Scoping Software Projects

Robert N. Sulgrove

The key to risk management is to be as complete as possible in identifying project risks. This paper discusses the project-scoping process, which is being successfully used by software developers at AT&T Global Information Solutions. Project scoping is a method or process used for identifying and assessing risks to determine a project's feasibility. Lists of requirement categories and risk factors are provided as facilitating tools. Project scoping provides a basis for defining a less risky project and for redefining or discontinuing projects that are too risky. The project-scoping process also provides a basis for continuously monitoring risks during development to detect emerging problems at the earliest possible moment—while there is still time to take effective corrective action. Thus, project management can focus on development problems in addition to tracking schedule compliance. The bottom line is that by implementing project scoping, management has better control over a project.

Introduction

Software projects are complex. Uncontrollable project variabilities that cause schedule overruns have plagued software development for decades. Planning the details of a project is complicated and difficult if all possible project variabilities are to be accounted for. Risk is a large part of the problem. The following passage from an IEEE tutorial on software risk management highlights the problem:

> Like many fields in their early stages, the software field has had its share of project disasters. Most post-mortem examinations of these failed software projects indicated that many problems would have been avoided or greatly reduced if there had been an explicit early concern with identifying and resolving high-risk elements. Frequently, these projects were swept along by a tide of optimistic enthusiasm during their early phases, which caused managers to miss some clear signals of high-risk issues that proved to be the projects' downfall later.[1]

While the situation is improving, the demand for on-time completion and ever-shorter development cycles is requiring that the project planning and project management processes be dramatically improved.

All software development projects have some degree of risk associated with them. Some of the primary reasons that risks exist include:

- Inherent product complexity,
- Unproved product technologies,
- Limited developer expertise, and
- Project constraints.

Traditionally, risk identification and assessment were informal procedures in which developers attempted to think of all the difficulties that could arise. The developers then attempted to accommodate these difficulties. Successful project management, however, involves managing not only risks but also the project and its resources. A realized risk can cause a project to overrun its budget or schedule or even fail. Nevertheless, eliminating all risks might not be possible. It might not be desirable either, because doing so can compromise

the rationale for undertaking a project.

Risk management strategy can be divided into four steps:

- *Risk identification and assessment*, which identifies and assesses risks as soon as possible to determine a project's potential problem areas;
- *Risk mitigation*, which reduces the likelihood of occurrence and the impact of each risk as much as possible;
- *Risk-influenced project plan*, which incorporates actions that address each risk throughout a project; and
- *Risk management*, which continuously monitors and reassesses the risks throughout a project as more information about each risk becomes available, adjusting the project plan accordingly.

The key to this strategy is to identify risks as completely as possible. To this end, a new *project-scoping process* was developed for use by AT&T Global Information Solutions (AT&T-GIS) software developers. The word "scoping" is used to convey the breadth—or scope—of risks that can arise during a project.

Project scoping is a method or process used for identifying and assessing risks to determine a project's feasibility. Additionally, project scoping allows for mitigation of the risks by providing an extensive list of risk factors for consideration.

The process is used as a first step in planning a project, but it can also be used effectively at other times throughout a project. Project scoping is relatively easy and inexpensive to implement, and it can be adapted to work on other types of projects besides software development.

The project-scoping process was adopted in stages by software developers at AT&T-GIS. Developers first tried the assessment on past projects. They learned that the total number of high and moderate risks did indeed match their experience with problems encountered during a project. This realization led developers to try the process on current projects that were still in the early stages of development. Project-scoping users quickly discovered that several significant risks had been overlooked. They also found that several project requirements needed further refinement.

The AT&T-GIS software developers determined that the project-scoping process was a good value based on these initial trials. At this point, they tuned the process to fit their specific organizational and product needs and have adopted it for use on all new projects.

Panel 1. Abbreviations, Acronyms, and Terms

AT&T-GIS—the Global Information Solutions business unit of AT&T
IEEE—Institute of Electrical and Electronics Engineers
SEI—Software Engineering Institute

Overview of Project Scoping

The project-scoping process seeks to establish an *acceptable* project having an *acceptable* level of risk to deliver an *acceptable* product in an *acceptable* time frame. Project scoping sets product requirements, establishes both a top-level project interval and cost estimate, and identifies risks to determine the feasibility of undertaking a project. It is intended to provide the solid foundation needed for the preparation of a realistic project plan, which in turn improves the likelihood that a project will be completed successfully and delivered on schedule.

Project scoping does more than just provide a go/no-go project indicator. It also identifies the risks that impede projects. Even when a project must be undertaken regardless of its associated risks, project scoping will identify the risks and lead to a more effectively planned and managed project. Project scoping provides the basis for continuously monitoring risks during development to detect emerging problems at the earliest possible moment—while there is still time to take effective corrective action.

Project scoping begins once an initial set of requirements is established. It is an interactive, structured, and collaborative effort involving product development, marketing, distribution, support, and other organizations. The project-scoping process consists of the following steps, which are described in detail in the next three sections:

- Identifying and assessing product requirements risks,
- Formulating a top-level estimate, and
- Identifying and assessing project risks.

After these project-scoping steps are completed, the issues of the significance of the risks and their projected impact on the development effort are addressed. The resulting outputs are repeatedly evaluated together until

Panel 2. Product Requirements Categories

Functional Definition Requirements
- Transactions
- Data storage
- Distribution
- Telecommunications
  - –Rate
  - –Distance
  - –Volume
  - –Protocol
- Network
- Devices and media
- Compatibility
- Backup
- Restore
- Recovery
- Displays
  - –Size
  - –Resolution
  - –Formats
- Foreign languages
- Maintenance
- Security
- Transaction logs

Technology Requirements
- Platform(s)
- Hardware
- Communications
- Interfaces
- End-users
- Service users
- Support users
- Interdependencies with other products, devices, protocols, telecommunications

Performance Requirements
- Throughput rates
- Response rates
- Continuous operation
- User maintenance-cycle frequency
- Data transmission capacity
- File capacity
- Degraded-mode performance

Market Requirements
- Market window of opportunity
- Geographic and vocational markets
- Expected competition

Regulatory Requirements
- Certification
- Privacy and security laws
- Warranty
- Safety
- Environment
- Pending legislation
- Import/export regulations

Quality Requirements
- Definition of all failure events
- Consequence of each failure
- Definition of degraded-mode operation
- Reliability
- Availability
- Serviceability
- Diagnostics
- Method of transmitting failure data to developer for review
- Method of transmitting corrections to customer quickly
- Failure work-arounds
- Failure predictor alerts; for example, out of cash
- Customer reliability objectives; willingness to pay for more
- Usability
- Installability
- Defects
- Unusual environment operating conditions

the following questions are resolved:

- *Top-level project estimate*: Are the estimated delivery date and effort within acceptable limits?
- *Product requirements*: Do the product requirements continue to define a marketable product?
- *Product requirements risks*: Are the product requirements risks acceptably low?
- *Development risks*: Are the development risks acceptably low?

The project-scoping process can be either easy or difficult depending on how many iterations and compromises are required. The resulting outputs of the process are then used to prepare a detailed project plan.

Identifying and Assessing Risks

Project scoping begins with a set of product requirements or a statement of product opportunity. This input can be in the form of a full product requirements document, a list of enhancements and changes to an existing product, or a prototype. The requirements can include almost anything that provides insight into the product being proposed for development, such as the following components:

- Function,
- Technology,
- Interface,
- Performance,
- Schedule,
- Cost,
- Regulations,
- Quality,
- Support,
- Introduction date, and
- Support requirements.

Using a checklist is a good approach for identifying product requirements. A checklist helps to ensure that areas of requirements are not inadvertently overlooked. Panel 2 provides a detailed checklist for establishing a comprehensive set of product requirements. This checklist can be tailored to meet individual needs as determined from previous projects.

Product requirements are continually refined throughout the project-scoping process until acceptable levels of product risk are achieved. Of course, it is assumed that the requirements continue to define a desirable product. A product must not be compromised so much that it becomes undesirable or only marginally desirable.

To begin the product requirement risk assessment, each individual statement of requirement is categorized by its degree of importance accordingly:

- *Critical*, which indicates a requirement that is the essence of the product;
- *Important*, which indicates a requirement that greatly enhances the marketability of the product and indicates that it would be used by many customers;
- *Desired*, which indicates a requirement that enhances the product and improves its marketability but that also can be deferred to a later release; and
- *Nice to have*, which indicates a requirement that could be dropped, if necessary.

Each individual product requirement statement is assessed against the eight risk factors shown in Panel 3. A risk level is identified for each factor. The lowest possible risk level is zero, and higher risks are indicated by larger numbers. The eight risk-level numbers are added together to determine a total risk indicator value for each single product requirement. The requirements are then ranked according to their total risk number.

All high-risk requirements should be reviewed individually and very carefully because any one of them can seriously impede or even cause the cancellation of a project. It may be that a high-risk requirement can be reworked to reduce its risk level or that the requirement can be deferred to a later product release. The degree of importance of an individual requirement helps to ascertain whether or not it must be included in the initial product release. Undoubtedly, the project team will have to address all the critical requirements regardless of the risk level.

Additionally, high-risk product requirements are compared to past projects having a similar number of high-risk requirements. This comparison provides the organization with an indication of a project's feasibility. When more high-risk product requirements exist than an organization has ever before successfully handled, management should be very concerned about the organization's ability to complete the new project successfully and without significant difficulty.

Formulating a Top-Level Estimate

A calculation of the cost and time (duration or interval) necessary to complete a project is referred to as a *top-level estimate*. Costs include those for personnel and

other necessities, such as laboratory facilities and overhead, as well as purchased software, hardware, or services. The top-level estimate is an initial approximation used to indicate whether a project will meet its market window and how much it will cost.

The first step in preparing a top-level estimate is determining the size of the software product to be developed. All project models require a size estimate of some type. The estimated product size is the key to preparing a valid top-level estimate. Inaccurate size estimates can be the most significant source of error in the entire project-scoping process. Consequently, product size estimation should be given careful attention.

Estimating the size of a software product is typically complex regardless of the units of measure used—for example, lines of code or number of function points. The following common size estimation techniques are discussed elsewhere in full detail[2]:

- *Expert judgment*; the opinion of a very well qualified individual.
- *Analogy*; a comparison of a new product with similar existing products for which sizes are known.
- *Delphi*; a procedure by which estimates of multiple qualified people are refined until a group consensus emerges. Several variations of this technique can be used to accommodate different situations and levels of formality.

No single technique is better than any other. Its value should be determined by the "goodness" of the estimates obtained. What works well in one situation might not work so well in another.

A reasonable approach is to have experienced developers prepare initial size estimates based on past project data. An advantage of having such multiple and independent estimates is that the variability of the estimates provides an indication of the uncertainty and risk.

After estimating a product's size, a top-level project estimate can be developed. The top-level estimate can be completed by implementing one of the three size estimation techniques mentioned earlier or by using a *parametric model*[3]. The most reliable method is the parametric model, which is sometimes called a *macro cost estimation model*. The primary advantage of the parametric model is that it is derived from past project data.

Identifying and Assessing Project Risks

Many factors affect a software development project, and any one of them can lead to significant cost and schedule overruns. The variability of these factors represents a project's uncertainties and influences the cost and time estimates.

The list of factors involved in identifying and assessing risk, shown in Panels 4 through 8, were derived from an analysis of past projects at AT&T-GIS and from the IEEE Standard for Software Productivity Metrics[4].

This initial project risk assessment provides an early indication of possible significant risks. A more detailed assessment should be made later when the project plan is prepared.

A group of experienced software developers should review all the risk factors shown in Panels 4 through 8 to identify the risk level associated with each factor. During such a review, the statement and risk factors that most closely match the current situation should be chosen.

As before, the lowest possible risk level is zero, and higher risks are indicated by larger numbers. Factors categorized as Level 3 pose the highest risk. Level-3 risks are most likely to cause problems during a project. Thus, ways to reduce them to Level 2 or lower should be explored—ways such as redefining the product, modifying requirements, or improving the work environment by introducing better software, hardware, or processes.

Factors categorized as Level 2 pose a moderate risk. These risks sometimes cause significant problems, but not always. Some effort to mitigate Level-2 risks is recommended because several moderate risks can prove to be just as damaging as a single high-level risk.

Factors categorized as Level 1 pose a mild risk. These risks rarely cause problems but they should still be reduced, if possible.

Factors categorized as Level 0 pose virtually no risk and should not cause any problems.

When high and moderate risks exceed the maximum number that an organization can successfully handle, the proposed project could encounter major problems.

Conclusion

The project-scoping process was developed to provide a methodical approach for comprehensively identifying and assessing project risks. Its twofold objective is to ensure that all possible product and project

Panel 3. Product Requirements Risk Assessment

Successful product development is dependent on a clear and concise set of requirements. Each requirement is assessed for eight factors. The first six factors represent the characteristics of a good requirements statement identified in the *IEEE Guide to Software Requirements Specifications*[5]. Two new factors used to evaluate complexity and susceptibility to change are added. 0 = no risk; 1 = some risk; 2 = moderate risk; 3 = highest risk.

—Requirement ambiguity risk

☐ 0 Unambiguous. The requirement has only one interpretation.

☐ 1 Ambiguous. The requirement can be interpreted in more than one way.

—Requirement completeness risk

☐ 0 All expected and unexpected responses to a user input are defined.

☐ 1 Either not all possibilities are covered or they are yet to be determined.

—Requirement verifiability risk

☐ 0 Verifiable. It is possible to verify fulfillment of the requirement.

☐ 1 Unverifiable. It is not possible to verify fulfillment of the requirement.

—Requirement consistency risk

☐ 0 Consistent. The requirement is consistent with all other requirements.

☐ 1 Inconsistent. The requirement is in conflict with some other requirement.

—Requirement modifiability risk

☐ 0 Modifiable. The requirement can be changed easily, completely, and consistently.

☐ 1 Unmodifiable. The requirement cannot be changed easily, completely, or consistently.

—Requirement traceability risk

☐ 0 Traceable. The origin of the requirement is clear and the forward referencing is unique.

☐ 1 Untraceable. Either the forward or backward traceability is not clear.

—Product complexity risk

How well known is the product technology to support this requirement?

☐ 0 Very familiar and low-risk technology.

☐ 1 Familiar technology, but difficult.

☐ 2 Prototyped/demonstrated new technology.

☐ 3 New, never before demonstrated technology.

—Requirement changeability risk

How susceptible is the requirement to change?

☐ 0 Well known and stable.

☐ 1 Several alternatives are feasible.

☐ 2 New requirement; alternatives might not be known.

☐ 3 Marketplace is in a state of change.

risks are identified and that the effects of such risks on a project are recognized.

To date, experience with the project-scoping process at AT&T-GIS has been successful and well worth the effort. Software developers have noted the following significant benefits:

- Helps to ensure that the requirements are well stated and well understood.
- Provides an assurance that risks will be noticed.
- Helps to set an upper limit on the amount of risk that can be successfully managed.
- Requires minimal time to perform. The process typically takes from one day for small projects involving several individuals to a week or more for larger projects involving 25 or more people. The developers believe this is not *extra* time spent but time they would have to spend anyway addressing risks.
- Provides a basis for team building among developers and the other groups with which developers interact (for example, marketing, support, distribution, and so forth). The risks highlighted during such interaction and discussion provide a focus for team effort.

Software developers have used project scoping primarily to ensure that a proposed project is sufficiently well defined and understood before detailed planning takes place. This means that the detailed planning of some projects is deferred until their requirements are better defined. Additionally, the process has been used on projects already under way to determine if modifications are necessary.

Panel 4. Organizational Risk Factors

Coordination is necessary among the people and organizations involved on a project. All functional areas (for example, marketing, customer services, distribution, publications, and so forth) that are involved in the project are considered. 0 = no risk; 3 = highest risk.

—Interactions between organizations

Interactions between people and projects in other organizations add to project complexity due to additional coordination and communication requirements. These interactions include both internal organizations and external companies.

- ☐ 0 Single organization.
- ☐ 1 Multiple organizations, but no interactions required.
- ☐ 2 Multiple internal organizations requiring moderate interaction.
- ☐ 3 Multiple internal organizations and/or external companies involved, or extensive involvement required.

—Resource availability, scheduling, and coordination

All nonpersonnel resources must be available to those personnel needing them. Delayed availability, periods of unavailability for any reason, and overloaded resources can result in significant project delays. The timely availability of the needed resources must be reviewed against the needs for the same resource on some other project within the organization.

- ☐ 0 Adequate dedicated resources always available.
- ☐ 1 Dedicated resources always available, but quantity is limited.
- ☐ 2 Occasional contention for shared resources.
- ☐ 3 Frequent contention for shared resources.

—Personnel selection, scheduling, assignment, coordination, and continuity

The timely availability of the personnel having the expertise and experience needed must be reviewed against the needs for the same people or person on some other project within the organization. Continuity (low rate of turnover and reassignment) of the personnel anticipated for assignment to the project is critical.

- ☐ 0 Sufficient expert personnel are readily available.
- ☐ 1 Sufficient experienced personnel are usually available.
- ☐ 2 Experienced personnel must be shared.
- ☐ 3 No experienced personnel are available.

—Software process maturity

Project success is related to the maturity of the organization's development processes.[6] A Capability Maturity Model, such as the one developed by the Software Engineering Institute (SEI) and described elsewhere[6], should be used as the basis for assessment. Note that both SEI Capability Maturity Levels 4 and 5 are both rated at risk level 0.

- ☐ 0 SEI Capability Maturity Level 5—Optimized. A high degree of control over the development process has been achieved. Process data are used in an iterative manner to improve the process and achieve optimum performance.
- ☐ 0 SEI Capability Maturity Level 4—Managed. The process is understood, quantified, measured, and reasonably well controlled. Tools are used to control and manage the development process.
- ☐ 1 SEI Capability Maturity Level 3—Defined. The process is well defined in terms of software engineering standards and methods. Organizational and methodological improvements have already been made.
- ☐ 2 SEI Capability Maturity Level 2—Repeatable. Project cost and schedule management is now used successfully. Standard methods and practices are used for managing software development activities.
- ☐ 3 SEI Capability Maturity Level 1—Initial. Development processes and modern software engineering management practices and tools are inconsistently applied.

Panel 5. Development Environment Risk Factors

The development environment, testing environment, languages, and tools used by developers—and the stability of each—are important factors that influence the success of a project. 0 = no risk; 3 = highest risk.

—Software development environment

The common development virtual machine, individual workstations, tools, and the communications between them comprise a software development environment. The number, sophistication, cohesiveness, and integration of the tools make for a better environment.

- ☐ 0 Each developer has an intelligent workstation, which is connected electronically to a common host development system. Integrated software development packages are used exclusively.
- ☐ 1 Each developer has an intelligent workstation, which is connected electronically to a common host development system. Many tools are used, but they are not integrated.
- ☐ 2 No electronic connection exists between the developers or the tools used.
- ☐ 3 Shared workstations and an *ad hoc* collection of tools characterize this environment.

—Product testing environment

The complexity of the environment that must be used for testing a product can pose a significant risk.

- ☐ 0 Target testing environment is available and integrated with the development environment.
- ☐ 1 Good target processors are available, but they are not integrated with the development environment.
- ☐ 2 Poor target processor environment is available, but it is not integrated with the development environment.
- ☐ 3 Target processor environment is not available.

—Languages and tools environment

The tools used are assessed for their power, sophistication, quality, and degree of integration with other tools.

- ☐ 0 Very easy and instinctive to use tool, powerful, and integrated with other tools.
- ☐ 1 Many stand-alone commercial tools are used.
- ☐ 2 *Ad hoc*, unsupported, and undocumented tools are used.
- ☐ 3 Few or no tools are used.

—Stability risk

The stability of any aspect of the environment on which the developers work can pose a significant risk. How stable or volatile is the development environment?

- ☐ 0 No recent changes and none planned (very stable).
- ☐ 1 Only infrequent and planned changes.
- ☐ 2 More frequent and impromptu changes.
- ☐ 3 Unplanned changes occur constantly (very volatile).

—Developer experience risk

Lack of developer experience with the development environment, test environment, and tools can pose a significant risk. How experienced are the developers with the individual factor being assessed?

- ☐ 0 Extensive experience, proven track record.
- ☐ 1 Limited experience.
- ☐ 2 Familiar with, but never used.
- ☐ 3 No familiarity or experience.

Panel 6. Product Complexity and Technology Risk Factors

Product complexity risks can be quite formidable. The inherent complexity of any part of a product is a significant risk in itself. In addition to the factors listed, all moderately complex product components should be assessed individually. 0 = no risk; 3 = highest risk.

—Target virtual machine

Software is developed for what is called the target virtual machine, which includes all hardware, firmware, software, communication, and devices that comprise the total system. How complex is the target virtual machine?

- ☐ 0 Very simple and well known target system.
- ☐ 1 Relatively simple target system, but not well known.
- ☐ 2 Moderately complex target system.
- ☐ 3 Very intricate target system.

—Software product type

The type of product to be developed and its inherent complexity can pose a risk to the project. How complex is the product to be developed?

- ☐ 0 Very simple and well known product (for example, an incremental release).
- ☐ 1 Relatively simple product, but not well known.
- ☐ 2 Moderately complex product.
- ☐ 3 Very intricate algorithms and processing requirements.

—External software interfaces

Software with which the product must interact can dramatically affect the complexity of the product being developed and its testing. How complex are the external software interfaces?

- ☐ 0 No interfaces.
- ☐ 1 Relatively simple interfaces.
- ☐ 2 Moderately complex interfaces.
- ☐ 3 Many intricate interfaces.

Technological advances are constantly occurring. Adoption of the latest technologies results in a product having a good competitive advantage, although there are risks associated with new technologies. A list of all the product's technologies should be prepared and assessed individually. Possibly overlapping technologies are not a concern. One should be concerned, however, with ensuring that all technologies, especially new ones, are listed.

—Technology experience

How mature is the technology being assessed?

- ☐ 0 The organization has extensive development experience with this technology.
- ☐ 1 The organization has limited development experience with this technology.
- ☐ 2 Development experience with this technology exists outside the organization, but none inside.
- ☐ 3 No development experience with this technology exists anywhere.

Panel 7. Product Constraining Risk Factors

These factors address the complexity of a product affected by product related constraints. Constraints make it more difficult or even impossible to produce a product. Each factor becomes a risk as the parameter approaches its constraining limit. The product constraining risks identified in this panel represent what is known in the industry about the effects of common factors on a project. All the constraints are evaluated by the following criteria:

- ☐ 0 Evidence shows that this constraint will not be a problem.
- ☐ 1 Indications are that this constraint should not be a problem.
- ☐ 2 No indication as to whether or not this constraint will be a problem; some difficulty could occur.
- ☐ 3 Evidence or suspicion that this constraint will only be satisfied with extreme difficulty; many iterations might be required.

—Memory constraints

If a software product uses nearly all available memory, additional processing requirements must accommodate the situation. The choices are to increase the memory available, expend more effort to make it fit, or subdivide the design into smaller modules.

—Response time constraints

These constraints are associated with realtime requirements and unusually rapid response requirements. They can be closely intertwined with other constraints. Tight constraints frequently result in reduced response time and reduced throughput performance.

—Throughput execution time constraints

Fully used processor capacity results in system degradation due to larger queues and more queue handling.

—Data transmission constraints

Fully used data transmission capacity results in system degradation due to larger queues and more queue handling.

—Mass storage constraints

When more of the available mass storage is used, additional effort is needed to fit programs into the limited space.

—Security constraints

This factor covers tightly controlled security of outputs, as well as connections to uncontrolled systems.

—Data synchronization constraints

This condition occurs when multiple processors use the same data element, which has multiple states.

—Quality constraints

This factor considers unusual requirements for an extremely reliable system or unusually severe consequences for specified failures. These constraints can be applied to all quality items, such as availability, serviceability, usability, installability, maintainability, and so forth.

Panel 8. Project Constraining Risk Factors

These factors address the added complexity of a project constrained by schedule and budget. Problems arise only when limitations are severe. 0 = no risk; 3 = highest risk.

—Schedule

Excessive compression of a project schedule presents problems, including more personnel, increased communication overhead, and difficulties in personnel and resource management. The industry has shown that project schedules cannot be compressed beyond a compressibility limit.[7] The desire to achieve an early market date must be assessed in terms of its effects on development activity.

☐	0	Not known to be a problem.
☐	1	Should not be a significant problem.
☐	2	Likely to cause some additional difficulty.
☐	3	Possible only with extreme difficulty; many iterations might be required.

—Budget

Issues of budget and staff levels are related and tend to be inversely related to schedule. A simple equation does not exist for cost as a function of the effort and development time.

☐	0	Not known to be a problem.
☐	1	Should not be a significant problem.
☐	2	Likely to cause some additional difficulty.
☐	3	Possible only with extreme difficulty; many iterations might be required.

References

1. B. W. Boehm, *Software Risk Management*, IEEE Computer Society Press, Washington, D.C., 1989.
2. J. P. Martino, *Technological Forecasting for Decisionmaking*, American Elsevier Publishing Company, New York City, 1972.
3. S. D. Conte, H. E. Dunsmore, and V. Y. Shen, *Software Engineering Metrics and Models*, Benjamin/Cummings Publishing Company, Menlo Park, California, 1986.
4. IEEE Standard 1045-1992, *IEEE Standard for Software Productivity Metrics*, The Institute of Electrical and Electronics Engineers, Inc., New York City, 1993.
5. ANSI/IEEE Standard 830-1984, *IEEE Guide to Software Requirements Specifications*, The Institute of Electrical and Electronics Engineers, Inc., New York City, 1984.
6. W. S. Humphrey, *Managing the Software Process*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.
7. L. H. Putnam and W. Myers, *Measures for Excellence: Reliable Software On Time, Within Budget*, Yourdon Press, Englewood Cliffs, New Jersey, 1992.

(Manuscript approved January 1996)

Robert N. Sulgrove *is a software process and technology specialist in the Software Technology Center at AT&T Bell Laboratories in Columbus, Ohio. When he developed the project-scoping process, Mr. Sulgrove was a senior software process consultant with AT&T Global Information Solutions in Dayton, Ohio. He has a B.S. degree in mechanical engineering from the General Motors Institute in Flint, Michigan, and an M.S. in management science from the University of Dayton. Mr. Sulgrove joined AT&T in 1970.*