4-2009

A Framework for Categorizing Key Drivers of Risk

Christopher J. Alberts
Carnegie Mellon University, cja@sei.cmu.edu

Audrey J. Dorofee
Carnegie Mellon University, ajd@sei.cmu.edu

Software Engineering Institute

A Framework for Categorizing Key Drivers of Risk

Christopher J. Alberts
Audrey J. Dorofee

April 2009

TECHNICAL REPORT
CMU/SEI-2009-TR-007
ESC-TR-2009-007

Acquisition Support Program

http://www.sei.cmu.edu

Carnegie Mellon

This report was prepared for the

SEI Administrative Agent
ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2100

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about SEI reports, please visit the publications section of our website (http://www.sei.cmu.edu/publications).

Table of Contents

List of Figures

List of Tables

Acknowledgments

We would like to thank Lisa Marino for the time and effort she contributed while editing this technical report. We would also like to thank the following people for the time and effort they contributed to reviewing this document and providing valuable technical feedback: Julia Allen, Robert Ferguson, Patricia Oberndorf, and Carol Woody.

Abstract

In today's business and operational environments, multiple organizations routinely work collaboratively in pursuit of a common mission, creating a degree of programmatic complexity that is difficult to manage effectively. Success in these distributed environments demands collaborative management that effectively coordinates task execution and risk management activities among all participating groups. Approaches for managing program risk have traditionally relied on tactical, bottom-up analysis, which does not readily scale to distributed environments. Systemic risk management is an alternative approach that is being developed by the Software Engineering Institute (SEI). A systemic approach for managing risk starts at the top—with the identification of a program's key objectives. Once the key objectives are known, the next step is to identify a set of critical factors, called drivers, that influence whether or not the key objectives will be achieved. The set of drivers also forms the basis for subsequent risk analysis. This technical report describes a driver-based approach for managing systemic risk in programs that acquire or develop software-intensive systems and systems of systems. It features a framework for categorizing drivers and also provides a starter set of drivers that can be tailored to the unique requirements of each program.

1 Introduction

The responsibilities for managing a software program, and the resources needed to carry out program activities, have traditionally aligned along organizational boundaries. However, circumstances in the business environment, such as the globalization of business and the fast pace of change, have led to an increase in outsourcing and partnering among organizations. It is now common for multiple organizations to work collaboratively in pursuit of a single set of objectives, creating a degree of programmatic complexity that is difficult to manage effectively. Success in these distributed management environments[1] demands collaborative management approaches that effectively coordinate task execution and risk management activities among all participating groups.

In distributed environments, management control of programs, business processes, and technologies is shared by multiple decision makers. The emergence of collaborative ventures has provided a wealth of new opportunities for many organizations. For example, vast amounts of information can be shared quickly and transparently among people who are geographically dispersed. In addition, organizations can quickly form partnerships to pursue new business ventures that take advantage of rapidly changing market conditions. However, along with these opportunities come new types of risks. Many people are having difficulty managing risks caused by program, process, and system complexity. Nearly everyone is having trouble managing shared risks that cross organizational boundaries. Our field experience[2] indicates that current approaches for managing risk are insufficient when employed in distributed environments—**new approaches** are needed.

1.1 New Approaches for Managing Risk Across Distributed Environments

Based on our experience, we believe the prevailing risk management paradigm needs to shift from tactically-oriented approaches to those that employ a systemic focus. Tactical approaches are designed to individually manage each event that could have an adverse impact on a program. These approaches tend to incorporate bottom-up analyses and do not readily scale to distributed environments. At their core, tactical approaches for managing risk assume a linear cause-and-effect relationship between each source of risk (i.e., each potential event) and its direct consequence.

However, distributed management environments are anything but linear. These environments comprise a network of highly complex components that are linked together. Matching a single source of risk to a single consequence is too simplistic and has proven to be ineffective in many instances. In these environments, risks tend to have many causes that work in tandem; the assumption of one source for every consequence does not apply. We have found systemic approaches to be better suited to managing risk in distributed environments. In contrast to the bottom-up analyses employed in tactical risk management,

[1] In this document, the term *distributed management environment* is defined as a program, process, or technology where management control is shared by multiple people from different organizations. It is used interchangeably with the terms *distributed environment* and *multi-enterprise environment*. A *distributed program* is one type of distributed management environment.

[2] The authors have a combined 32 years of experience in the field of risk management. We have developed methods for managing risk in software acquisition and development programs. We have also developed methods for managing cyber security risk. Our current methods integrate our work in both areas and define a life-cycle approach for managing risk. We have conducted numerous field pilots over the years when developing our methods.

systemic approaches incorporate top-down analyses of risk, which provides a holistic view of risk in relation to program objectives. Current SEI research is focused on systemic risk management, and this report highlights a key aspect of that research.

1.2 Risk Management Research

In 2006, the Carnegie Mellon® Software Engineering Institute (SEI) chartered the Mission Success in Complex Environments (MSCE) project to develop new and innovative management approaches for managing risk. We began our research by directing our attention toward developing risk management approaches designed specifically for the unique requirements of multi-enterprise environments, including (but not limited to)

- distributed software acquisition and development programs
- Department of Defense supply chains
- organizations in dynamic, rapidly changing business environments
- distributed information-technology (IT) processes
- distributed business processes
- processes supporting critical infrastructures
- software-intensive systems and systems of systems

While each of these examples has unique characteristics, all share a common aspect—complex, distributed environments that are inherently risky. Over the past three years, we have primarily focused our research on two domains:

- software acquisition and development programs
- cyber-security incident management processes

During this time, we developed Mosaic—a suite of methods that can be used to manage risk across the life cycle and supply chain. We have successfully used Mosaic to assess risk in both research domains.

This technical report presents the key concept underlying our research into systemic approaches for managing risk in the distributed software acquisition and development programs: A framework for categorizing key drivers of risk.

1.3 Framework for Categorizing Key Drivers of Risk

In a distributed program, management control is shared by multiple people; no one has absolute authority over the end-to-end program. Systemic approaches for assessing risk in a distributed program begin with the identification of its key objectives. Once a program's key objectives are known, the next step is to identify a set of critical factors, called drivers, that influence whether or not the key objectives will be achieved. This set of drivers for is used to assess the program's current strengths and weaknesses, and it also forms the basis for the subsequent risk analysis.

Our pilot activities have demonstrated the contextual nature of drivers in two main ways.

® Carnegie Mellon is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

1. *Drivers are derived directly from a program's key objectives.*
   As a result, a unique set of drivers must be identified for each specific environment.
2. *Drivers differ at various points in the life cycle.*
   For example, early in the life cycle, programs are focused on developing plans and identifying requirements. Later, programs are focused on executing those plans and deploying the system being developed. The nature of the activities performed in each life-cycle phase is different, and consequently, the drivers for each phase will reflect this difference.

During our research, we analyzed the results of a variety of assessments, looking for patterns among the findings. The result of this analysis was the identification of a framework for categorizing key drivers. This report presents that framework and also provides a starter set of drivers that can be tailored to a software program[3] to create the unique set suitable for that program.

1.4 Audience and Structure

Our primary audience for this technical report is managers and lead engineers who have experience assessing and managing risk in software development programs. However, anyone who has experience with or is interested in risk, issue, and opportunity management or process improvement might also find this report useful.

This technical report comprises nine sections and two appendices. The first three sections provide the conceptual background for systemic risk management. The focus of the technical report shifts from the concepts to practice in Section 4. The following summarizes the content for the remainder of this report.

- *Section 2: Focus on Objectives*—presents the basic structure of a software program and illustrates how the objectives of a work program define its picture of success
- *Section 3: Two Fundamental Approaches for Managing Risk*—describes tactical and systemic approaches for managing risk
- *Section 4: Driver Framework*—presents a framework for categorizing drivers
- *Section 5: Driver Identification*—provides a starter set of program drivers
- *Section 6: Driver Analysis*—outlines an approach for determining how drivers are influencing a program's objectives
- *Section 7: Using Drivers and the Driver Framework*—describes different scenarios for using the drivers
- *Section 8: Extending the Driver Framework to a Distributed Program*—outlines future research directions for this work
- *Section 9: Summary and Future Directions*—provides a brief overview of pilot results and outlines future research directions
- *Appendix A*—provides a questionnaire for analyzing the starter set of drivers
- *Appendix B*—contains a glossary of terms used in this document

[3] In this document, the term *software program* refers to any acquisition or development program that produces a software-intensive system or a system of systems.

2 Focus on Objectives

In its broadest sense, the term *objective* refers to a desired result or outcome that is being pursued. A *program*[4] is defined as a collection of interrelated work tasks or activities that achieves a specific result. It includes all tasks, policies, procedures, organizations, people, technologies, tools, data, inputs, and outputs required to achieve a specific, predefined set of objectives.

Figure 1 depicts a generic program and the activities required to achieve its associated objectives. The basic function of the program depicted in Figure 1 is to transform the input into the desired output. To achieve its objectives, the program must perform four activities in the order shown, while also adhering to any cost and schedule constraints. Program execution begins when Activity 1 receives its input. Upon completion of Activity 1, its output triggers Activities 2 and 3, which are then performed in parallel. When Activities 2 and 3 are complete, their outputs are forwarded to Activity 4, the last in the sequence. Upon completion of Activity 4, the program is finished with its work. If all of the activities have been performed correctly, the overall set of objectives for the program will have been successfully achieved.

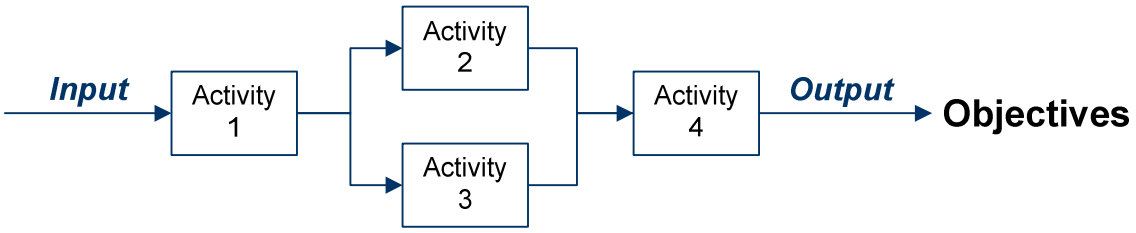

Figure 1 Program with Four Activities

A program is more than a collection of activities, however. It is a complex *organizational system* that brings together a variety of diverse components, or assets (people, technologies, equipment, facilities, information, procedures, and work products). These assets are organized in a specific way to achieve a particular set of objectives or mission. The assets are organized into a set of interacting, interrelated, and interdependent parts that must function as a whole to accomplish given objectives.

With outsourcing and collaboration becoming so widespread, a set of objectives often extends beyond a single organization to include multiple organizations that are often geographically distributed, culturally diverse, and independently managed. Since management control is shared in multi-enterprise programs, program planning, decision-making, and execution becomes complicated. In these multi-enterprise, or distributed, environments, decision makers must work collaboratively to strike a balance between

4 Throughout this report, we use a single term, *program*, to refer to a collection of interrelated work tasks that achieves a specific result. We use this term because our current research deals primarily with large software acquisition and development programs. The term project is used interchangeably with the term *program* throughout this document. Finally, significant distinctions between *processes* and *programs* do exist. However, our research has shown that the basic concepts presented in this report also apply to assessing and managing risk in *information technology processes*.

achieving their own objectives and the overall program objectives, which are often in conflict with each other.

2.1 Distributed Programs

In a *distributed program*, management control is shared by multiple people who are often from different internal and external organizations. Over the past several years, outsourcing, collaboration, and globalization have become increasingly commonplace and necessary, which has resulted in distributed programs becoming commonplace and necessary. The paradigm of having a single person with effective management authority over an entire program is becoming obsolete. It is being replaced by a collaborative model where management authority is shared by several people, each overseeing a part of the overall program, who may or may not be communicating with each other.

Figure 2 depicts a simplified example of a distributed program. Notice that the program from Figure 1 is now part of a larger program that links four distinct sub-programs. Each sub-program has its own unique set of local objectives that define its mission. Each local set of objectives is supported by local activities and they, in turn, have their own objectives. Successful completion of local activities is required for the local objectives to be successfully achieved. However, the overall program objectives are not achieved until all activities in all sub-programs have been successfully completed. As illustrated in Figure 2, four organizations have pooled their resources to complete a single set of objectives, thus creating a distributed program. Each group in Figure 2 must adhere to the mission of its parent organization. However, the four organizations must also work collaboratively toward achieving a common set of overall program objectives, which creates a virtual enterprise with its own unique mission.

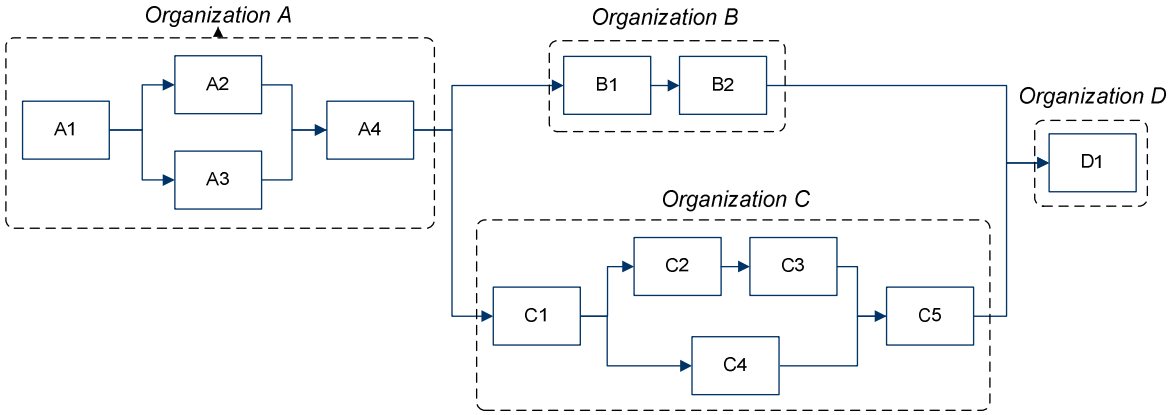

Figure 2 Program Spanning Four Organizations[5]

[5] This figure is a simplification of a distributed program. We recognize that distributed programs are much more complex with interdependencies among activities within organizations and between organizations.

2.2 Key Objectives

A *key objective* is a vital outcome intended to be achieved in the future; it provides a benchmark against which success will be judged. A program typically focuses on three distinct types of key objectives: (1) product, (2) cost, and (3) schedule. Product objectives define the nature of the items produced and are often referred to as technical objectives in the software development domain. [6] For example, if you are developing a software-intensive system, the product (i.e., technical) objectives minimally define the functional and performance characteristics of the system as well as other desired attributes, like safety or security.

Focusing solely on such characteristics limits the context within which the product objectives are viewed. In such a narrow context, success is based on whether the system satisfactorily meets its functional and performance requirements. Broader issues, such as whether the system effectively supports operations and whether people can use and maintain the system, are *sometimes* considered to be out of scope. However, operational, usage, and maintenance issues are vitally important to whether a system is ultimately perceived to be a success. To establish a broader benchmark of product success, you should add deployment, transition, and operational considerations to your product objectives. Product objectives must define the overall parameters of success for the system being developed.

In most cases, constraints must be considered in relation to product objectives. Managers generally do not have unlimited funds at their disposal, nor do they have unlimited time in which to complete work tasks. As a result, cost and schedule objectives must be considered alongside the product or service objectives, and in many cases are the key drivers of management's decision, especially as time passes and costs accrue.

These three types of objectives, when viewed together, typically define a basic set of objectives for a program. They specify what will be accomplished, the anticipated costs to complete all activities, and the timeframe in which work will be completed. When appropriate, these objectives can be supplemented with other objectives (such as business or financial objectives) to build an even broader picture of success. Once that picture is established, decision makers can focus their attention on making sure that results satisfy those objectives. Risk management is an activity that plays a vital role in achieving a program's objectives, and it is the focus of the next section.

[6] An operational process will have service objectives instead of product objectives. Service objectives define the nature of the services provided to the recipients of those services (i.e., customers). For example, if the service you are providing is help desk support, the service objectives will define the quality of help desk support provided to constituents (such as the required response time based on the priority of the request).

3 Two Fundamental Approaches for Managing Risk

The term *risk* is used universally, but different audiences often attach different meanings to it [Kloman 1990]. In fact, the details about risk and how it supports decision making depend upon the context in which it is applied [Charette 1990]. For example, safety professionals view risk management in terms of reducing the number of accidents and injuries. A hospital administrator views risk as part of the organization's quality assurance program, while the insurance industry relies on risk management techniques when setting insurance rates. Each industry thus uses a definition that is uniquely tailored to its perspective. As a result, no universally accepted definition of risk exists.

However, whereas specific definitions of risk might vary, a few characteristics are common to all definitions. In fact, for risk to exist in any circumstance, the following three conditions must be satisfied [Charette 1990]:

1. The potential for loss must exist.
2. Whether the risk will occur is not known with certainty; however a probability of occurrence can be determined.
3. Some choice or decision is required to deal with the risk.

These characteristics can be used to forge a very basic definition of the word *risk*. Most definitions focus on the first two conditions—loss and probability—because they are the two quantifiable aspects of risk. Bearing this in mind, the essence of risk, no matter what the domain, can be succinctly captured by the following definition: *Risk is the likelihood of suffering loss*. Put another way, risk is a measure of the likelihood that a threat will lead to a loss coupled with the magnitude of the loss.

3.1 Components of Risk

As illustrated in Figure 3, a risk can be thought of as a cause-and-effect pair, where the threat is the cause and the resulting consequence is the effect. In this context, *threat* is defined as a circumstance with the potential to produce loss, while a *consequence* is defined as the loss that will occur when a threat is realized.

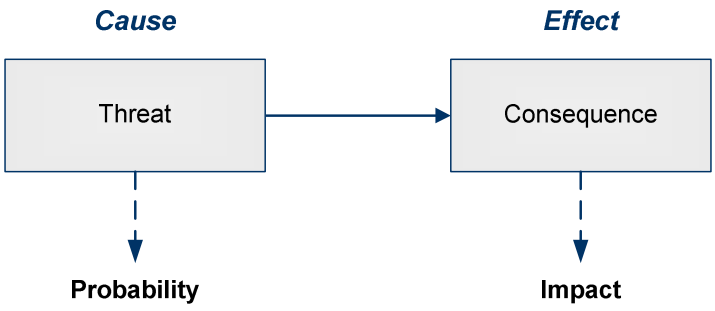

Figure 3 Components of Risk

Three measures are associated with a risk: (1) probability, (2) impact, and (3) risk exposure. The relationships between probability and impact and the components of risk are shown in Figure 3. In this context, *probability* is defined as a measure of the likelihood that a threat will occur, while *impact* is defined as a measure of the loss that will occur if the threat is realized. *Risk exposure* provides a measure of the magnitude of a risk based on current values of probability and impact.

No matter what approach is employed to manage risk, the core components of risk (threat and consequence) and its measures (probability, impact, and risk exposure) are universal. While the principles of risk management are time-tested and universal, decision makers have many options regarding *how* to manage their risk. The remainder of this section presents two basic approaches that can be employed when managing risk in software programs: (1) tactical approaches and (2) systemic approaches.

3.2 Tactical Risk Management

Tactical risk management views a threat as a potential event that might or might not occur and is focused on the *direct consequences* of that threat. This concept is illustrated in Figure 4. In this document, we define *tactical risk* as a measure of the likelihood that an individual potential event will lead to a loss coupled with the magnitude of the loss.[7] In most instances, a tactical risk will directly affect program performance; the impact on a program's key objectives is most often an *indirect consequence* of a tactical risk. With respect to the tactical risk highlighted in below, many additional events must occur before the objectives will be directly affected. In fact, Figure 4 depicts four separate potential events, which translates to four distinct risks that could be identified from a tactical point of view.

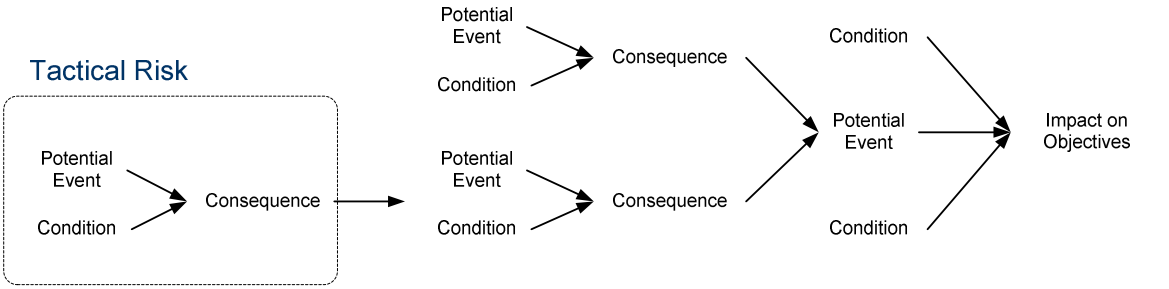

Figure 4 A Tactical View of Risk

The starting point for tactical risk management is the identification of all known risks that can adversely affect a program's performance. A separate statement is documented for each risk that is identified. Probability and impact are established for of each risk statement, and risk exposure is then determined from the individual values of probability and impact. Using this approach, the typical software program can easily identify hundreds of risk statements. Managing such a large number of risks can be challenging, thus, two strategies are often employed to meet this challenge.

[7] A potential event is the main focus of a tactical risk. However, as shown in Figure 4 each potential event in influenced by one or more current conditions and can also be influenced by the consequences triggered by other potential events.

The first strategy uses a Pareto analysis to identify the highest-priority risks. Decision makers use the values of probability, impact, and risk exposure to sort the list of risks. They then focus their attention on managing the top 10-20% of the risks on the list. The remainder of the risks should be reviewed periodically for changes. However, in practical terms, risks that do not rank in the top 10-20% are rarely revisited. Decision makers often miss changes in their program's risks despite the reality that risk measures for all risks change over time. Low-priority risks might become more important over time, while high-priority risks might become less important. These changes can be missed if only a small percentage risks are being actively managed.

The second strategy is to use a technique like affinity grouping to categorize risks into groups. An overarching risk statement can be documented for each risk group, and the risk measures for each risk group can then evaluated.[8] By assigning tactical risks to groups, decision makers can focus their management actions on the groups instead of each individual risk. The drawbacks to this strategy are (1) the amount of time needed to categorize risks when a large number of risk statements have been identified and (2) the potential inconsistency of categories across partners in a multi-enterprise program.

From the tactical point of view, each risk provides a detailed piece of information about a potential loss. To create a big-picture view of a program's risk, you must aggregate detailed risk information. Because tactical approaches rely on aggregation techniques to provide a big-picture view of risk, we refer to them in this document as incorporating a *bottom-up analysis*.

Tactical approaches normally lead to the development of many distinct point solutions, where each is intended to mitigate a specific risk statement. In practice, coordination of numerous point solutions across a large program has proven to be difficult and can lead to inefficient use of resources and ineffective mitigation of risk.

Some programs have been successful employing tactical approaches for managing risk. However, many struggle to effectively manage high numbers of risk statements. In some cases, decision makers in these programs spend too much time manipulating and analyzing risk statements and too little time actually managing their risk.

3.3 Systemic Risk Management

Systemic risk management is an alternative approach that is the focus of our current research and development activities. As shown in Figure 5, systemic approaches assume a holistic view of risk to objectives by examining the *aggregate effects* of multiple conditions and potential events on a program's key objectives. A systemic approach for managing risk thus starts at the top—with the identification of a program's key objectives. Once the key objectives have been explicitly articulated, a set of drivers that influence the outcome are identified. As used in this way, the term *driver* is defined as a factor that has a strong influence on the eventual outcome or result (i.e., on whether or not key objectives will be achieved).

[8] Some tactical approaches assign an overarching risk statement for each risk group and then evaluate probability, impact, and risk exposure for each group. Other approaches do not assign an overarching risk statement and do not evaluate risk measures for each group; risk groups are only used during risk mitigation to identify high-leverage mitigation solutions.

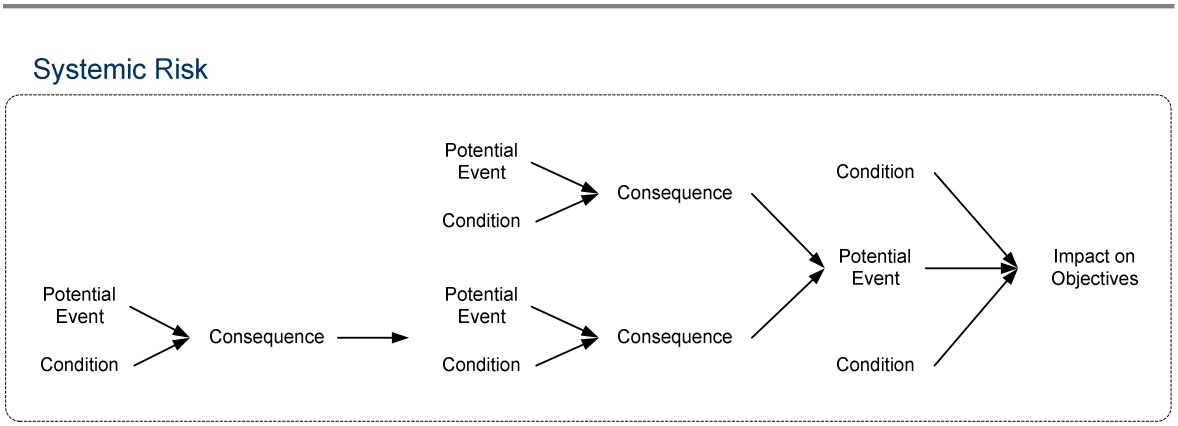

Figure 5 A Systemic View of Risk

Drivers are important because they define a small set of factors that a manager can use to determine whether a program is on track to achieve its key objectives. A manager can focus his or her attention on a set of 10-20 drivers at any given time. (See Section 5.3 for a starter set of program drivers.) A typical set of program drivers will address a broad range of factors, such as whether

- the program's objectives are realistic and achievable
- the plan for developing and deploying the system is sufficient
- tasks and activities are performed effectively and efficiently
- the program complies with all relevant policies, laws, and regulations
- the program has sufficient capacity and capability to identify and manage potential events and changing circumstances
- the system being developed will effectively support operation
- users will be prepared to operate the system being developed

When you analyze a set of drivers, you analyze how conditions and potential events are influencing each driver. To accurately assess a given driver, you must consider which conditions and potential events have a positive influence on that driver as well as which have a negative influence on it. In this way, you can establish the driver's current state and then determine how it is currently influencing the outcome. The relationship among key objectives, drivers, conditions, and potential events is shown in Figure 6.

Drivers provide a means of translating vast amounts of detailed data about current conditions and potential events into useable information that supports program decision making. One of the main advantages of employing a systemic approach is the traceability among key objectives, drivers, conditions, and potential events. This traceability is extremely useful when making tradeoff decisions and when looking for high leverage ways to mitigate a program's risks.

The goal of systemic risk management is to assess and manage the risk triggered by each driver. Systemic approaches tend to be easier to perform than tactical approaches, especially when applied to complex environments. Our research indicates that systemic approaches scale to distributed management

environments more readily than do tactical approaches (see Section 8 for more information on scaling systemic approaches to distributed environments).

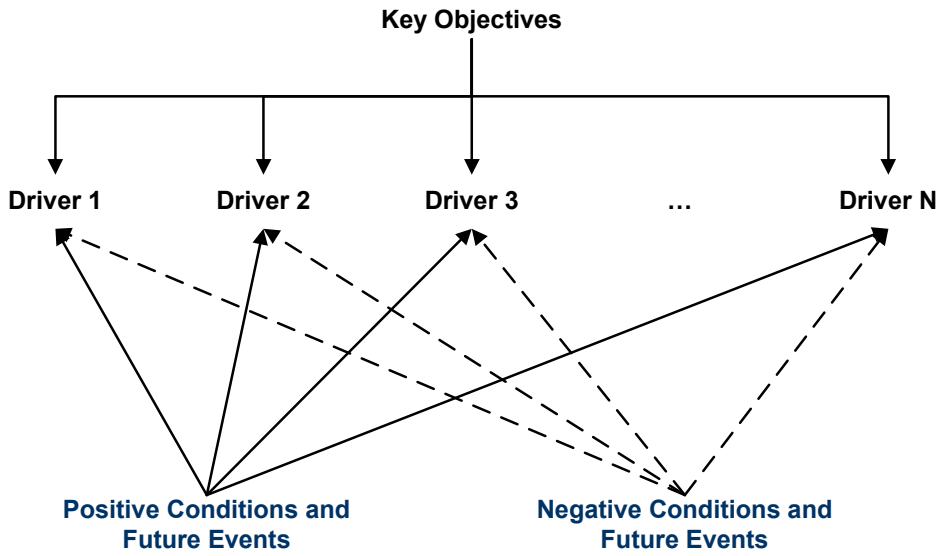

Figure 6 Relationships Among Key Objectives, Drivers, Conditions, and Potential Events

From the systemic point of view, the risk triggered by each driver provides an aggregate view of potential loss. To get more detailed information about the root causes of a risk, you need to perform additional analysis. Because systemic approaches begin with the big-picture view of risk, we refer to them in this document as incorporating a *top-down analysis*.

When you employ a systemic approach for managing risk, you need to make sure that the set of drivers appropriately reflects your key objectives and management context (e.g., current life-cycle phase). You also need to look for indications of outlying issues that do not map to any existing drivers. These outliers might indicate the need to add an additional driver to the set you are managing. If you do not adjust the set of drivers over time, you might miss key indications of risk.

Our recent research has focused on developing and piloting Mosaic, an approach for implementing systemic risk management. The focal point of Mosaic is the identification and analysis of a set of drivers. The next several sections take a closer look at the nature of drivers.

4 Driver Framework

One goal of our research has been to develop a way to derive a set of drivers for the unique requirements of each program. While working toward this goal, we analyzed the results of past SEI risk management research that cataloged sources of risk in software development [Dorofee 1996, Williams 1999], system acquisition [Gallagher 1999], and operational security [Alberts 2002]. Our analysis showed similarities and patterns among the types of conditions and events that produced risk in each setting. The result of our analysis was the development of a common structure, or framework, for classifying a set of drivers that influence a program's outcome. As illustrated in Figure 7, the driver framework[9] comprises six categories:

- objectives
- preparation
- execution
- environment
- resilience
- result

Each category is examined in this section.

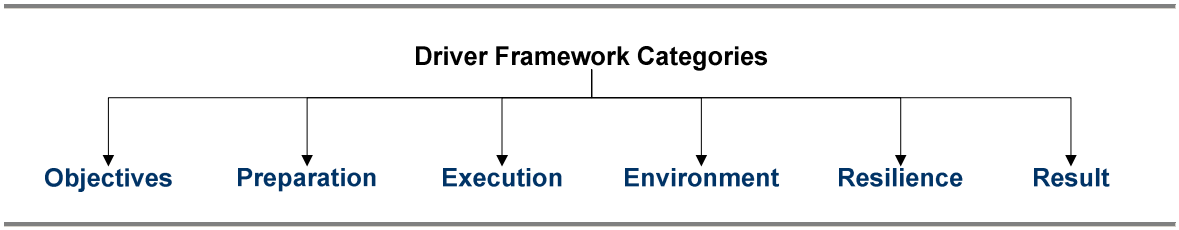

Figure 7 Six Categories of the Driver Framework

4.1 Objectives

In Section 2, the topic of a program's objectives was addressed. We defined an objective as a desired result or outcome of a program. Put another way, a set of objectives defines the mission being pursued by a program. As discussed in Section 2, a program typically focuses on three distinct types of key objectives: (1) product, (2) cost, and (3) schedule. Other key objectives, such as business or financial objectives, can be added as appropriate when defining a program's picture of success. For a program to achieve success, all of its key objectives must be balanced, realistic, and achievable. Issues with key objectives can have a profound effect on a program's potential for success. Some programs, for example, begin with extremely aggressive schedules, poor funding, and high-risk technology, leaving them unable to deal with any adverse events. Because of its impact on program success, the first, and most

[9] The driver framework can also be referred to as the OPEERR (pronounced *oh-peer*) framework. The acronym uses the first letter from each driver category.

fundamental, category of drivers is *objectives*. The drivers in this category are focused on the purpose and scope of a program.

4.2 Preparation

Whereas the set of key objectives defines what success looks like, preparation provides the roadmap for achieving that picture of success. With respect to the driver framework, preparation focuses on the processes and plans required to achieve objectives. Preparation activities typically include

- outlining people's roles and responsibilities
- ensuring that activities are sequenced correctly
- identifying dependencies and interrelationships among activities
- defining the processes used by the program
- establishing practices and procedures that must be followed
- providing artifacts, such as decision-making guidelines, templates, and written procedures
- defining technologies that are needed to support the program
- establishing measures and metrics for managing the program

Objectives and preparation, when viewed together, define a plan of action and structure for a program. This foundation provides the blueprint for conducting program tasks and activities.

4.3 Execution

While objectives and preparation provide the foundation for the program, execution examines how tasks and activities are managed and performed. The management of these activities is focused on assembling, organizing, and overseeing the assets required to bring that plan to life. Examples of assets include

- people tasked with doing the work
- technology and equipment directly supporting program execution
- information used to support execution of tasks and activities
- facilities in which the work will be completed

The performance of these activities is focused on the effective and efficient completion of assigned activities, including, for example, whether people actually follow defined processes and how well tasks are coordinated across groups.

4.4 Environment

Ideally, management and staff could focus exclusively on the tasks at hand and ignore how the broader environment affects program performance. However, the environment typically plays a major role in how efficiently and effectively activities are performed. Management and staff must be aware of their surroundings and understand how environmental conditions affect their work tasks. The program's environment includes

- organizational structure
- culture
- politics
- communication infrastructure

Also included are any constraints that a program inherits from its parent organization(s) or from the broader business environment. Constraints can include restrictions imposed by laws and regulations as well as limitations with services provided by third parties.

4.5 Resilience

Thus far, the driver categories have been focused on what it takes to plan and execute a program based on current and known conditions. However, effective management must also take into account the possibility of problems resulting from potential events and changing circumstances. A program must be nimble enough, or *resilient*, to adapt to a range of potential events. In some cases, events can be anticipated and planned for. In others, people must be able to quickly respond and take timely action to avoid fallout from unexpected events.

So, we define resilience as the ability to effectively manage potential events and changing circumstances. It is an important aspect of program management because it enables people to handle a variety of situations that can arise and ultimately place program objectives at risk.

4.6 Result

For a software program, the term *result* refers to the correctness and completeness of the software-intensive system or system of systems that is being developed (i.e., the *product* that is developed).[10] In particular, drivers in this category examine whether the product will be successfully operated, used, and maintained in its operational environment. The types of issues addressed by this category include

- the extent to which requirements are understood
- whether functional and nonfunctional requirements will be satisfied
- sufficiency of the design and architecture
- degree of integration and interoperability with other systems
- ability to support operations
- adoption barriers
- certification and accreditation of the system

4.7 Primary Relationships Among Driver Categories

The six driver categories define a broad range of issues that should be considered when managing a software program. From our piloting activities, we have noticed core relationships among the driver categories. These relationships are illustrated in Figure 8.

[10] If you apply this framework to the delivery of a service, the result category addresses the ability to provide quality and timely services that meet customer's needs.

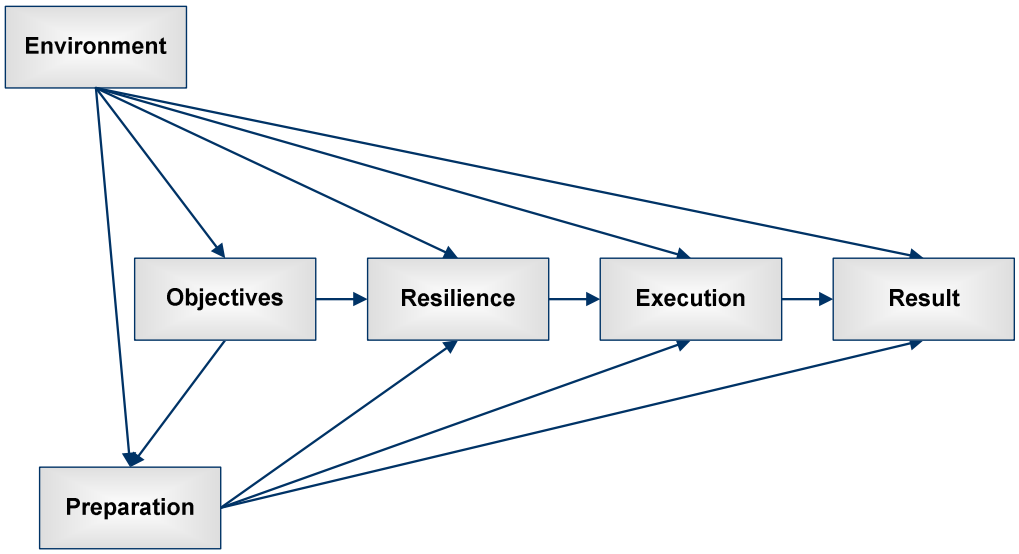

Figure 8 Relationships Among the Driver Categories

The relationships among driver categories can be useful when determining the root causes of a risk. For example, suppose that you have determined that you will likely have trouble integrating the system you are developing with other operational systems. Product issues, like a potential integration problem, belong to the *result* category. However, the program plan might not have allocated sufficient time for integration testing, which shifts the focus to *preparation*. Similarly, planning for integration testing might have been shortened because the schedule was reduced during contracting negotiations. Here, the focus broadens to include *objectives*. In turn, the schedule might have been reduced because a sponsor arbitrarily decided to do so, which adds a source from the *environment* category. Drivers in categories that influence other categories can be viewed as leading indicators of success or failure.[11] In this example, the arbitrary reduction in schedule was a leading indicator of potential problems for the program.

[11] In this report, a *leading indicator* is defined as a factor that provides insight into future performance.

5 Driver Identification

We originally focused our research and development efforts on driver identification and analysis because we needed an efficient and effective means of evaluating a program's current state. Once you establish the current state, you can then employ a variety of back-end analysis methods to interpret the results and chart a course for improvement. In our research, we have applied several back-end analysis methods, ranging from basic gap analysis to integrated risk and opportunity analysis. Each back-end analysis provides a different view of how well a program is currently performing and provides a means of inferring how well that program might perform in the future. No matter which back-end analysis you decide to employ, the first step is to ensure that you are collecting the right data. With a driver-based approach, collecting the right data requires you to first identify the correct set of drivers.

5.1 Driver Attributes

A driver is a factor that has a strong influence on the eventual outcome or result, that is, on whether or not key objectives will be achieved. Each driver comprises four key attributes, *name*, *success state*, *failure state*, *category*, which are described in more detail in Table 1.

Table 1 Example of an Attribute Table for a Driver

Attribute	Description	Example
Name	A concise label that describes the basic nature of the driver	Process
Success State	A driver exerts a positive influence on the outcome	The process being used to develop and deploy the system is sufficient.
Failure State	A driver exerts a negative influence on the outcome	The process being used to develop and deploy the system is insufficient.
Category	The category to which the driver belongs	Preparation

Each driver also has two possible states—a success state and a failure state. When analyzing a driver, you determine how it is currently acting (i.e., its current state) by examining the effects of conditions and potential events on that driver. The goal is to determine if the driver is

- almost certainly in its success state
- most likely in its success state
- equally likely in its success or failure states
- most likely in its failure state
- almost certainly in its failure state

By analyzing each driver in a set, you establish a benchmark of the program's current state.

5.2 Deriving a Set of Drivers

The starting point for identifying a set of drivers is determining the program's key objectives. (See Section 2.2 for a discussion of key objectives.) Once key objectives have been articulated, a set of drivers can be derived from them. This relationship between drivers and key objectives is depicted in Figure 9.

To establish a set of drivers for specific objectives, you need information from people with experience and expertise relevant to those objectives. For example, if you want to identify a set of drivers for software development, you would obtain information from people who manage software programs and who develop software-intensive systems. Similarly, if you are looking to establish a set of drivers for organizational security, you would consult with security experts.

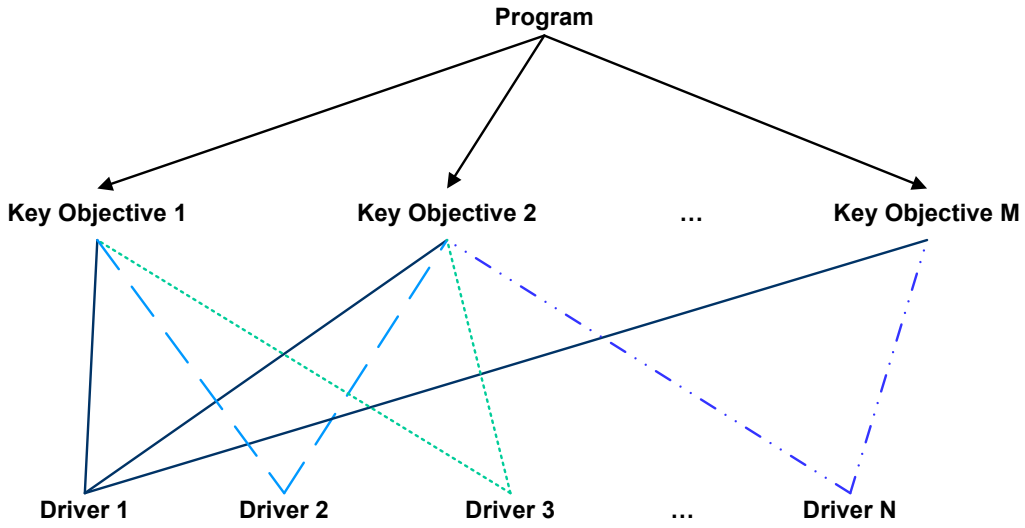

Figure 9 Relationship Between Key Objectives and Drivers

The experts from whom you elicit information should be familiar with the key objectives that have been defined. You can use the key objectives to focus your discussion with them. The experts need to answer the following questions:

- What circumstances, conditions, and events will drive your program toward a *successful* outcome?
- What circumstances, conditions, and events will driver your program toward a *failed* outcome?

After you have gathered information from the experts, you need to organize the information they provided (i.e., circumstances, conditions, and events) into approximately 10-20 groups that share a central idea or theme. The driver is the central idea or theme of each group. You then need to define the four attributes of each driver. (See Table 1 for descriptions of the driver attributes.) The main rule when compiling a set of drivers is to make sure you include at least one driver for each of the six driver categories. Analyzing drivers from all categories will help to ensure an adequate breadth of data collection.

We have employed this approach to identify a set of drivers in a variety of areas, including software acquisition and development programs, cyber-security processes, and business portfolio management.

Our recent focus has been on software acquisition and development programs. Over time, we have been able to identify patterns in the driver sets we have used to assess a range of software programs. Analysis of these patterns enabled us to establish a starter set of drivers for software programs. This starter set of drivers provides a basis from which you can tailor a set of drivers to a given program's unique environment. The next section presents a set of drivers you can use as a starting point for tailoring activities.

5.3 A Starter Set of Drivers

The starter set of 20 drivers for software programs, and their attributes, is documented in Table 2. These drivers in are based on the typical set of objectives for a program: product, cost, and schedule.

Table 2 Starter Set of Drivers for Software Programs

Driver Name	Success State	Failure State	Category
1. Program Objectives	Program objectives (product, cost, schedule) are realistic and achievable.	Program objectives (product, cost, schedule) are unrealistic or unachievable.	Objectives
2. Plan	The plan for developing and deploying the system is sufficient.	The plan for developing and deploying the system is insufficient.	Preparation
3. Process	The process being used to develop and deploy the system is sufficient.	The process being used to develop and deploy the system is insufficient.	Preparation
4. Task Execution	Tasks and activities are performed effectively and efficiently.	Tasks and activities are performed ineffectively and inefficiently.	Execution
5. Coordination	Activities within each team and across teams are coordinated appropriately.	Activities within each team and across teams are not coordinated appropriately.	Execution
6. External Interfaces	Work products from suppliers, partners, or collaborators will meet the program's quality and timeliness requirements.	Work products from suppliers, partners, or collaborators will not meet the program's quality and timeliness requirements.	Execution
7. Information Management	The program's information is managed appropriately.	The program's information is not managed appropriately.	Execution
8. Technology	The program team has the tools and technologies it needs to develop the system and transition it to operations.	The program team does not have the tools and technologies it needs to develop the system and transition it to operations.	Execution
9. Facilities and Equipment	Facilities and equipment are sufficient to support the program.	Facilities and equipment are insufficient to support the program.	Execution
10. Organizational Conditions	Enterprise, organizational, and political conditions are facilitating completion of program activities.	Enterprise, organizational, and political conditions are hindering completion of program activities.	Environment

Driver Name	Success State	Failure State	Category
11. Compliance	The program complies with all relevant policies, laws, and regulations.	The program does not comply with all relevant policies, laws, and regulations.	Environment
12. Event Management	The program has sufficient capacity and capability to identify and manage potential events and changing circumstances.	The program has insufficient capacity and capability to identify and manage potential events and changing circumstances.	Resilience
13. Requirements	System requirements are well understood.	System requirements are not well understood.	Result
14. Design and Architecture	The design and architecture are sufficient to meet system requirements and provide the desired operational capability.	The design and architecture are insufficient to meet system requirements and provide the desired operational capability.	Result
15. System Capability	The system will satisfactorily meet its requirements.	The system will not satisfactorily meet its requirements.	Result
16. System Integration	The system will sufficiently integrate and interoperate with other systems when deployed.	The system will not sufficiently integrate and interoperate with other systems when deployed.	Result
17. Operational Support	The system will effectively support operations.	The system will not effectively support operations.	Result
18. Adoption Barriers	Barriers to customer/user adoption of the system have been managed appropriately.	Barriers to customer/user adoption of the system have not been managed appropriately.	Result
19. Operational Preparedness	People will be prepared to operate, use, and maintain the system.	People will not be prepared to operate, use, and maintain the system.	Result
20. Certification and Accreditation	The system will be appropriately certified and accredited for operational use.	The system will not be appropriately certified and accredited for operational use.	Result

5.4 Tailoring an Existing Set of Drivers

The starter set of drivers provides a basic set that you can use to assess a software program. You will need to tailor the set to ensure that the

1. set of drivers accurately reflects the key objectives of the specific program you are assessing
2. set of drivers is adjusted appropriately based on the program's context and characteristics
3. phrasing of each driver is consistent with the program's terminology

The first step when tailoring an existing set of drivers is to establish the program's key objectives. Once the program's objectives are clearly articulated, you then select a predefined set of drivers consistent with those objectives to use as the basis for tailoring. Prior to tailoring the drivers, you should meet with

management and staff from the program to learn about what the program is trying to accomplish and to gain an appreciation for its unique context and characteristics.

After you have developed a basic understanding of the program, you are ready to tailor the drivers. Review the predefined set of drivers that you are using as the starting point for tailoring activities. Based on the program's key objectives and the data that you have gathered,

- Determine which drivers do not apply to the program. Eliminate extraneous drivers from the set.[12]
- Establish whether any drivers are missing from the list. Add those drivers to the set.
- Decide if multiple drivers from the set should be combined into a single, high-level driver. Replace those drivers with a single driver that combines them.
- Decide if any drivers should be decomposed into multiple, more detailed drivers. Decompose each of those drivers into multiple drivers.

Finally, adjust the wording of each driver attribute to be consistent with the program's terminology and language. At this point, you will have a set of drivers that can be used to assess the program's current state.

[12] Remember that you need to ensure that you include at least one driver from each driver categories.

6 Driver Analysis

The previous section discussed the importance of developing or tailoring a set of drivers for the unique needs of each program. This section builds on the concept of drivers by examining how to analyze an individual driver's current state.

6.1 Assessing a Driver's Current State

The goal of driver analysis is to determine how each driver is influencing a program's key objectives. More specifically, you need to establish the probabilities that each driver is in its success and failure states. Any of the following four approaches can be used as the basis for driver analysis:

1. Convert each driver into a yes/no question, where each question is phrased from the *success perspective*. Each driver question is then answered based on the available information about the program.
2. Convert each driver into a yes/no question, where each question is phrased from the *failure perspective*. Each driver question is then answered based on the available information about the program.
3. Use the driver's *success state* as a true/false statement. Each statement is then evaluated based on the available information about the program.
4. Use the driver's *failure state* as a true/false statement. Each statement is then evaluated based on the available information about the program.

You should choose the approach that best suits your needs. When we conduct driver analysis, we normally convert drivers into questions that are phrased from the success perspective. Table 3 provides an example question for the *Process* driver from one of our surveys. This example will be used throughout this section when discussing driver analysis.

Table 3 Driver Question and Range of Responses

Driver Question	Answer				
3. Is the process being used to develop and deploy the system sufficient? Consider: process design; measurements and controls; process efficiency and effectiveness; acquisition and development life cycles; training	No	Likely no	Equally likely	Likely yes	Yes
	❑	❑	❑	❑	❑

Because the question in the figure is phrased from the success perspective, an answer of *yes* indicates the driver is in its success state and an answer of *no* indicates it is in its failure state. We use a range of answers to capture probabilities (likely yes, equally likely yes or no, likely no) when the answer is not a definitive yes or no. In addition, we often include key items to consider when answering each question.

These considerations highlight important areas to think about when answering the question. A complete questionnaire for the starter set of drivers can be found in Appendix A.

A set of *driver value criteria*, such as those shown in Table 4, are normally used to support driver analysis. Driver value criteria serve two main purposes:

- They provide a definition of applicable responses to a driver question.
- They translate each response into the probability that the driver is in its success state as well as the probability that it is in its failure state.

Table 4 Driver Value Criteria

Answer	Definition	Values	
		Probability of Success State	Probability of Failure State
Yes	The answer is almost certainly "yes." Almost no uncertainty exists. There is little or no probability that the answer could be "no." (~ > 95% probability of yes)	Maximum	Minimal
Likely yes	The answer is most likely "yes." There is some chance that the answer could be "no." (~ 75% probability of yes)	High	Low
Equally likely	The answer is just as likely to be "yes" or "no." (~ 50% probability of yes)	Medium	Medium
Likely no	The answer is most likely "no." There is some chance that the answer could be "yes." (~ 25% probability of yes)	Low	High
No	The answer is almost certainly "no." Almost no uncertainty exists. There is little or no probability that the answer could be "yes." (~ < 5% probability of yes)	Minimal	Maximum

The criteria for analyzing a driver must be tailored for each application of driver analysis. For example, the criteria in Table 4 are based on a five-point scale. This type of scale allows decision-makers to incorporate different levels of probability in their answers. More or less than five answers can be incorporated into the analysis when appropriate. In addition, some people prefer to include a response of *don't know* to highlight those instances where more information or investigation is needed before a driver can be analyzed appropriately.

When you analyze a driver, you need to consider how conditions and potential events are affecting that driver. In general, you should think about the following items for each driver you analyze:

- positive conditions that support an answer of yes
- negative conditions that support an answer of no

- potential events with positive consequences that support an answer of yes
- potential events with negative consequences that support an answer of no

The table in Figure 10 shows an example of an analyzed driver. The answer to the driver question is *likely no*. This means the program's objectives are most likely unrealistic or unachievable, that is, most likely in its failure state. Rationale for each response is also documented. The rationale captures the reasons underlying the response to a driver question (e.g., which conditions and potential events are steering the driver toward its success and failure states).

Question	Answer				
3. Is the process being used to develop and deploy the system sufficient? **Consider:** process design; measurements and controls; process efficiency and effectiveness; acquisition and development life cycles; training	No	Likely no	Equally likely	Likely yes	Yes
	❑	☑	❑	❑	❑

Rationale

+ Previous programs have a 90% history of delivering on-time.

- The process for integration testing is likely inadequate. Historically, integration testing has used "verbal" agreements between a few managers who already know each other. With this system, there are managers and team leads who have never worked together and there are other barriers in place that make "verbal" agreements tenuous.

- There are a lot of brand new programmers (45%).

- This program required a significant change in our standard processes. There was no new training created for the new processes.

- QA did not have a chance to review the new and revised processes before they were put into practice.

- The person who developed the new processes quit last week.

Figure 10 Assessed Driver

6.2 A Snapshot of Current Conditions

A *driver profile* provides a snapshot, or summary, of all drivers relevant to a program. Figure 11 provides an example of a driver profile. In the figure, a bar graph is used to show twenty drivers, which correspond to the starter set of drivers.

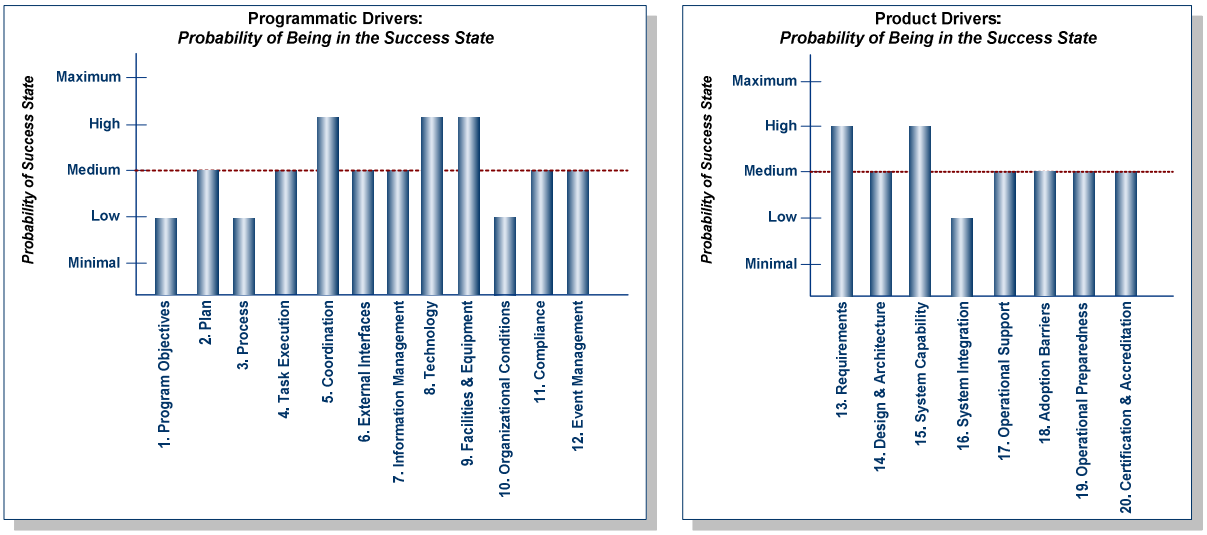

Figure 11 Driver Profile[13]

The graph depicts the probability that each driver is in its success state. In addition, programmatic drivers are separated from the product drivers. A driver profile is useful because it provides a snapshot of current conditions. The profile in Figure 11 indicates that the following four drivers are likely in their failure states: program objectives, process, organizational conditions, and system integration. These drivers should concern the program's decision makers. However, to better articulate those concerns, decision makers typically perform additional analysis. Several options for follow-on analysis are presented in the next section.

[13] The programmatic drivers map to drivers from the objectives, preparation, execution, environment, and resilience categories. The product drivers map to drivers from the result category.

7 Using Drivers and the Driver Framework

Driver analysis gathers numerous data about a program and produces a concise snapshot of that program's current state. However, decision makers generally often perform additional analysis to more fully understand how the current state will likely affect a program's key objectives. As shown in Figure 12, once you establish the current state using driver analysis, you can then use a variety of back-end analyses to interpret the results.

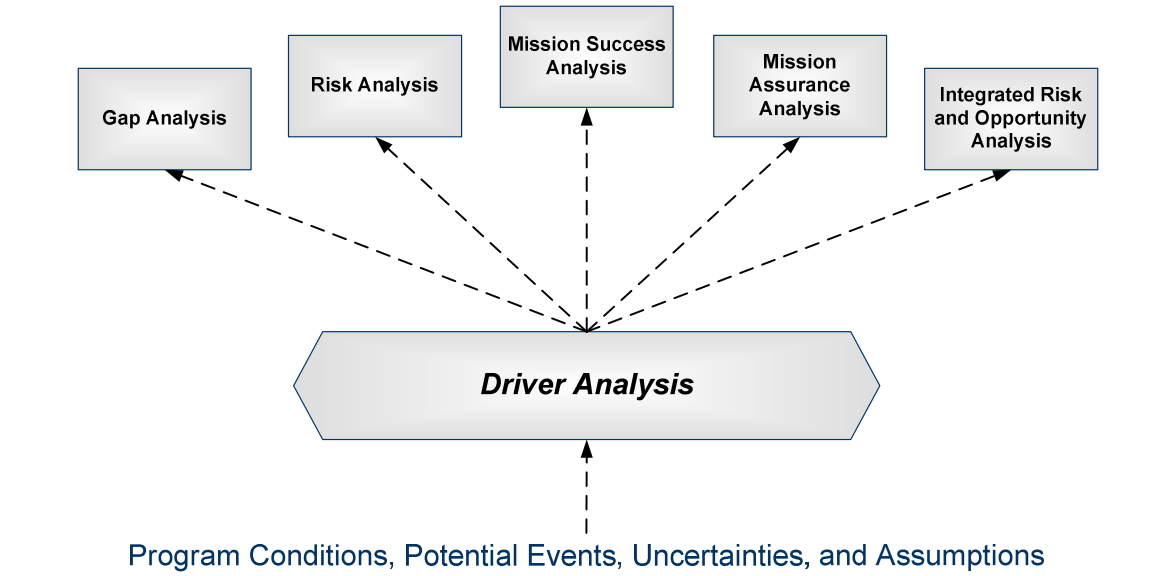

Figure 12 Multiple Views into a Program

7.1 An Overview of Back-End Analyses

Decision makers have many options regarding which back-end analysis they use, ranging from a very basic gap analysis to an integrated analysis of risk and opportunity. The ultimate goal is to ensure that you have data that are sufficient to support effective decision making. Table 5 provides a brief summary of the back-end analyses that we have used in conjunction with driver analysis.

Table 5 Additional Back-End Analyses

Gap Analysis

A gap analysis examines the difference between current and desired states of each driver. It can be performed very quickly and does not require specialized skills to conduct. Gap analysis can help decision makers identify basic concerns about their chances of success. However, because gap analysis does not analyze the consequences of those concerns, decision makers might not have all of the information they need to make appropriate tradeoffs.

Risk Analysis

Risk analysis examines the potential loss produced by each driver in relation to the program's key objectives. Most decision makers are familiar with risk analysis, so they can easily learn to use it as a follow-on to driver analysis. Applying risk analysis to driver results is presented in more detail in Section 7.2.

Mission Success Analysis

A mission success analysis uses scenarios to determine the probability of successfully achieving each key objective. This type of predictive analysis requires substantial experience and expertise to develop relevant scenarios and then assess them. It can take considerable time to conduct and is difficult to automate. However, it presents results in scenario form, which many decision makers find useful.

Mission Assurance Analysis

This type of analysis establishes a measure of mission assurance for each driver based on risk and uncertainty. It is very useful for establishing justifiable confidence that key objectives will be achieved. Mission assurance analysis can require a considerable time investment to collect and analyze data. However, it is very useful for when analyzing processes and systems that are mission critical.

Integrated Risk and Opportunity Analysis

An opportunity is the likelihood of realizing a gain resulting from an allocation (or reallocation) of resources. Integrated risk and opportunity analysis examines the risks and opportunities inherent in a situation and helps decision makers strike an appropriate balance between the two. Integrated risk and opportunity analysis requires considerable experience and expertise to conduct. However, it is extremely useful for enabling decision makers to look beyond product, cost, and schedule objectives when making tradeoffs.

All of the analyses in Table 5 provide decision makers with useful decision-making data. Of the five approaches featured in the table, risk analysis is by far the one that is most commonly used by decision makers in programs and organizations. The next section looks at risk analysis in more detail.

7.2 Risk Analysis

We use the term *mission risk*[14] when referring to the risk produced by each driver. Used in this context *mission risk* is defined as a measure of potential loss in relation to key objectives. As described in Section 3.1, all risks comprise the two components, threat and consequence; Figure 13 shows how these components apply to mission risk. The failure state of the driver acts as the threat because it defines a circumstance with the potential to produce loss. The consequence of a mission risk is the negative impact on a program's key objectives triggered by a driver's failure state.

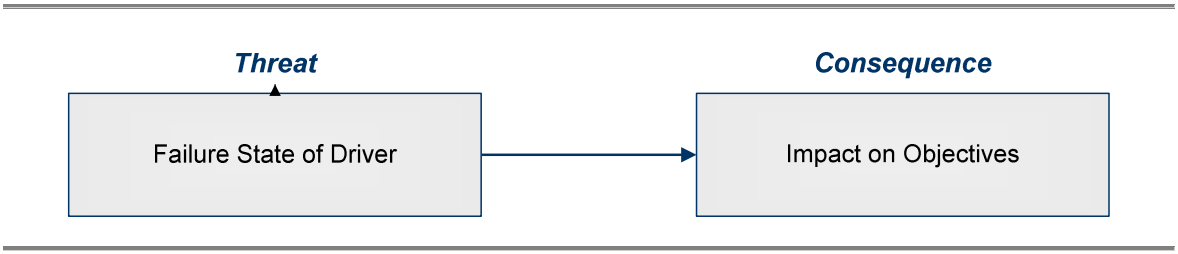

Figure 13 Components of Mission Risk

The first step when managing any type of risk is to effectively articulate, or communicate, the basic concern underlying that risk. A risk statement provides a unique and succinct description of a risk and is commonly used as the main construct for articulating a risk. Figure 14 provides an example of a risk statement and its associated measures for a mission risk (the risk produced by the *Process* driver that was featured in Section 6.1).

Risk Statement	Probability	Impact	Risk Exposure
3. The process being used to develop and deploy the system is insufficient.	High	Severe	High

Figure 14 Example of a Mission Risk and its Measures

As shown in Figure 14, the risk statement **is** the failure state of the driver.[15] Risk probability is listed as *high*, which corresponds to the response of *likely no* that was selected during driver analysis. (Refer to the driver value criteria in Table 4 to see the relationship between the responses to driver questions and their associated probabilities.)

[14] A *mission risk* can also be referred to as a *systemic risk*.

[15] A risk statement often includes both the threat and consequence components of a risk. The *direct consequence* for all mission risks, by definition, is that the program will not be able to achieve one or more of its key objectives. Because it does not uniquely differentiate one mission risk from another, consequence can be omitted from the risk statement for a mission risk, if desired. However, for completeness, some people prefer to include the consequence in the risk statement for a mission risk.

Risk impact in the example is evaluated as *severe* because the driver has a very strong influence on key objectives. In other words, if the process for developing and deploying the system were determined to be insufficient, then the negative impact on the program's key objectives will be extremely large. Finally, risk exposure combines the values of probability and impact to establish the magnitude of a risk. As shown in Figure 14, the risk exposure is *high*.[16]

7.3 Risk Profile

A *risk profile* provides a summary of all risks relevant to a program at a specific point in time. It can be a tabular listing of risk statements (similar to the table in Figure 14) or a graphical portrayal of risk information. A risk profile is useful because it provides decision makers with a succinct snapshot of their current risks, which helps them to make appropriate tradeoffs and establish mitigation priorities. Figure 15 depicts a graphical risk profile that is based on the driver framework and the risk exposure for each driver (i.e., for each mission risk).

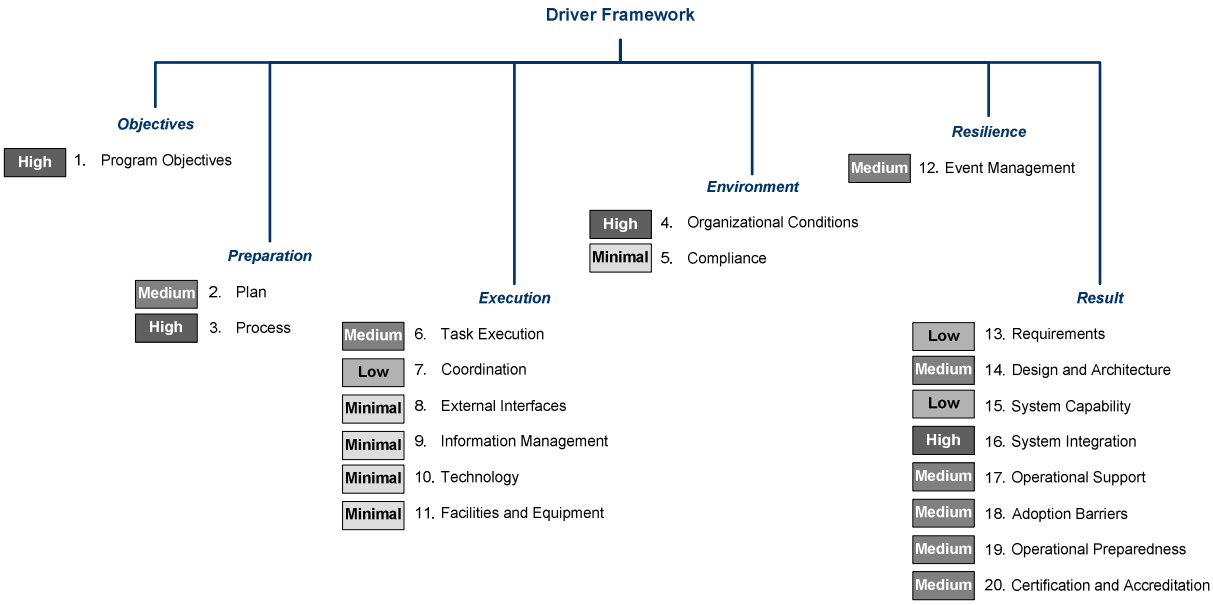

Figure 15 Example Risk Profile

By looking at the risk profile like the one shown in the figure, decision makers can quickly determine the general health of a program and begin to establish mitigation priorities. This profile clearly indicates that the program's biggest risks are in the following areas: program objectives, process, organizational conditions, and system integration. In contrast, external interfaces, information management, technolo-

[16] *To assess impact*, you must first define a set of risk impact criteria. Similar to the driver value criteria shown in Table 4, risk impact criteria define a set of measures (e.g., severe, high, medium, low, minimal) that can be used to evaluate the severity of a risk's impact. *To assess risk exposure*, you must first define a risk exposure matrix, which defines a set of measures (e.g., severe, high, medium, low, minimal) based on the individual values of probability and impact. Since these are standard risk management practices; the details are omitted from this discussion. More information can be found in the *Continuous Risk Management Guidebook* [Dorofee 1996].

gy, facilities and equipment, and compliance pose minimal risk to the program. Decision makers can now focus their attention on the program's high-risk areas.

7.4 An Integrated View of Tactical Data

Most programs currently employ tactical approaches to manage risk, which tend to produce a large number of risks that must be addressed. In addition, some of these programs augment their risk management approaches by including issue management, sometimes referred to problem management. In this context, an *issue* is defined as a loss that has occurred or is certain to occur, that is no uncertainty exists. Programs that explicitly identify and track issues usually end up with a large number of issues that must be managed. In the field, we have seen many programs create confusing definitions for the terms *risk* and *issue*. When definitions are ambiguous or confusing, people find it difficult to determine whether something is a risk or an issue. This confusion and ambiguity can ultimately affect people's decisions and actions, which can put a program at even greater risk.

To complicate matters further, many programs are now being asked to manage opportunities in addition to risks and issues, where an opportunity is viewed as a potential gain. Some programs are attempting to extend their tactical approaches to address opportunity management, and as a result, are now beginning to identify numerous tactical opportunities[17] in addition the vast numbers of risks and issues that they are already managing. Our field experience indicates that many programs are struggling to effectively manage risks, issues, and opportunities using tactical approaches.

Systemic approaches are beginning to show promise as a means of integrating tactical data. As mentioned in Section 6.1, the following data is normally documented when a driver is analyzed:

- positive conditions that steer a driver toward its success state (i.e., strengths)
- negative conditions that steer a driver toward its failure state (i.e., weaknesses/issues/problems)
- potential events with positive consequences that could steer a driver toward its success state (i.e., tactical opportunities)
- potential events with negative consequences that could steer a driver toward its failure state (i.e., tactical risks)

Although the main focus of a systemic approach is to manage risk at the mission level (i.e., risks to key objectives), it also is useful for organizing a program's tactical data. As shown in Figure 16, each driver integrates strengths, issues, tactical risks, and tactical opportunities affecting that driver.

---

[17] In this document, we make a distinction between a tactical opportunity and a mission opportunity. A *tactical opportunity* is a circumstance that has the potential to improve *program performance*; it does not necessarily translate to a gain from the business or mission perspective. For example, a new practice might enable you to complete a task more effectively or efficiently (i.e., a potential to improve performance). However, it might not improve the expected return on investment or improve the operational capability being developed, which are examples of gains from the business or mission perspectives. In contrast, a *mission opportunity* is a circumstance that has the potential to provide a gain from the business or mission perspective. The integrated risk and opportunity analysis described in Table 5 is focused on mission risk and mission opportunity (as opposed to tactical risk and tactical opportunity). Because tactical risks and tactical opportunities influence a driver's state, they can be managed using any of the driver-based analyses in Table 5. See Figure 16 for a diagram that shows how tactical risks and tactical opportunities influence a driver.

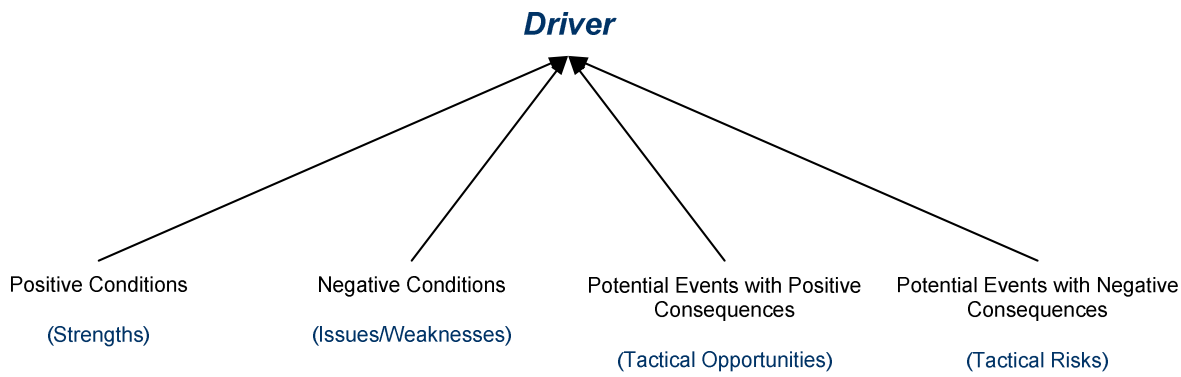

Figure 16 Strengths, Weaknesses, and Tactical Data Affecting Drivers

Driver identification and analysis provides a foundation for a variety of back-end analyses. This flexibility enables managers to customize a risk-based practice focused on their needs and requirements. The inherent flexibility of the approach is further explored in the next section, which looks at applying driver identification and analysis in multi-enterprise environments.

8 Extending the Driver Framework to a Distributed Program

As discussed in Section 2.1, a distributed program is defined as a program where management control is shared by multiple people from different organizations. Our field experience indicates that people employing traditional, tactical risk management approaches tend to have significant difficulty assessing and managing risk in distributed programs. The bottom-up nature of tactical risk management makes it difficult and time consuming to establish a comprehensive risk profile for distributed programs. In addition, many critical risks are not identified or are overlooked when people rely on tactical approaches. In contrast, systemic approaches are better suited for application in distributed management environments.

8.1 Network of Objectives

A broad network of objectives exists *within* all organizations. Success at the organizational level requires ensuring that all objectives within the network are aligned. Ensuring alignment among an organization's objectives helps establish confidence that

- core business objectives within the organization will be achieved
- the organization's overall objectives will also be accomplished

The network of objectives can also *extend across multiple* organizations. For example, when multiple companies collaborate on a joint venture, such as building and fielding a complex software-intensive system, they pool their resources toward achieving a common set of objectives. Each organization must balance its local objectives against the shared set of objectives defined by the overarching program. This concept is illustrated in Figure 17.

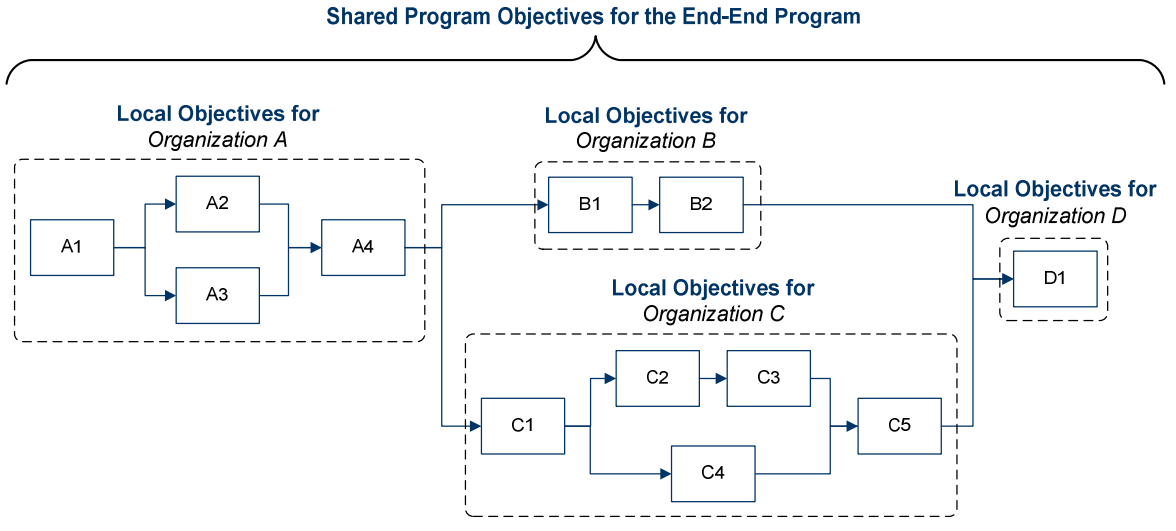

Figure 17 Network of Objectives

Each group in Figure 17 has a local set of objectives based on their program roles. In addition, there is an overall set of objectives for the end-to-end program. Once the network of objectives is determined, drivers across the distributed program can be assessed.

8.2 Applying the Driver Framework to a Network of Objectives

Figure 18 illustrates how a driver-based approach can be used to assess a distributed program. A driver framework is first established for the local objectives of each group within the network. In addition, a driver framework is also established for the end-to-end program objectives. In this way, risks to local objectives as well as risks to the end-to-end objectives are assessed, which provides a comprehensive risk profile for the distributed program.

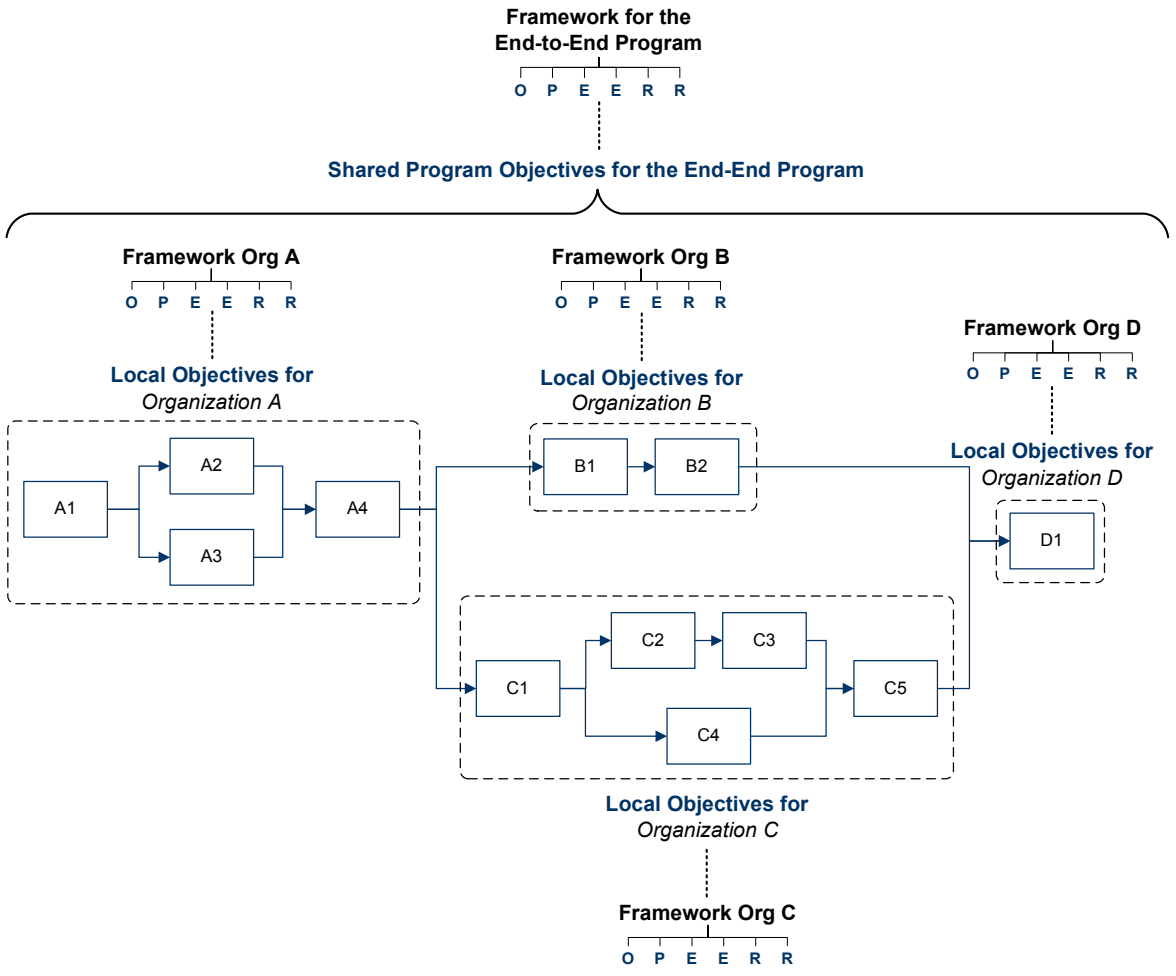

Figure 18 Applying the Driver Framework to a Distributed Program

The assessment approach for a distributed program requires analysis of the following three classes of risk:

1. risk to local objectives
2. inherited and imposed risk
3. emergent risk

First, the risks to each group's key objectives are assessed. This view identifies local risks that might affect the performance of each individual group. The second perspective examines how local risks can propagate throughout the distributed program. Each group in a distributed program imposes some degree of risk on downstream activities. The amount of risk imposed generally depends on two factors: (1) how much risk is *inherited* from upstream activities and (2) the amount of risk generated locally. Risk is also then *imposed* on the downstream activity. Risk thus flows in unison with work products as they move throughout a distributed program; it is amplified or dampened at any particular point in the workflow based on conditions at that location. By analyzing inherited and imposed risk, you can determine how risk cascades throughout a distributed program.

Finally, the driver framework for the end-to-end program enables analysis of risks caused by emergent properties and overarching program conditions. An *emergent property* is a characteristic of a system that is derived from the interaction of its parts and that is not observable or inherent in the parts when considered separately. Some risks arise from the emergent properties of a distributed program. These particular risks, called *emergent risks*, are particularly problematic because they are not easily observed from the vantage points of participating groups. As a result, emergent risks are most often neither identified nor effectively managed using tactical risk management approaches. Our research has demonstrated that systemic approaches, when applied across a network of objectives as describe above, enable identification of emergent risks.

9 Summary and Future Directions

Our research for the past three years has been focused on developing risk assessments designed for the unique requirements of distributed management environments. During this time, we piloted our assessments in two main areas: (1) distributed software acquisition and development programs and (2) distributed cyber-security incident management processes.[18] A key part of our research and development activities has been the development of an approach for assessing risk from a systemic point of view. The focal point of that work is the framework for categorizing program drivers.

Distributed environments typically comprise a network of highly complex components that are linked together. During our research, we came to the conclusion that traditional, tactical approaches for assessing and managing risks were unable to handle the complex risks inherent in these environments. The nonlinear, interrelated characteristics of distributed environments led us to explore the merits of systemic approaches for assessing risk in these complex settings. In contrast to the bottom-up analyses employed in tactical risk management, systemic approaches incorporate top-down, system-oriented analyses. Through our piloting of Mosaic assessments, we found systemic approaches to be better suited for assessing and managing risk in distributed environments.

The centerpiece of the Mosaic approach is the driver framework. As used in this context, a driver is defined as a factor that has a strong influence on the eventual outcome or result. Drivers are important because they define a small set of factors that a manager can use to determine whether or not a program is on track to achieve its key objectives. We developed the driver framework as a common structure for classifying the set of drivers that are used to assess a program's current state. The driver framework comprises six basic categories of drivers: (1) objectives, (2) preparation, (3) environment, (4) execution, (5) resilience, and (6) result. The main rule when compiling a set of drivers is to make sure you include at least one driver from each of the six driver categories in the set.

Assessing drivers from all categories helps to ensure an adequate breadth of data collection. Once driver values have been established, you can employ several types of follow-on analyses. In our research activities, we have used in the following back-end analysis approaches: (1) gap analysis, (2) risk analysis, (3) mission success analysis, (4) mission assurance analysis, and (5) integrated risk and opportunity analysis. In this report, we have focused primarily on risk analysis.

9.1 Distributed Management Environments

We have successfully applied systemic approaches for assessing risk to distributed environments, where management control of a program, process, or technology is shared by multiple people from different organizations. We evaluated a government organization's acquisition of an enterprise-wide business application using our assessment methods.[19] This pilot comprised four distinct organizations, all of which were in different geographic locations. At the conclusion of the assessment, we were able to pro-

[18] See *Preview of the Mission Assurance Analysis Protocol (MAAP): Assessing Risk and Opportunity in Complex Environments* [Alberts 2008] for a more detailed discussion of the MAAP, which is a method for assessing risk and opportunity in distributed programs.

[19] See the *Lessons Learned Applying the Mission Diagnostic* [Dorofee 2007] technical note.

vide managers with a comprehensive profile of the program's risks. The risk profile included a broad range of risks and featured several risks that were not previously identified.

We also used Mosaic assessment methods to evaluate a government organization's cyber security incident management process, which included three distinct points of management control and three geographic locations. At the conclusion of the assessment, senior managers from the government organization understood exactly how well events and incidents were managed. In this pilot, we analyzed risks for a variety of operational circumstances, providing a snapshot of the process' likely performance during expected and stressed conditions. Similar to our assessment of the government organization's acquisition of an enterprise-wide business application, our assessment of the incident management process identified several high-priority risks that were previously not known to decision makers.

9.2 Future Directions

This technical report provides a brief summary of key results from our research and development activities from the past three years. While this research has already provided many tangible results, we believe many additional research avenues should be explored. Foremost, we intend to continue to refine, pilot, and transition our work in systemic risk management. Candidate areas for future applications of our methods include

- software assurance
- supply chain management
- critical infrastructures

Finally, the complexity of programs, processes, and technologies continues to grow. With this increasing complexity comes new, even more subtle, forms of risk. Many of these increasingly complex risks will be too nuanced for our current assessment methods to detect. For this reason, we believe that modeling and simulation of risk in complex environments is a research area that would be important to investigate further. Despite the considerable progress we have made in the past three years, we view the work documented in this technical report as a starting point for extending the discipline of risk management rather than as a completed body of research.

Appendix A Starter Set of Drivers

This appendix provides a questionnaire for analyzing the starter set of drivers. To analyze the set of drivers, you must complete the following two steps.

1. Answer each question in the survey by checking the most appropriate box. Each question requires a yes/no answer. The following table defines the range of possible answers for each question:

Answer	Definition
Yes	The answer is almost certainly "yes." Almost no uncertainty exists. There is little or no probability that the answer could be "no." (~ > 95% probability of yes)
Likely Yes	The answer is most likely "yes." There is some chance that the answer could be "no." (~ 75% probability of yes)
Equally Likely	The answer is just as likely to be "yes" or "no." (~ 50% probability of yes)
Likely No	The answer is most likely "no." There is some chance that the answer could be "yes." (~ 25% probability of yes)
No	The answer is almost certainly "no." Almost no uncertainty exists. There is little or no probability that the answer could be "yes." (~ < 5% probability of yes)
Don't Know	More information is needed to evaluate the driver.

2. After you answer each question, document the rationale for your answer in the space provided. Include the following in your rationale when possible:
   - positive conditions that support an answer of *yes*
   - negative conditions that support an answer of *no*
   - potential events with positive consequences that support an answer of *yes*
   - potential events with negative consequences that support an answer of *no*

Driver Questionnaire

Question	Answer						Rationale
	No	Likely No	Equally Likely	Likely Yes	Yes	Don't Know	
1. Are program objectives (product, cost, schedule) realistic and achievable? Consider: alignment of technical, cost, and schedule objectives; inherent technical risk; technology maturity; resources available	❑	❑	❑	❑	❑	❑	
2. Is the plan for developing and deploying the system sufficient? Consider: acquisition or development strategy; program plan; resources; funding; schedule; roles and responsibilities	❑	❑	❑	❑	❑	❑	
3. Is the process being used to develop and deploy the system sufficient? Consider: process design; measurements and controls; process efficiency and effectiveness; acquisition and development life cycles; training	❑	❑	❑	❑	❑	❑	
4. Are tasks and activities performed effectively and efficiently? Consider: experience and expertise of management and staff; staffing levels; experience with the acquisition and development life cycles	❑	❑	❑	❑	❑	❑	

Driver Questionnaire (cont'd)

Question	Answer						Rationale
	No	Likely No	Equally Likely	Likely Yes	Yes	Don't Know	
5. Are activities within each team and across teams coordinated appropriately? Consider: communication; information sharing; dependencies; relationships; partners and collaborators	❑	❑	❑	❑	❑	❑	
6. Will work products from suppliers, partners, or collaborators meet the program's quality and timeliness requirements? Consider: applications; software; systems or sub-systems; hardware	❑	❑	❑	❑	❑	❑	
7. Is the program's information managed appropriately? Consider: usability; confidentiality; integrity; availability	❑	❑	❑	❑	❑	❑	
8. Does the program team have the tools and technologies it needs to develop the system and transition it to operations? Consider: software applications; infrastructure; systems; databases	❑	❑	❑	❑	❑	❑	

Driver Questionnaire (cont'd)

Question	Answer						Rationale
	No	Likely No	Equally Likely	Likely Yes	Yes	Don't Know	
9. Are facilities and equipment sufficient to support the program? Consider: building; physical work spaces; support equipment; supplies; other resources	❑	❑	❑	❑	❑	❑	
10. Are enterprise, organizational, and political conditions facilitating completion of program activities? Consider: stakeholder sponsorship; actions of upper management; effect of laws, regulations, and policies	❑	❑	❑	❑	❑	❑	
11. Does the program comply with all relevant policies, laws, and regulations? Consider: policies; laws; regulations; standards of care	❑	❑	❑	❑	❑	❑	
12. Does the program have sufficient capacity and capability to identify and manage potential events and changing circumstances? Consider: risk management plan, process, and tools; schedule slack; funding reserve; risk mitigation plans; program continuity and contingency plans; opportunity management plan, process, and tools	❑	❑	❑	❑	❑	❑	

Driver Questionnaire (cont'd)

Question	Answer						Rationale
	No	Likely No	Equally Likely	Likely Yes	Yes	Don't Know	
13. Are system requirements well understood? Consider: customer, user, and stakeholder requirements and needs; functional and non-functional requirements; operational requirements; system growth and expansion needs; technology maturity	❑	❑	❑	❑	❑	❑	
14. Are the design and architecture sufficient to meet system requirements and provide the desired operational capability? Consider: interfaces; dependencies; software and system architecture; operational requirements; technology maturity	❑	❑	❑	❑	❑	❑	
15. Will the system satisfactorily meet its requirements? Consider: functional; performance; operational; reliability; security; safety; usability; maintainability; technology maturity	❑	❑	❑	❑	❑	❑	
16. Will the system sufficiently integrate and interoperate with other systems when deployed? Consider: interfaces, applications, tools, hardware, data; technology maturity	❑	❑	❑	❑	❑	❑	

Driver Questionnaire (cont'd)

Question	Answer						Rationale
	No	Likely No	Equally Likely	Likely Yes	Yes	Don't Know	
17. Will the system effectively support operations? Consider: business and operational workflows; support of organizational and enterprise missions; operational risk mitigation; disaster recovery, contingency and business continuity plans; technology maturity	❑	❑	❑	❑	❑	❑	
18. Have barriers to customer/user adoption of the system been managed appropriately? Consider: user acceptance; stakeholder sponsorship; transition to operations; user support	❑	❑	❑	❑	❑	❑	
19. Will people be prepared to operate, use, and maintain the system? Consider: policies; procedures; training	❑	❑	❑	❑	❑	❑	
20. Will the system be appropriately certified and accredited for operational use? Consider: compliance with policies, laws, and regulations; acceptable mitigation of risk	❑	❑	❑	❑	❑	❑	

Appendix B Glossary

Term	Definition
category	A driver attribute; category classification or group (from a standard set of six) to which a driver can belong.
consequence	The loss that will occur when a threat is realized; one of the two components of risk
distributed management environment	A program, process, or technology situation where management control is shared by multiple people from different organizations. Also, *distributed environment*, *multi-enterprise environment.*
distributed program	A program where management control is shared by multiple people from different organizations; a type of distributed management environment.
driver	A factor that has a strong influence on the eventual outcome or result, that is, on whether or not key objectives will be achieved; comprises four attributes, *name*, *success state*, *failure state*, *category*.
driver analysis	A method used to determine how each driver is influencing key objectives.
driver attributes	The four main features of a driver.
driver framework	A common structure for classifying a set of drivers that influence a program's outcome; comprised of six categories Also, *OPEERR* (pronounced *oh-peer*) framework. See *driver framework categories.*
driver framework categories	The classifications of the driver framework—objectives, preparation, execution, environment, resilience, result. Also, *categories*, *framework categories.*
driver identification	A method used to translate key objectives into drivers. See *key objective*, *driver.*
driver profile	A snapshot, or summary, of all drivers relevant to a program.
driver value criteria	A set of criteria used to support driver analysis; provides a definition of applicable responses to a driver question, and translates each response into the probability that the driver is in its success state as well as the probability that it is in its failure state.
execution	A driver framework category; drivers in this category focus on assembling, organizing, and overseeing the assets required to bring that plan to life (how tasks and activities are managed and performed).
environment	A driver framework category; drivers in this category focus on how the broader environment affects program performance (how environmental conditions, inherited constraints affect work tasks).
failure state	A driver attribute; a situation where a driver exerts a negative influence on the outcome.

gap analysis	An analysis method; can be used in conjunction with driver analysis to examine the difference between current and desired states of each driver.
impact	A measure of the loss that will occur if a threat is realized.
integrated risk and opportunity analysis	An analysis method; can be used in conjunction with driver analysis to examine the risks and opportunities inherent in a situation and help strike an appropriate balance between the two.
issue	A loss that has occurred or is certain to occur.
key objective	A vital outcome intended to be achieved in the future; it provides a benchmark against which success will be judged.
leading indicator	A factor that provides insight into future performance.
mission opportunity	A circumstance that has the potential to provide a gain from the business or mission perspective
mission risk	A measure of potential loss in relation to key objectives; the risk produced by each driver. Also, *systemic risk*
mission assurance analysis	An analysis method; can be used in conjunction with driver analysis to establish a measure of mission assurance for each driver based on risk and uncertainty.
mission success analysis	An analysis method; can be used in conjunction with driver analysis to determine, using scenarios, the probability of successfully achieving each key objective.
name	A driver attribute; a concise label that describes the basic nature of the driver.
objective	A desired result or outcome that is being pursued.
objectives	A driver framework category; drivers in this category are focused on the purpose and scope of a program, for example: • a desired result or outcome that is being pursued • a set of objectives defines the mission being pursued by a program
opportunity	The likelihood of realizing a gain resulting from an allocation (or reallocation) of resources.
preparation	A driver framework category; drivers in this category focus on the processes and plans required to achieve objectives (the roadmap for achieving a picture of success).
probability	A measure of the likelihood that a threat will occur.
program	A collection of interrelated work tasks or activities that achieves a specific result.
resilience	A driver framework category; drivers in this category focus on the ability to effectively manage potential events and changing circumstances.
result	A driver framework category; drivers in this category focus the correctness and completeness of the product being developed or the service being provided.

risk	The likelihood of suffering loss; a measure of the likelihood that a threat will lead to a loss coupled with the magnitude of the loss.
risk analysis	An analysis method; can be used in conjunction with driver analysis to examine the potential loss produced by each driver in relation to the program's key objectives.
risk exposure	A measure of the magnitude of a risk based on current values of probability and impact.
risk profile	A snapshot, or summary, of all risks relevant to a program at a specific point in time.
success state	A driver attribute; a situation where a driver exerts a positive influence on the outcome.
systemic risk	A measure of potential loss in relation to key objectives; the risk produced by each driver.
systemic risk management	An approach for managing systemic risk; assumes a holistic view of risk to objectives by examining the aggregate effects of multiple conditions and potential events on a program's key objectives.
tactical opportunity	A circumstance that has the potential to improve program performance; it does not necessarily translate to a gain from the business or mission perspective
tactical risk	A measure of the likelihood that an individual potential event will lead to a loss coupled with the magnitude of the loss.
tactical risk management	An approach for managing tactical risks; views a threat as a potential event that might or might not occur and is focused on the direct consequences of that threat.
threat	A circumstance with the potential to produce loss.

References/Bibliography

URLs are valid as of the publication date of this document.

[Alberts 2008]
Alberts, Christopher, Dorofee, Audrey, & Marino, Lisa. *Preview of the Mission Assurance Analysis Protocol: Assessing Risk and Opportunity in Complex Environments* (CMU/SEI-2008-TN-011, ADA488415). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2008. http://www.sei.cmu.edu/publications/documents/08.reports/08tn011.html

[Alberts 2002]
Alberts, Christopher & Dorofee, Audrey. *Managing Information Security Risks: The OCTAVE*^{SM} *Approach*. Boston, MA: Addison-Wesley, 2002.

[Charette 1990]
Charette, Robert N. *Application Strategies for Risk Analysis*. New York, NY: McGraw-Hill Book Company, 1990.

[Dorofee 2007]
Dorofee, A.; Marino, L; Alberts, A. *Lessons Learned Applying the Mission Diagnostic*. (CMU/SEI-2007-TN-028). Pittsburgh, PA, Software Engineering Institute, Carnegie Mellon University, 2007. http://www.sei.cmu.edu/publications/documents/08.reports/08tn004.html

[Dorofee 1996]
Dorofee, A.; Walker, J.; Alberts, C.; Higuera, R.; Murphy, R.; & Williams, R. *Continuous Risk Management Guidebook*. Pittsburgh, PA, Software Engineering Institute, Carnegie Mellon University, 1996. http://www.sei.cmu.edu/publications/books/other-books/crm.guidebk.html

[Gallagher 1999]
Gallagher, Brian. *Software Acquisition Risk Management Key Process Area (KPA) – A Guidebook Version 1.02* (CMU/SEI-99-HB-001, ADA370385). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1999.
http://www.sei.cmu.edu/publications/documents/99.reports/99hb001/99hb001abstract.html

[Kloman 1990]
Kloman, H. F. "Risk Management Agonists." *Risk Analysis 10,* 2 (June 1990): 201-205.

[Williams 1999]
Williams, R.; Pandelios, G.; & Behrens, S. *Software Risk Evaluation (SRE) Method Description (Version 2.0)* (CMU/SEI-99-TR-029, ADA441900). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1999.
http://www.sei.cmu.edu/publications/documents/99.reports/99tr029/99tr029abstract.html

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	April 2009	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
A Framework for Categorizing Key Drivers of Risk	FA8721-05-C-0003

6. AUTHOR(S)

Christopher J. Alberts and Audrey J. Dorofee

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2009-TR-007

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	ESC-TR-2009-007

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

In today's business and operational environments, multiple organizations routinely work collaboratively in pursuit of a common mission, creating a degree of programmatic complexity that is difficult to manage effectively. Success in these distributed environments demands collaborative management that effectively coordinates task execution and risk management activities among all participating groups. Approaches for managing program risk have traditionally relied on tactical, bottom-up analysis, which does not readily scale to distributed environments. Systemic risk management is an alternative approach that is being developed by the Software Engineering Institute (SEI). A systemic approach for managing risk starts at the top—with the identification of a program's key objectives. Once the key objectives are known, the next step is to identify a set of critical factors, called drivers, that influence whether or not the key objectives will be achieved. The set of drivers also forms the basis for subsequent risk analysis. This technical report describes a driver-based approach for managing systemic risk in programs that acquire or develop software-intensive systems and systems of systems. It features a framework for categorizing drivers and also provides a starter set of drivers that can be tailored to the unique requirements of each program.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Risk, risk analysis, risk management, risk and opportunity management, systemic risk management, risk and opportunity analysis, mission success, mission risk	54

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. Z39-18
298-102