The Quadrus Estimation Methodology (QEM) applies a combination of statistical methods and industry experience to the software estimation process. We explore why traditional estimation techniques often prove unsuccessful and explain the benefits of better planned and managed projects.

# Quadrus Estimation Methodology

The Quadrus Estimation Methodology (QEM) applies a combination of statistical methods and industry experience to the software estimation process. QEM delivers estimates for project effort (total amount of work to be done), duration (total calendar time) and resources (number of FTEs) together with confidence intervals. Our estimation solution also takes into account individual task or user story estimates, non-development project overheads (such as meetings, planning and reporting), and new system requirements that may be identified as the project unfolds.

There are many reasons why traditional approaches to project estimation often don't produce accurate results. Even obtaining accurate estimates for individual tasks or user stories doesn't necessarily mean that your overall project estimate is accurate.

## The Challenges With Project Estimation

Estimates for software projects are typically based on single-point estimates of individual tasks. A developer is asked how long "feature x" will take to develop and they will provide an answer in a form such as n days. What is lacking from this is any indication of the range of likely values: for instance, is it n days with absolute certainty or n ± 10 days?

More experienced developers will provide a range of estimates to indicate the potential best-case versus worst-case scenarios in the form of n to m days. This still lacks valuable information and any indication as to where on the scale the most likely case will be. It also introduces the problem of how to combine multiple estimates. Is the lower range for the project estimate the sum of all the lower (best-case) values for each task estimate? Is the higher range for the project estimate the sum of all the higher (worst-case) values for each task estimate?

"Humans have a natural tendency to either pick the value that they think is likely to occur or the midpoint value (50% of cases are lower and 50% are higher). However, software project estimation techniques falsely assume that each individual estimate provided is the average of the range (i.e., arithmetic mean)."

If so, the resulting project estimate range will vary from wildly optimistic (every task comes out at the lowest estimate) to extremely pessimistic (where every task takes the maximum estimated).

Sometimes estimates are formed by resorting to averaging the individual task estimate ranges in order to add them: n to m days becomes n + (m-n) / 2 days (e.g., an estimate of 3 – 7 days becomes 5 days). But this doesn't really help – it simply turns the range estimate back into a single point estimate – an inaccurate point estimate, as it turns out!

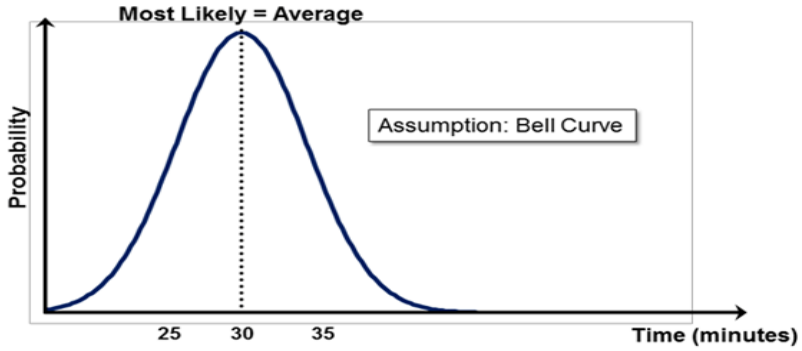
The root of the problem is that humans are conditioned to think a certain way and there are unstated, almost unconscious assumptions made when giving estimates. As long as we are aware of these we can combine the individual task estimates in a meaningful way to produce accurate project estimates.

But how big a problem is this? Won't these slight differences in individual estimates all even themselves out when they are combined to produce an overall project estimate? Unfortunately, no; combining these individual task estimates incorrectly actually amplifies the problem and can result in a huge discrepancy for the overall project.

## How The Statistical Methods Can Be Inaccurate

This example demonstrates how mishandling estimates can lead to project cost and schedule overruns. Suppose we need to estimate a task such as the commute from work to home. If asked how long our commute is we may say "30 minutes" or, if giving a range, we may say about "25 to 35 minutes".

When giving estimates like this, humans have a natural tendency to either pick the value that they think is likely to occur or the midpoint value (50% of cases are lower and 50% are higher). However, software project estimation techniques falsely assume that each individual estimate provided is the average of the range (arithmetic mean). This incorrect assumption leads to incorrect project estimates. If the arithmetic mean of a 30 minute commute time was the most likely to occur then that would suggest a bell curve distribution as shown below:



"Software development task timelines are similar to commute times in that we find the lower limit outcomes less likely to happen and with smaller potential savings than are higher limit outcomes with their much higher associated costs."

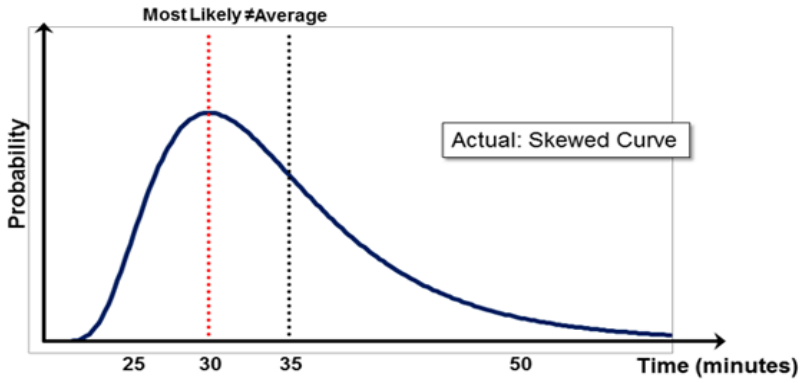
You may have already noticed the issue here – this distribution assumes that we are just as likely to be 10 minutes quicker than we are to be 10 minutes slower. Any commuter will tell you this is wrong.

The issue is that the time for someone to commute from work, just like a software development task, doesn't have a bell distribution curve but is in fact skewed. There is only a finite scope for saving time on a commute: assuming all the lights are green, there is light traffic, there are no incidents on-route and the weather is good (and we obey the speed limit!) we may save 5 or 10 minutes – but it is unlikely for all these factors to align and occur in a single trip.

On the other hand, there are numerous ways that the trip could take longer than expected due to bad weather, traffic congestion, incidents on-route, etc. Not only can a time overrun be more likely (requiring only a single thing to delay us) but the consequences to the commute time can be much more severe. In extreme cases, commute time could double, triple or even quadruple.

Software development task timelines are similar to commute times in that we find the lower limit outcomes less likely to happen and with smaller potential savings than are higher limit outcomes with their much higher associated costs.

The distribution is actually a skewed curve as shown below. While the 30 minute estimate example could be the most likely to happen and may be our typical commute experience, the average (in the form of the mean) is actually higher.




This difference and the failure to account for this statistical reality when combining multiple task estimates into a project estimate can lead to big differences at the project level. Individual mistakes are amplified and the project is, ultimately, underestimated.

"QEM takes into account those project tasks that are often left out of project estimates. Activities such as requirements clarification, task management and coordination, meetings, demos, testing and deployment are all included in the project estimate."

Suppose for example that we tracked our commute time over a one year period, recording 230 trips in total and plotted these at 5-minute intervals to give us the data below:

| Time | Count | Time | Count |
|---|---|---|---|
| 0-5 | 0 | 45-50 | 21 |
| 5-10 | 0 | 50-55 | 17 |
| 10-15 | 1 | 55-60 | 15 |
| 15-20 | 3 | 60-65 | 7 |
| 25-30 | 22 | 70-75 | 5 |
| 30-35 | 45 | 75-80 | 3 |
| 35-40 | 39 | 80-85 | 1 |
| 40-45 | 30 | 85+ | 2 |

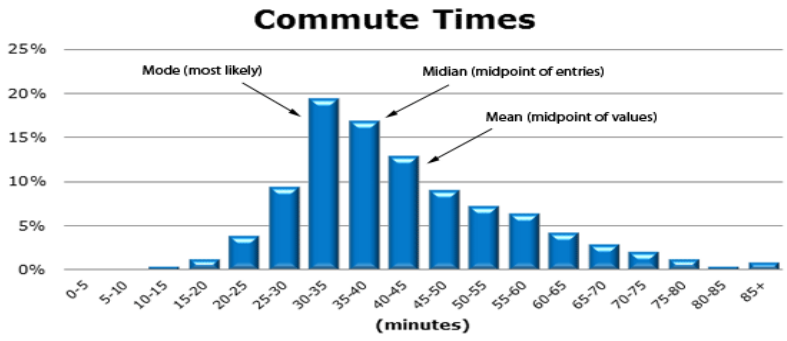
We would find that the sum time actually recorded would be considerably higher than if we multiplied our estimated trip time by the number of trips – even if that individual estimate was accurate! Here's why: The most likely trip time (called the mode in statistics) is 30-35 minutes and the mid-point (the median) is 35-40 minutes. If we use the upper range of these values to estimate the total we would get:

- 35 x 230 = 8,050 minutes total (using the mode)
- 40 x 230 = 9,200 minutes total (using the median)

If we look at the actual data though we find that the combined trips took a total of 10,350 minutes which gives an average (the mean) of 45 minutes. How far off would our estimates be?

- 10,350 – 8,050 = 2,300 minutes = 22% error (using the mode)
- 10,350 – 9,200 = 1,150 minutes = 11% error (using the median)

The examples shows how far off from reality a project estimate can be when the modes or medians of individual task estimates – rather than the means – are combined. We see below that the distribution of commute estimates is skewed. The right reflects the higher chance of a trip taking dramatically longer than the potential saving from it being quicker.



"QEM recognizes that most people naturally create single-point estimates for tasks or stories – and QEM is able to work with these (less-than-perfect) estimates."

The range of possible values for a typical software development task is an order of magnitude higher and wider than that of a simple commute. The associated costs are considerably greater, and as such, the risk of handling software development estimates incorrectly is all the more significant.

## How Does QEM Help?

The Quadrus Estimation Methodology is different for a few key reasons:

- QEM recognizes that way in which people estimate – and when forming overall project estimates, QEM employs Monte-Carlo simulation to aggregate individual estimates in a statistically correct way.
- QEM takes into account those project tasks that are often left out of project estimates. Activities such as requirements clarification, task management and coordination, meetings, demos, testing and deployment are all included in the project estimate.
- QEM recognizes that often not all project requirements are known up front. QEM provides a mechanism for estimating the percentage of known requirements (versus the percentage of unknown requirements), and the overall project estimate is appropriate scaled to reflect the reality of unknown requirements.

QEM recognizes that most people naturally create single-point estimates for tasks or stories – and QEM is able to work with these (less-than-perfect) estimates. The input to QEM consists, quite simply, of the single-point estimate that the developer feels is the most intuitive estimate (the median) together with an uncertainty factor (Low, Medium or High) to indicate the range of the distribution curve to use. If a task seems to have many unknowns or perhaps uses new technologies – or if it may have an element of research or invention – then the uncertainty will be higher than a task that is known and recognized (lower uncertainty).

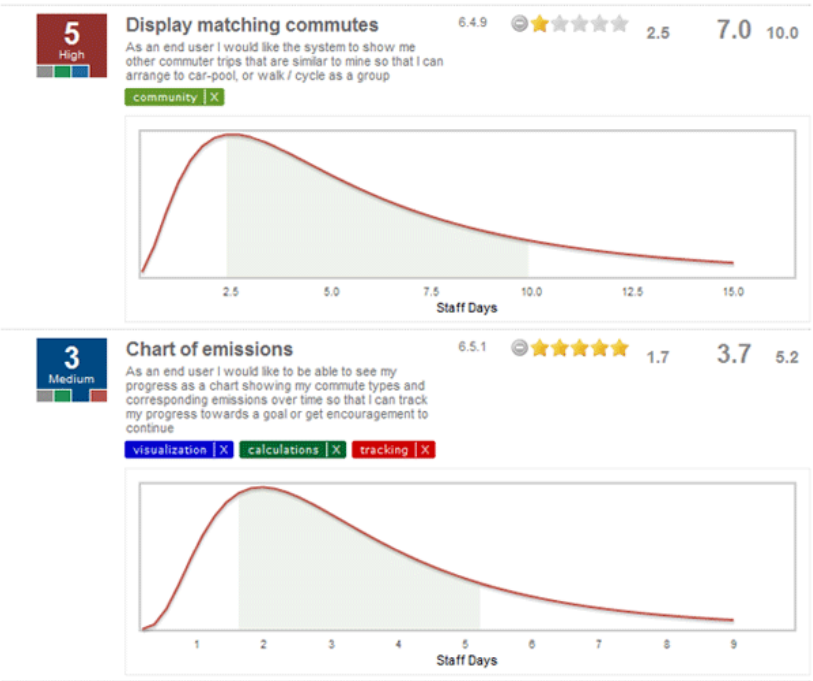
This uncertainty is very important; recognizing it exists can drastically change how much time we should assume a task can take. One story may have a lower numeric estimate than another but with a greater level of uncertainty could actually end up taking more time.

These inputs are not onerous to produce compared to the inputs demanded by other estimation methodologies that have been tried in the past (e.g., feature points, estimated lines of code, etc.). Together the single-point estimate/uncertainty pair provides enough information to calculate an expected average estimate for the individual story together with a range.

"Quadrus has codified the Quadrus Estimation Methodology as a modern AJAX-powered, intuitive and easy to use web application named the Quadrus Estimator™."

## The Quadrus Estimator Estimator

Quadrus has codified the Quadrus Estimation Methodology as a modern AJAX-powered, intuitive and easy to use web application named the Quadrus Estimator™. Here is an example from the Quadrus Estimator, showing the estimates and uncertainties for two user-stories and the resulting optimistic, expected (mean) and pessimistic effort values generated together with the distribution curve of probabilities:




The Quadrus Estimator has many features, such as support for the user-story importance to be ranked using a star-rating and enabling stories to be labeled. The star-rating and labels can be used to manage and filter the list of tasks and also to include or exclude stories from an estimate for "what-if" scenarios based on less important stories or certain features being postponed for future phases.

The simulation also accepts additional inputs to allow additional effort to be counted for which is often overlooked when creating estimates. This is in the form of "Story Glue" to represent non-development time that is still project related and must be accounted for (e.g., meetings, writing reports, demos, etc.).

## About Quadrus

Quadrus is a recognized leader in IT professional services and solutions. Headquartered in Calgary, Alberta, Quadrus has delivered hundreds of successful projects across Western Canada since 1993. We are committed to providing the highest quality service to our valued clients.

**Contact us:**
info@quadrus.com
www.quadrus.com
+1 (403) 257 0850

Another factor to consider is a measure of the overall percentage of stories already identified (to account for new requirements or scope-creep during the development process). This will be a subjective assessment based on how detailed or thought-out the user stories are. For a new system that includes an element of invention the "Percentage of Stories Known" will likely to be smaller than a project to mostly replace the functionality of an existing system. QEM offers a process for estimating such percentages.

The Quadrus Estimator calculates the total effort to complete the project in man-months. Along with the effort estimate, a combination of proven industry guidelines and research is applied to identify the ideal balance between resources (people) and duration (timescale). While either can be adjusted in the Estimator, the system shows the cost of favoring the other in terms of additional effort (overall cost), more resources (for faster delivery), or longer timescales (for team-size constraints).

The benefits of adding more resources to a project quickly diminish and at a certain point it actually results in a net negative benefit as the amount of coordination effort introduced negates any increase in productivity.

The exact benefit gained depends on the team dynamics: an already-formed team typically out-performs a new group working together for the first time. The exact skill match between the developers implementing the system and the skills assumed when creating the estimates will also have a significant impact on the project outcome: the difference in productivity between a highly skilled and average skilled developer has been shown to be in the region of one to two orders of magnitude (i.e., 10x to 100x) rather than the 10% to 20% often imagined.

Being able to make use of the Quadrus Estimation Methodology is just part of the Quadrus Advantage you get when you engage us for your projects. We also provide Software Estimation as a Service together with a program to train and mentor your staff on QEM and the Quadrus Estimator application, enabling you to gain the benefit of improved estimation on your other projects.