Good Planning – Not Development Methodology – Is the Key to Successful Project Delivery

IT development teams have long been searching for a magic formula of success in software development. How can teams find the greatest efficiency and the least cost, without sacrificing quality?

Teams have tried a myriad of methodologies to achieve these goals. Twenty years ago, developers looked to waterfall development as the answer. Waterfall then gave rise to object-oriented incremental or spiral, Rational Unified Development (RUP) practices.

Today, it’s all about agile development. Companies are investing lots of capital to develop agile methodologies and committing significant resources to train employees to work within agile frameworks.

Despite this investment in agile methodologies, many projects still fail, clients are unsatisfied, and IT departments often miss deadlines. Why?

The method doesn't matter. It's all about planning and how teams use resources.

When projects get behind schedule, team leaders almost automatically think that adding more staff will help. After all, many hands make light work, right? The more staff a team has working on a project, the faster and shorter development time needed.

The old adage doesn't necessarily apply in software development. In fact, the opposite effect can occur: adding more people to a project can make more work or even slow things down further. The additional person-hours may give the team a short-term boost, but over time, the team will have to manage higher costs and more connection points—each of which creates an opportunity for a mistake or defect, bringing additional risk to the project.

What's more, how teams allocate resources is just as important, if not more, to successful projects as which development method teams use. While methods have changed, the allocation of resources has remained paramount over the last two decades. This finding was underscored in QSM's latest analysis, which leveraged data from the most recent QSM Software Project Database update. The update includes new insights into agile development processes and the staffing models that agile teams are employing for their projects.

The following table provides a glimpse into how adding staff affects a project's outcome. It compares 390 applications of the same size featuring both 10,000 and 20,000 lines of newly developed code, with a significant portion using agile methods and tools. One sample uses an average of less than four people; the other, nine people or more. While the additional staff reduced the schedule by approximately 30 percent, the project cost actually increased by 350 percent. The additional staff also created 500 percent more defects that had to be fixed during testing.
In the graphs below, the lines represent the average behavior for size developed vs. schedule, effort, defect, and average staffing. Notice the large variation in effort and defects and small variations in schedule.

<table>
<thead>
<tr>
<th>Staffing Scenario</th>
<th>Size IU's</th>
<th>Schedule (months)</th>
<th>Effort (hours)</th>
<th>Cost @ 110 per hour</th>
<th>Peak Staffing</th>
<th>Defect found in Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average staff is greater than 9 people</td>
<td>10,000</td>
<td>3.85</td>
<td>8,790</td>
<td>$966,900</td>
<td>17.0</td>
<td>122</td>
</tr>
<tr>
<td>Average staff is less than 4 people</td>
<td>10,000</td>
<td>5.33</td>
<td>1,920</td>
<td>$211,200</td>
<td>2.7</td>
<td>18</td>
</tr>
<tr>
<td>Difference Absolute Value</td>
<td>None</td>
<td>1.48</td>
<td>6,870</td>
<td>$755,700</td>
<td>14.3</td>
<td>104</td>
</tr>
<tr>
<td>Percentage change</td>
<td>None</td>
<td>-27.7%</td>
<td>357.0%</td>
<td>357.0%</td>
<td>529.0%</td>
<td>577.0%</td>
</tr>
<tr>
<td>Average staff is greater than 9 people</td>
<td>20,000</td>
<td>4.71</td>
<td>11,230</td>
<td>$1,235,300</td>
<td>18.0</td>
<td>153</td>
</tr>
<tr>
<td>Average staff is less than 4 people</td>
<td>20,000</td>
<td>6.80</td>
<td>2,670</td>
<td>$293,700</td>
<td>3.0</td>
<td>29</td>
</tr>
<tr>
<td>Difference Absolute Value</td>
<td>None</td>
<td>2.09</td>
<td>8,560</td>
<td>$941,600</td>
<td>15.0</td>
<td>124</td>
</tr>
<tr>
<td>Percentage change</td>
<td>None</td>
<td>-30.7%</td>
<td>321.0%</td>
<td>321.0%</td>
<td>500.0%</td>
<td>427.0%</td>
</tr>
</tbody>
</table>
Over the past 15 years, QSM has performed this same study in five-year increments and has found the same results—staffing decisions have more of an impact on project success than any development methodology. Instead of relying on a particular development method, project managers should estimate early and use predictive analysis ... executives who use them gain actionable insights that help with course correction throughout a given project's lifecycle.

Individual projects may differ, but all projects can benefit from a step-by-step process that uses predictive analysis and early estimation to more accurately account for staffing needs.

1. Gather all current project data in one place.
   It's important to know the status of current resources before beginning a new project. Creating a central place that ... available to work on a given project and where that project fits into an organization's overall development cycle.

   Software lifecycle management solutions provide the tools through which team leaders can monitor projects that are in ... completes and allow users to generate subsequently more accurate forecasts with each project entered and tracked.

2. Examine historical data.
   Once team leaders have current project data in place, they should run a forecasting analysis to see if they have the ... give a good idea of how many resources may be needed to complete the project within the budget and time allotted.

   Unfortunately, deadlines and client expectations don't always match. It is up to team members to use available historical ... (to meet goals or save money) do so from a position of strength – they have the numbers to back up their projections.

3. Compare project data to industry trends.
   Teams can compare forecasts to industry-wide trends to see if their numbers are tracking similarly. A software lifecycle ... data against their own data points to see how their staffing and resource levels compare with other, similar projects.

   Consider the following example: PI represents a project's "Productivity Index," a measurement taken from QSM founder Larry Putnam's Putnam Model, which has been the standard in software estimation for nearly 40 years. The red dots represent a company's current resource level, while the blue lines indicate industry averages.

   Note that the productivity assumption is much higher than normal, resulting in a shorter than usual schedule and effort – a recipe for disaster at delivery time.

   A visual representation makes it easier to see where adjustments need to be made to bring a project more in line with the rest of the industry. Tech teams can tell if they are falling behind the curve, ahead of it, or right where they need to be.

4. Determine a project's impact on an organization's overall budget.
   No matter which project they work on, teams should be able to see the potential impact an organization's projects have on ... track when specific projects may be deviating from estimates and threatening to throw an entire IT budget out of whack.

   Here's another example of an at-a-glance view. The red dots represent given projects. The goal is to get them within the green rectangle – the "sweet spot."

   Having this information at the ready will help teams identify projects that may be in danger of becoming risky or wasteful. Once again, predictive data helps teams course correct and get back in the green zone.

5. Use available resources fully.
   Perfecting the way staff are allocated increases efficiency and minimizes the risk of project failure. In this final ... of the optimization process and match demand with current capacity, which ultimately leads to a realistic portfolio plan.

   With an entire portfolio of staffing needs captured at a glance, executives can more accurately determine how to allocate ... project on time and within budget? Are new personnel needed? If so, how many should be hired to get to "peak staff?"

   Again, methodology clearly doesn't matter here. Knowing which resources are programmatically optimized how, makes cost buffering...