GALORATH

Deciphering Software Estimation

A Comprehensive 10-Step Process

www.galorath.com

Table of Contents

Intro

A disciplined estimation process builds confidence in every project decision. Consistency helps organizations manage uncertainty, compare alternatives, and maintain accountability across technical and financial teams.

This 10-step process provides a structured approach for analysts and project managers to follow from initial scoping through project tracking. Each step promotes transparency, traceability, and alignment between assumptions and outcomes.

Use this framework as a working reference. Adapt the details to match your organization's internal review process, terminology, and reporting standards. The goal is not just to complete an estimate but to create one that can be explained, reviewed, and improved over time.

The Art and Science of Software Estimation

Successful estimation is both analytical and interpretive. The science relies on structured methods, validated models, and data that support repeatable outcomes. The art depends on professional judgment. It requires the ability to recognize patterns, evaluate uncertainty, and apply context to the numbers.

Accurate estimates are grounded in three fundamentals:

- **Defined scope** – clarity about what is being built, modified, or reused.
- **Validated data** – inputs that reflect current technical and organizational realities.
- **Transparent reasoning** – documentation that explains how inputs lead to results.

No estimate can eliminate uncertainty entirely, but a disciplined process makes every estimate explainable. The purpose of estimation is not to predict a single outcome but to understand possible ranges and the factors that shape them. When applied consistently, software estimation becomes both a technical method and a management tool that improves communication, strengthens accountability, and builds confidence in delivery

The Power of Software Metrics

Software metrics provide the foundation for credible estimation. They translate technical activity into measurable factors that support planning, budgeting, and performance assessment. Metrics help teams compare projects objectively, identify risks early, and communicate progress in terms that decision-makers understand.

The true value of metrics lies in their consistency and context. When collected and applied systematically, they reveal patterns that guide better forecasting and process improvement. Metrics drawn from past projects establish a knowledge base that strengthens future estimates and reduces reliance on intuition.

Reliable metrics also promote accountability. They allow analysts and managers to verify assumptions, explain cost and schedule variations, and identify where process changes will yield the greatest benefit. By connecting quantitative data to qualitative judgment, metrics bridge the gap between technical detail and management insight.

Accurate, well-documented metrics transform estimation from a one-time exercise into an ongoing feedback process. Each completed project contributes new data that refines models, improves accuracy, and increases organizational maturity. Over time, the consistent use of software metrics builds confidence, supports transparency, and drives better outcomes across the enterprise.

The Project Estimation Process

A structured estimation process improves consistency, strengthens accountability, and supports better decision-making. Each step builds on the last, forming a repeatable framework that guides analysts and project managers from initial scoping through tracking and review.

The following ten steps outline a proven method for developing clear, traceable, and defensible software estimates.

1 **Establish Estimate Scope and Purpose:** Estimation begins with clarity about what is being analyzed and why. Define the boundaries, goals, and points of contact before collecting or interpreting data. Specify whether the estimate is an EAC, ETC, or another type, and note any integration with related programs or external systems. State the primary objective of the estimate, such as cost forecasting, schedule planning, or resource allocation. Identify the individuals responsible for supplying technical data and contractual or requirements information. A clearly defined scope prevents confusion later and keeps every participant working from the same baseline.

2 **Establish Technical Baseline, Ground Rules, and Assumptions:** A solid technical baseline forms the foundation for reliable estimation. Describe each CSCI or comparable component and explain how it interacts with others in the system. Record all ground rules and mandatory constraints that will shape development, including schedule limits, mandated tools, or environmental factors. Capture every assumption that influences the estimate, especially where data is incomplete or subject to change. Ensure that the baseline aligns with the Work Breakdown Structure or reporting framework used across the project so that traceability is maintained.

3 **Collect Data:** Accurate estimates depend on credible input data. Gather, verify, and document all relevant parameters before modeling begins. Customize data worksheets or templates to speed collection and improve consistency. Complete the data set for each software component, reviewing values for realism and internal consistency. Confirm uncertain entries with developers, contractors, or subject-matter experts. Taking time to collect accurate data early prevents rework and supports transparency throughout the estimation process.

4 **Determine Software Size:** Size is the primary driver of cost, effort, and duration in software development. Define size precisely for new, reused, and modified components. Estimate new code in terms of Source Lines of Code, Function Points, or another recognized measure, and record least, likely, and most values. Identify all reused or commercial components and note the level of modification or integration required. Include anticipated growth factors so the estimate reflects the expected increase in size during development. Clearly defining size early ensures that all subsequent calculations are grounded in consistent, defendable data.

5 **Prepare Baseline Estimate:** The baseline estimate serves as the foundation for all future comparisons and reviews. Enter verified data into the estimation model and document the reasoning behind each input. Check that the model's outputs make sense given the project's objectives, team capability, and technical scope. Conduct joint reviews with engineers and analysts to confirm that all assumptions and results align with expectations. Record any differences between this version and earlier estimates so that future reviews can trace changes and rationale. A well-prepared baseline creates a stable reference point for progress measurement and audit readiness.

6 **Quantify Risks and Risk Analysis:** Uncertainty is inevitable in software projects, but quantifying it improves control and decision quality. Identify risks associated with each major element and determine which should be reflected within the estimate. Document why each risk is included or excluded and describe its potential impact on cost and schedule. Consider whether risks will be presented collectively or individually to illustrate different levels of exposure. An estimate that incorporates risk provides decision-makers with a realistic range of possible outcomes rather than a single point value.

7 **Estimate Validation and Review:** Validation ensures that the estimate is credible, traceable, and aligned with the organization's objectives. Review the major cost and schedule drivers to confirm they reflect current conditions and priorities. Check that productivity assumptions are consistent with team performance and comparable industry data. Verify staffing levels and confirm that labor rates, cost elements, and inflation factors are accurate. Evaluate how risks and growth factors are represented in the model. A thorough validation process not only increases confidence but also builds understanding among reviewers who rely on the results.

8 **Generate a Project Plan:** Once the estimate is validated, translate it into a project plan that connects financial, technical, and schedule elements. Use the estimate to define milestones, staffing profiles, and key deliverables. Verify that the assumptions made during estimation are reflected in the plan's structure and timelines. This step links analytical modeling to operational execution, ensuring that commitments are realistic and resource needs are visible from the start. A project plan grounded in an approved estimate supports accountability and coordination across teams.

9 **Document Estimate and Lessons Learned:** Each time you complete an estimate and again at the end of software development, document the information that defines the estimate. This includes rationale for key parameters, version history, and any unique conditions that shaped the work. Maintaining clear documentation is not just about record-keeping; it is about learning from experience. Well-documented estimates create a reference library that improves accuracy over time and provides new analysts with context and guidance. This practice also preserves institutional knowledge and ensures that lessons learned are applied to future projects.

10 **Update Estimate:** Track and Update During Development
Estimation continues to add value throughout the life of the project. As actual performance data becomes available, compare it against the baseline to monitor trends in cost, effort, and schedule. Update the estimate with new information to maintain visibility into current conditions. Reporting these updates allows leaders to adjust plans, manage expectations, and make informed decisions as the project evolves. Continuous tracking turns estimation into a living management tool that strengthens both accountability and forecasting accuracy.

Step 1: Establish Estimate Scope and Purpose

Defining the Scope

A reliable estimate begins with a clear definition of what is included. Identify which parts of the system are in scope and where the boundaries lie. Clarify whether the estimate covers the whole project or specific components, and note dependencies on other systems or contracts. A well-defined scope keeps contributors aligned and reduces the risk of missed work or double counting.

A well-defined scope keeps contributors aligned and reduces the risk of missed work or double counting.

Identifying the Purpose

The purpose determines how much detail is required and how much uncertainty is acceptable. State whether the estimate supports budgeting, planning, trade studies, proposal development, or another management decision.

Making the purpose explicit helps reviewers interpret results correctly and confirms that the approach fits the decision it must inform.

Documenting the Scope and Purpose

Record the scope and purpose in the estimate's documentation and share them with stakeholders. Summarize what is being estimated, why the estimate is needed, and who will use the results. Clear documentation improves traceability, supports validation, and enables future reuse with the proper context.

Clear documentation improves traceability, supports validation, and enables future reuse with the proper context.

Step 2: Establish Technical Baseline, Ground Rules, and Assumptions

Setting the Technical Baseline

The technical baseline defines the foundation of what is being estimated. Describe the intended functions, interfaces, performance expectations, and relevant constraints.

Capture the information necessary for consistent cost, schedule, and risk reasoning so that contributors work from a common technical picture.

Establishing Ground Rules

Ground rules are the agreed conditions that shape the estimate. Record schedules, standards, policies, and tool or process choices that affect how the work will be performed.

Align these with stakeholders at the start so expectations are consistent and the estimate remains comparable over time.

Making Assumptions

Assumptions fill gaps where details are not yet known. State each assumption clearly, explain why it is reasonable, and plan when it will be revisited. As new information arrives, update assumptions and reflect changes in the estimate.

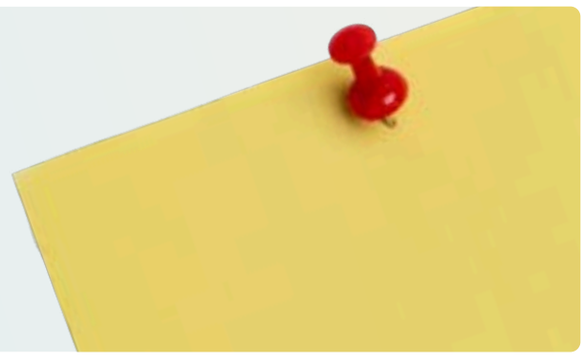
This practice preserves transparency and keeps results defensible.

Step 3: Collect Data

Identifying Data Sources

Accurate estimation begins with identifying the sources that provide the information required for modeling. Determine where cost, schedule, and performance data will come from, such as historical databases, engineering records, or subject-matter experts. Select sources that are complete, relevant, and consistent with the project's technical scope. Confirm the availability and reliability of each source before using it.

Select sources that are complete, relevant, and consistent with the project's technical scope. Confirm the availability and reliability of each source before using it.



Collecting and Organizing Data

Once data sources are identified, gather all necessary inputs in a consistent format. Organize the information so that similar items can be compared easily. Validate units, definitions, and timeframes to avoid misinterpretation. Review the data with team members to confirm that each entry accurately represents the activity or component being estimated. Well-organized data supports traceability and simplifies later analysis.

Review the data with team members to confirm that each entry accurately represents the activity or component being estimated. Well-organized data supports traceability and simplifies later analysis.

Documenting the Data

Record where each data element originated and when it was collected. Note any calculations, conversions, or adjustments made before entry into the model. Keep these records with the estimate so future reviewers can understand how inputs were derived and confirm their validity.

Maintaining complete documentation strengthens auditability and allows future estimates to build on proven

Step 4: Software Sizing

Sizing determines how large the software will be and sets expectations for cost, effort, and duration. Treat this step as the foundation for later modeling. Clarity and consistency here improve every downstream decision.

Understanding Software Sizing

Software size represents the functionality that must be delivered. Use an accepted measure such as Source Lines of Code or Function Points, and apply it consistently across components. Choose the method that best fits the level of detail available and the project's development approach.

The goal is a measure that reviewers can understand, compare, and reuse.

Prioritizing Sizing Efforts

Because size drives cost and schedule, invest adequate time in getting it right. When time is limited, focus on sizing before refining secondary details. Document the method and assumptions used so that future updates can build on a clear record rather than starting over.

Considering All Aspects of Size

Account for more than new development. Include reused, adapted, or modified software and note the work required to integrate and test it. Identify commercial or prototype components that contribute to delivered capability. A complete sizing view prevents gaps and double counting.

A complete sizing view prevents gaps and double counting.

Predicting Size

There are several methods to estimate size:

> **Manual sizing:** Manually inspecting the software to determine its size. For example, performing a function point count, or running a SLOC (Software Lines of Code) counter on the software.

> **Expert opinion:** This is an estimate based on recollection of prior systems and assumptions regarding what will happen with this system, and the experts' past experience.

> **Analogy:** A method by which you compare a proposed component to a known component it is thought to resemble at the most fundamental level of detail possible. Most matches will be approximate, so for each closest match, make additional size, as necessary. A relative sizing approach can provide viable size ranges based on comparisons to known projects.

> **Formalized methodology:** Use of automated tools and/or pre-defined algorithms such as counting the number of subsystems or classes and converting them to function points.

> **Statistical sizing:** Provides a range of potential sizes that is characterized by least, likely, and most.

Each method has its strengths and weaknesses, and the best approach often involves using a combination of methods.

Steps to Estimating Software Size

The process of estimating software size involves several steps:

1. **Establish a baseline definition of the size metric** you will use and identify a normalization process to use if size information is provided in a format different from the definition chosen.
2. Define sizing objectives—are you trying to describe the size of individual computer programs, plan major milestones in the estimation process, or adjust for project replanning? Varying granularities of sizing detail will be appropriate.
3. Plan data and resource requirements of the sizing activity.
4. Identify and evaluate software requirements, as unambiguously as possible.
5. Use several independent techniques and sources and put findings in a table.
6. Track the accuracy of the estimate versus actuals as the project progresses and re-estimate the product size periodically with actual data.

Documenting the Sizing Process

The sizing process and its results should be documented. This includes the methods used to size the software, the data used, the assumptions made, and the results obtained. This documentation provides a record of the sizing process and can be useful for validating the estimate and for learning lessons for future estimates.

By accurately sizing the software, you can create a solid foundation for your software estimate.

Step 5: Prepare Baseline Estimate

The fifth step in the software estimation process is to prepare the baseline estimate.

This step involves taking all the information gathered so far and using it to create an initial estimate of the size, cost, and duration of the software project.

Assigning the Right People

Preparing the baseline estimate should be entrusted to individuals trained, experienced, and skilled in software estimation. These individuals should have a deep understanding of the software development process, the technology being used, and the methods and tools of software estimation.

The team preparing the estimate should also be diverse, representing different perspectives and areas of expertise. This can help to ensure that all aspects of the project are considered and that the estimate is robust and comprehensive.

Using the Right Tools and Techniques

The baseline estimate should be prepared using appropriate tools and techniques. This might include parametric estimation software, spreadsheets, statistical analysis tools, or other tools that can help to analyze the data and calculate the estimate.

The techniques used might include parametric estimation, where the estimate is calculated based on statistical relationships between the project parameters; analogy-based estimation, where the estimate is based on comparisons with similar past projects; or expert judgment, where the estimate is based on the opinions of experienced individuals.

Documenting the Baseline Estimate

Once the baseline estimate has been prepared, it should be documented. This documentation should include a detailed breakdown of the estimate, showing how it was calculated and what assumptions were made. It should also include a description of the uncertainty and risk associated with the estimate, as well as any potential opportunities for cost savings or schedule acceleration.

By preparing a thorough and well-documented baseline estimate, you can provide a solid foundation for project planning and decision-making.

Step 6: Quantify Risks and Risk Analysis

The sixth step in the software estimation process is to quantify risks and perform risk analysis.

This step involves identifying potential threats to the project, assessing their likelihood and potential impact, and planning how to mitigate, eliminate or transfer the risks.

Understanding Risk in Software Projects

In the context of software projects, risk is characterized by a potential loss of time, quality, control, understanding, and so on. The loss associated with a risk is called the risk impact. We must also have some idea of the probability that the event will occur. The likelihood of the risk, measured from 0 (impossible) to 1 (certainty) is called the risk probability. When the risk probability is 1, then the risk is called a problem since it is certain to happen.

Risk Management Techniques

Effective managers of software projects often use risk management techniques to anticipate potential problems and devise mitigation strategies. Risk management enables you to identify and address potential threats to a project, whether they result from internal issues or conditions or from external factors that you may not be able to control.

Risk Control

For each risk, we must determine what we can do to minimize or avoid the impact of the event. Risk control involves a set of actions taken to reduce or eliminate a risk. By using risk management techniques to anticipate potential risks, the manager is protected against liability because if the problem does occur, it can be demonstrated that the cause was beyond what any prudent manager could have foreseen.

Risk Analysis in Estimation Process

Although cost, schedule, and product performance risks are interrelated, they can also be analyzed independently. In practice, risks must be identified as specific instances to be manageable. Statistical risk/uncertainty analysis should be a part of your schedule and effort estimation process.

By quantifying risks and performing risk analysis, you can anticipate potential problems and plan how to mitigate them.

Step 7: Estimate Validation and Review

The seventh step in the software estimation process is estimate validation and review.

This step involves systematically confirming the integrity of the estimate, ensuring that the data is sound, the methods are effective, the results are accurate, and the focus is properly directed.

Understanding Estimate Validation

Estimate validation is a crucial part of the estimation process. It involves checking the estimate to ensure that it is reasonable, accurate, and based on sound data and methods. This validation should ideally be performed by someone who was not involved in generating the estimate itself, who can view it objectively.

Reviewing the Estimate

When reviewing an estimate, it's important to assess the assumptions made during the estimation process. This includes checking that the adopted ground rules are consistently applied throughout the estimate. It's also important to consider whether any costs or risks have been underestimated or overlooked, and whether any productivity estimates have been overstated. Using a checklist is a great aid in reviewing the estimate. The checklist should include all aspects of the estimate that need to be reviewed, the adequate review criteria and an area to document pass/fail results and observations.

Identifying and Addressing Problems

A rigorous validation process can help to identify any faulty assumptions, unreliable data, or estimator bias. By identifying these problems, you can gain a clearer understanding of the risks inherent in your projections. You can then take steps to contain these risks, resulting in a more realistic picture of what your project will require to succeed.

The Importance of Validation

Despite the costs of performing a validation, it should be scheduled into every estimation project before the estimate is used to establish budgets or constraints on your project process or product engineering. Failing to do so may result in much greater downstream costs, or even a failed project.

By validating and reviewing your estimate, you can ensure that it is not only accurate but also credible and defensible. This step helps to ensure that your estimate is based on sound data and methods, and that it accurately reflects the scope and complexity of the software project.

Step 8: Generate a Project Plan

The eighth step in the software estimation process is to generate a project plan.

This step involves taking the estimate and allocating the cost and schedule to a function and task-oriented work breakdown structure. This usually takes the form of a Gantt Chart or similar task breakdown structure.

The Role of Project Planning in Software Development

Project planning is a critical component of software development. It provides a roadmap for the project, outlining the tasks that need to be completed, the resources required, and the timeline for completion. A well-structured project plan can help to ensure that the project stays on track and meets its objectives.

Creating a Function and Task-Oriented Work Breakdown Structure

A function and task-oriented work breakdown structure (WBS) is a key tool in project planning. The WBS breaks down the project into smaller, manageable tasks, making it easier to estimate the time and resources required for each task. This helps ensure that the project is properly resourced and that tasks are completed promptly.

Allocating Cost and Schedule to the WBS

Once the WBS has been created, the next step is to allocate the estimated cost and schedule to the tasks in the WBS. This involves determining how much time and resources each task will require, and then assigning these costs to the tasks in the WBS. This helps to ensure that the project stays within budget and on schedule. It is important that the estimated resources (hours, duration, labor roles, etc.) are accurately transferred to the project plan. In simple words, the estimate and the WBS project plan should be in synchrony.

The Importance of Project Planning

Project planning is crucial for the success of any software development project. Without a clear plan, projects can easily go off track, leading to delays, cost overruns, and other issues. By generating a project plan as part of the software estimation process, you can help to ensure that your project is well-organized, properly resourced, and on track to meet its objectives.

Addressing Challenges in Project Planning

Project planning can be challenging, particularly for complex software development projects. Some managers may lack the necessary experience to evaluate the long-term effects of their decisions, while others may make decisions based on what they think higher management wants to hear. A good software manager will understand what a project can realistically achieve, even if it is not what higher management wants. Their job is to explain the reality in a language their managers can understand.

Utilizing Software Management Models

Software management models, such as SEER-SEM from Galorath, can assist in guiding appropriate changes in the environment related to people, processes, and products. These models can help address issues by providing a structured approach to project planning, helping to ensure that projects are well-managed and successful.

Step 9: Document Estimate and Lessons Learned

The ninth step in the software estimation process is to document the estimate and lessons learned.

This step involves recording the pertinent information that constitutes the estimate, as well as the insights gained during the estimation process.

The Importance of Documentation

Documentation is a critical part of the software estimation process. By documenting the estimate, you create a record of the assumptions, methods, and data used to generate the estimate. This can be useful for future reference, for auditing purposes, and for improving the estimation process over time.

Recording the Estimate

When documenting the estimate, it's important to include all the relevant information. This includes the size and scope of the software project, the methods and tools used to generate the estimate, the assumptions made during the estimation process, and the estimated cost and schedule. It's also important to document any uncertainties or risks associated with the estimate.

Learning from the Process

In addition to documenting the estimate itself, it's also important to record any lessons learned during the estimation process. This could include insights into the effectiveness of different estimation methods, challenges encountered during the process, and areas for improvement. By recording these lessons, you can continually refine and improve your estimation process.

Reviewing and Updating the Documentation

The documentation should be reviewed and updated at the end of the software development process. This allows you to compare the initial estimate with the actual outcomes, identify any discrepancies, and understand the reasons for these discrepancies. This can provide valuable insights for future estimation projects.

Historical Data Collection

An organization that wants to establish a robust estimation process needs to refine that process continuously. Historical data is a fundamental part of this continuous improvement process: The outputs of the estimation process need to be compared against the historical data to determine whether the process is delivering results that are in line with actual performance. By measuring and comparing the estimation accuracy, an organization can refine and calibrate its estimation processes and tools.

The Role of Documentation in Continuous Improvement

By documenting the estimate and lessons learned, you not only create a valuable resource for future reference, but also contribute to a culture of continuous improvement. This can help to improve the accuracy and reliability of your software estimates over time, leading to better project planning and management.

Step 10: Update Estimate

The last step in the software estimation process is to update the estimate.

This step involves revising the estimate based on new information, changes in the project, and actual outcomes.

The Need for Updating

An estimate is not a static document. It should be viewed as a living entity that evolves with the project. As the project progresses, new information will become available, assumptions may change, and actual outcomes may differ from the initial estimates. All these factors should be reflected in the estimate.

When to Update the Estimate

The estimate should be updated regularly throughout the project. This could be at set intervals, such as every week or month, or it could be in response to specific events, such as a change in the project scope or a significant deviation from the planned schedule or budget.

How to Update the Estimate

Updating the estimate involves revising the data, methods, and assumptions used to generate the estimate. This could involve changing the estimated size of the software, adjusting the estimated effort or schedule, or modifying the risk assessment. It's also important to document the reasons for these changes, to provide a record of the project's evolution. An update to the estimate may trigger a re-execution of some (or all) of the previous 9 steps described in this document. For example, if a significant change in scope is introduced, it may be necessary to go back and revisit the estimate scope, update the technical baseline, estimate the size of the change, revisit the baseline estimate, etc.

Comparing Estimates with Actual Outcomes

One of the key benefits of updating the estimate is that it allows you to compare the initial estimate with the actual outcomes. This can provide valuable insights into the accuracy of your estimation methods and the reliability of your data. It can also help to identify areas where the project is deviating from the plan, allowing you to take corrective action if necessary. Project management techniques such as Earned Value Analysis can be used to compare actuals against the established baseline and forecast variations, identify issues, and apply timely corrective actions.

Learning from the Update Process

The process of updating the estimate can also provide valuable lessons for future projects. By analyzing the differences between the initial estimate and the actual outcomes, you can identify areas where your estimation process could be improved. This could involve refining your estimation methods, calibration estimation tools, improving your data collection processes, or enhancing your risk management practices.

By regularly updating your estimate, you can ensure that it remains accurate and relevant throughout the project. This can help to improve project management, enhance decision-making, and increase the likelihood of project success.

Conclusion

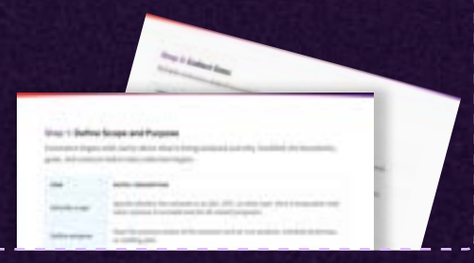
The process of software estimation is a critical component of successful software development. It provides a roadmap for project planning, resource allocation, risk management, and decision-making. However, it's not a one-time activity. It's a dynamic process that evolves with the project, requiring regular updates and revisions to reflect new information, changes in the project, and actual outcomes.

In this guide, we've walked you through the ten-step process of software estimation, from establishing the estimate scope and purpose, through data collection, sizing, risk analysis, and estimate validation, to updating the estimate. Each step is critical, and together they form a comprehensive approach to software estimation.

Software development is filled with uncertainty, but a disciplined estimation process turns that uncertainty into insight. With clear steps and the right tools, you can guide decisions with confidence, control cost and schedule risk, and deliver reliable outcomes.

As you plan your next software project, remember that **estimation is more than a calculation**. It is a management practice that links technical performance to business goals.

Use Galorath's 10-step checklist for a repeatable framework to produce transparent and defensible estimates.



Visit **Galorath.com** to learn how we can help you improve your estimates, manage projects more effectively, and increase the likelihood of success.