmu.edu).