THE ART OF BALANCING SCOPE, TIME, AND TEAMS IN SOFTWARE DELIVERY PROJECTS

Insights and lessons from 13 years of experience managing software projects, exploring hybrid methodologies, roadmap planning, and practical strategies for delivering value in complex environments.

By: Fredy Mateus (Author)
Contributions from: Deepika Jonnalagadda (Review & Additional Text)

Our main responsibility as project managers is to lead the project team in achieving the project's objectives and delivering intended value within agreed constraints. Some of these constraints are commonly referred to as the triple constraint: time, cost, and scope (Project Management Institute [PMI], 2021).

In software delivery, a key project objective is often to release the first version of the intended product to production and transition it to operations, enabling the organization to begin realizing strategic benefits and return on investment (PMI, 2021). To accomplish this objective, product features are implemented according to defined requirements, and a project plan is established to ensure that Product, Architecture, Development, DevOps, QA, and UAT teams are aligned toward shared project goals (Sommerville, 2016).

Cost is driven by the effort and resources required across these teams, while the timeline is constrained by scope objectives and available team capacity. As a result, project management becomes the practice of balancing product scope, delivery timelines, and team capacity (PMI, 2021). The key question then becomes how these constraints can be effectively managed to achieve the ultimate project goal.

In software development, however, this has always been challenging. Based on my 26 years of experience in the industry, these constraints have rarely remained stable for long. When I started my career, waterfall approaches dominated (Royce, 1970), and the software development lifecycle typically began with a project schedule that time-boxed tasks for requirements elicitation, followed by analysis, design, coding, testing, and deployment. These activities were supported by long, linear schedules that were often inaccurate and not feasible, since the practice of refining the schedule as scope information became clearer—providing better insight into solution complexity and enabling more accurate implementation estimates—was not part of the regular waterfall approach (Sommerville, 2016).

This project approach attempted to answer the question of when the most feasible release date would be. However, in practice, that date was always shifting. Project schedules were rarely accurate, release dates changed frequently, teams were pushed to work long hours, and project tasks were cut—especially those related to QA and documentation. As a result, solutions were often released to production with only a partial level of quality, limited documentation, and inconsistent implementation of best practices (Sommerville, 2016).

Fast forward 26 years, and in my most recent "agile" project, we still relied on a waterfall-style schedule for the overall release plan, as we faced the same critical question: When will the MVP be released, including all must-have and selected nice-to-have features? I want to describe the approach we used to answer this question and manage the project plan. It was not a perfect plan, but it proved to be a practical way to address the problem.

However, our project plan eventually had to change due to risk management decisions and shifts in strategic product priorities. Despite this, I believe the experience offers valuable lessons that can be applied to other projects (PMI, 2017).

Our Program Director and I—acting as Director of Development—were responsible for identifying the release date. The project followed a hybrid Scrum methodology (Schwaber & Sutherland, 2020), where some practices were applied as designed, while others were adapted to our context. In particular, scope definition, backlog grooming, and sprint planning leaned more toward a sequential approach than a traditional Scrum model.

When the initial direction from the business was presented, it included several hundred line-item features spanning an MVP and a General Availability (GA) release. Through a structured ballpark estimation exercise with cross-functional SMEs, we projected a delivery timeline that was not commercially viable for the organization.

In collaboration with executive leadership, we entered multiple rounds of disciplined scope negotiation to redefine what constituted a viable MVP versus deferred functionality. Each iteration required recalculating timelines based on proven team velocity, fixed headcount, existing infrastructure, and architectural constraints.Leadership required a forecast with 90% confidence, which meant formalizing assumptions, incorporating risk buffers, and stress-testing capacity scenarios before presenting a revised plan.

After several iterations of estimation, scope negotiation, and revised forecasting, we finalized an MVP timeline followed by a GA release. The program roadmap was reviewed, externally validated, and formally approved.

With these conditions in place, we adopted a program-based approach. The Program Director worked closely with the product team to identify and prioritize the feature list. Once priorities were defined, the development and QA teams collaborated to understand scope, provide high-level estimates, and identify dependencies. This information was then used to create a program plan (PMI, 2017).

Each program had a fixed duration of four three-week sprints, based on team capacity. At this stage, the plan resembled a waterfall-style schedule, defining the expected scope for each sprint within the program. The product team followed this plan sequentially to ensure that features were ready for development when needed.

The next step involved backlog grooming and sprint preparation. Features were selected for development, sprint plans were refined, and each sprint was launched only after feasibility was confirmed against team capacity. I worked closely with each developer to define weekly goals aligned with sprint objectives, giving the team clear visibility into expectations and deliverables (Schwaber & Sutherland, 2020).

We worked as a cohesive unit with the QA team, with the shared goal of delivering features ready for UAT (Sommerville, 2016). When QA identified blockers, the development team prioritized bug fixes to ensure features could be completed within the sprint. This was possible because capacity had been intentionally split between feature development, bug fixes, administrative work, meetings, and context switching. This allocation was critical to maintaining a realistic and sustainable plan.

As execution began, we experienced some initial challenges, including lower velocity. However, as the team gained momentum, the process stabilized and began to perform as expected. We were tracking ahead of schedule, and the hybrid approach appeared to be working effectively.

Unfortunately, a change in product vision ultimately put the project plan on hold. Priorities had to shift due to a new technical direction and significant changes to the product scope, demonstrating how even a well-executed roadmap can be disrupted by strategic changes outside the delivery team's control.

Going back to the initial question we were trying to solve—when will the MVP be released, including all must-have and selected nice-to-have features?—we defined a project methodology that could be extended to other projects and domains, assuming the product vision and corporate strategic priorities remain stable (PMI, 2017).

Our methodology combined roadmap planning, a waterfall-like program plan, a Scrum-like detailed sprint plan, and weekly goal execution within each sprint, including prioritization of bug fixes to achieve overarching goals between the Development and QA teams (Schwaber & Sutherland, 2020).

Beyond this, several challenges remain. First, how to conduct effective risk management when potential changes in the product vision may not be visible in advance. Second, how to handle a UAT schedule when the business team has limited capacity and is primarily focused on feature definition. Third, how to ensure production readiness, considering other technical elements such as infrastructure, security, operational readiness, communication, business processes, and governance (Sommerville, 2016).

Another interesting topic to explore is how to accurately control the budget and costs for a project of this size in a startup-like environment, where funding management and proper expectation-setting for potential investors are critical. There are clearly many valuable topics to continue exploring in this discussion. For now, I want to conclude that hybrid project methodologies hold significant potential for addressing fundamental project management questions. By leveraging project management practices and learning from similar experiences, we can adopt a stronger approach to software delivery. The focus should be on roadmap planning, program scheduling, sprint and capacity planning, controlled sprint goal execution, and tracking progress against the program plan—which should continue evolving as more information becomes available.

References

- Project Management Institute. (2021). A guide to the project management body of knowledge (PMBOK® Guide) – Seventh Edition. Project Management Institute.
- Project Management Institute & Agile Alliance. (2017). Agile Practice Guide. Project Management Institute.
- Schwaber, K., & Sutherland, J. (2020). The Scrum Guide: The definitive guide to Scrum: The rules of the game. https://scrumguides.org/scrum-guide.html
- Royce, W. W. (1970). Managing the development of large software systems. Paper presented at the Western Electronic Show and Convention (WESCON).
- Sommerville, I. (2016). Software engineering (10th ed.). Pearson.
- Project Management Institute. (2017). Pulse of the Profession®: Success rates rise: Transforming the high cost of low performance. Project Management Institute.

Author’s note: This article was initially edited for clarity and flow with the assistance of AI-based language tools. However, all concepts, analysis, and conclusions presented are the author’s original work.