7pace

LEAN REPORTING

HOW SOFTWARE TEAMS CAN MAKE DATA-DRIVEN DECISIONS WITHOUT DATA-DRIVEN FRUSTRATION

WHO SHOULD READ THIS EBOOK?

Software teams that make data-driven project decisions are more likely to deliver projects on time, within budget, and meet user expectations. Yet most teams fail to adopt reporting tools that give them the necessary project data.

This is a practical guide for decision-makers who lead software teams and want to make more data-driven decisions to improve the odds of success.

In this guide, we'll explore the top reasons software projects fail, how reporting can minimize the chances of software project failure or overruns, what minimum viable reporting is, and practical examples of minimum viable reporting in action.

TABLE OF CONTENTS

WHY (MOST) SOFTWARE PROJECTS FAIL

Projects that don't deliver the promised benefits within budget and time are failures.

PROJECTS THAT DON'T DELIVER THE PROMISED BENEFITS WITHIN BUDGET AND TIME ARE FAILURES.

A whopping, "99.5 percent of big projects fail to deliver all their core promises", reveal Bent Flyvbjerg and Dan Gardner, the authors of How Big Things Get Done.

Tell me if this sounds familiar: The initial excitement around a new, big, and important software project quickly dissipates into a scattered trail of documents.

Soon, the only way to get progress updates or find project data is by organizing a stand-up meeting or typing up a frantic "Dear all" email. And because getting the right project data costs too much time and effort, you make key decisions such as resource allocation, change management, and testing based on gut feelings and immediate resource availability.

Eventually, developers relegate reporting to the lowest possible function – an afterthought that needs to be updated just before each sprint ends.

The data is questionable.

Your team is frustrated.

And by the end of the project, you're lucky to even get enough project updates to fill out the project report.

But what's all this that got to do with software project failures and overruns? Everything.

The top 4 reasons for software project failures, delays, and cost overruns are:

1. Superficial planning
2. Mistaking the means for benefits
3. Not evolving your plans based on reality
4. Not choosing the best team

Let's dig into each one and unpack how they affect your project's success.

SUPERFICIAL PLANNING

How long will it take to build a piece of software?

This question haunts most software teams because it feels impossible to answer.

As such, many teams abandon hope for creating accurate estimates and wing it based on intuition or "gut feeling."

Obviously, this creates problems.

Superficial planning based on intuition alone fails to identify big problems that come back to haunt the project during the execution phase.

Planning isn't about filling report templates, creating flowcharts, or defining software features. It is about creating a detailed plan that can evolve based on feedback,

progress, and changing requirements. However, most software projects take a leap of faith, relying solely on ingenuity.

But overconfidence results in underestimating the amount of time needed to complete a task.

OVERCONFIDENCE RESULTS IN UNDERESTIMATING THE AMOUNT OF TIME NEEDED TO COMPLETE A TASK.

Psychologists Daniel Kahneman and Amos Tversky who coined the term planning fallacy have demonstrated in several ways that people are prone to making inaccurate predictions when they rely on intuitive judgments alone.

The solution to planning fallacies is clear.

Data.

Using best guesses and complex user story estimation processes can provide a framework. But ultimately, estimates, plans, and budgets should be informed by actual, real-world data from current and past projects.

PRIORITIZING MEANS OVER OUTCOMES

Many teams get engrossed in using the right technology framework, debating technology trade-offs, and discussing testing protocols. As a result, user outcomes take a back seat.

Apple's legendary founder Steve Jobs once said, "You've got to start with the customer experience and work back toward the technology." Successful project teams understand what the end outcomes are and then work backward to achieve them. Not the other way around.

This "prioritizing user outcomes" mindset has helped Apple successfully commercialize many innovations. For example, they adopted the mouse from Xerox and more recently Siri from Nuance. Apple found the best possible technology option to deliver a great user experience.

Similarly, successful teams take an outside view, to choose the best option for an end outcome.

At a practical level, this mindset can help you curb project overruns on many levels. For instance, you may choose an open-source framework over a custom development project. Or you may consider buying features instead of engineering them in-house.

RIGID PLANS WITHOUT DATA

Every plan is a guess.

Even if it's informed by data, the plan that you set forth at the beginning of a project is practically guaranteed to deviate from reality.

As such, you need even more data—real-time data—that helps you quickly identify issues and adapt the plan to the real-world circumstances that are affecting your project.

Scheduling software sprints at the beginning of a project and not evolving the plan based on feedback and data exacerbate the planning fallacy.

Projects that don't use data such as how backlogs are progressing, time-tracking patterns of their developers, and tracking completed tasks are likely to fail.

Successful project teams see planning as a chance to test their ideas, understand what works, and constantly update their project plans.

But software teams are handicapped by the challenges of collecting quality data. As a result, they fail to iteratively validate assumptions and refine forecasts based on real data.

POOR RESOURCE PLANNING

The success of any project depends on getting the right team with complementary skills.

All software projects eventually encounter changes in scope, requirements, and team members. But typically managers don't have a centralized and updated resource scheduling system. This in turn hinders ideal resource allocation.

As a result, it is quite common for software teams to double-book talented developers. Such ad-hoc resource allocation based on "who's available right now" results in project delays.

HOW LEAN REPORTING REDUCES PROJECT FAILURES

Most of the challenges we covered above can be solved in a simple way.

Reporting.

Having better data allows you to create smarter plans, identify issues earlier, and resolve problems more quickly.

But there's a problem.

Most teams struggle with reporting. It's often seen as a burden or a barrier rather than a helpful tool for the team. This leads to unreliable data, dubious progress reports, or no reporting at all.

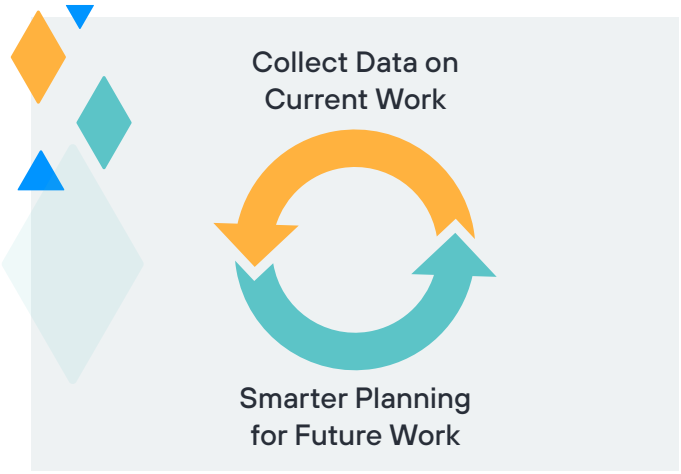
Lean reporting is a system for capturing the critical information your team needs to make smarter decisions while creating as little organizational drag as possible.

It's about doing just enough reporting to glean the value while minimizing the additional work and overhead that most teams have come to associate with things like filling out timesheets or providing in-depth project updates.

INFORMS CURRENT & FUTURE PLANNING

Reporting is a cyclical process.

The data you collect today helps inform your current project.



But, perhaps more importantly, it provides the data you need to plan future projects.

Lean reporting should provide the data you need for both purposes:

1. **Current planning:** Keeps you grounded in the realities of your software project with reliable data that you can compare to the initial plan
2. **Future planning:** Provides data on analogous projects, work items, user stories, etc that you can use to better estimate time and effort on upcoming work

Lean reporting provides granular data on which team members spent how much time on what activity. Without this data, you cannot identify resource constraints, resource overloads, or optimize workflows.

Lean reporting helps you compare progress against the estimated effort, time, and budget. So it gives you enough evidence to prevent projects from spiraling into huge resource sinks.

Lean reporting also aids you in identifying overworked and underworked team members. It also enables you to make better resource-leveling and resource-scheduling decisions.

MEASURES PROGRESS

Lean reporting helps you differentiate the ***busy work*** from ***important work.***

Akin to the lean methodology, lean reporting strives to create the most value while eliminating redundant and time-consuming processes.

For instance, tracking metrics such as lines of code, number of tickets closed, and code commits alone can result in developers gaming these metrics to perpetuate a viscous cycle of low-value activities.

Why you should stop measuring these productivity metrics

Lines of code: This metric can lead to busy work and doesn't necessarily result in a better product.

The number of tickets closed: People can game the system by getting creative in opening tickets or picking ones that are easy to fix.

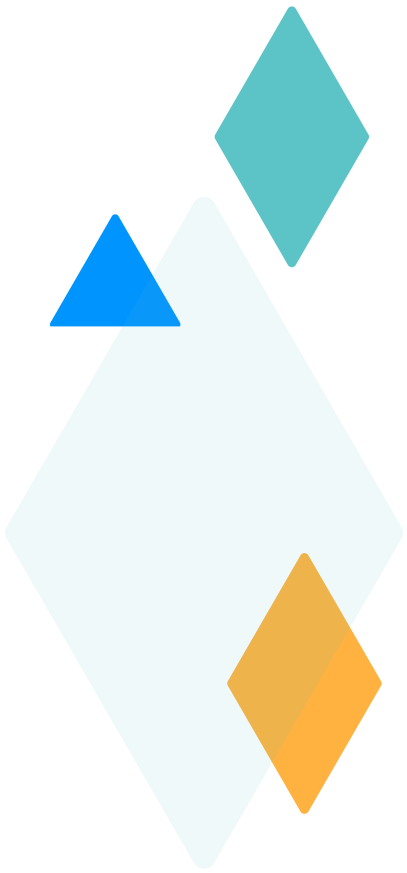
The numberof commits: This KPI can result in small and frequent commits that don't move the needle.

So in addition to tracking such metrics, you must also give weightage to the importance and quality of work. Ideally, the metrics you track must be measurable and independently verifiable.

Lean reporting helps you correlate project metrics with business outcomes such as improving revenue and average customer lifetime values. Some productivity metrics to consider include:

- Pull request size
- Work-in-progress
- Time-to-open
- Code coverage
- Time spent on testing
- Time spent on fixing bugs
- Average lead time from “on hold” to accept

Data from a lean reporting tool provides objectivity because it helps you understand which metrics drive key business outcomes.



TRACKS IMPORTANT KPIS

KPIs are crucial for evaluating the efficiency of your engineering team. Lean reporting helps you track KPIs across your project's development, operations, and feedback phases.

DEVELOPMENT PHASE KPIS:

Lead Time measures the total time it takes for a task or feature to go from initial request to completion, including the time spent waiting in queues. This KPI tells you how quickly the team can deliver new features and address customer requests.

Cycle time measures the time it takes to complete a task or feature once the team starts working on it, excluding any waiting time. This KPI helps you identify bottlenecks or inefficiencies in the development process.

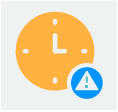
Time to market is the duration from an idea to its release. Organizations with shorter TTMs will respond quickly to the market, capitalize on opportunities, and stay ahead of the competition.

Time in Queue measures the waiting time before the team can start work on a task or feature. This KPI is useful in optimizing internal processes and reducing waiting times.

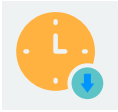
Code Review Turnaround Time is the duration it takes for a team member to review, provide feedback, and approve or request changes to a code submission. Focusing on this KPI can improve code quality standards and minimize development delays.

OPERATION PHASE KPIS:

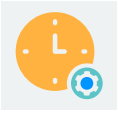
Time to Value (TTV) measures the duration it takes for a customer or user to start benefiting from a new feature after its release. This KPI can be used as a proxy for user satisfaction.



Time to First Response (TTFR) measures the time it takes for the team to provide an initial response to a customer issue, request, or inquiry. A shorter TTFR demonstrates the team's commitment to addressing customer concerns.

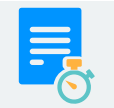


Time Between Failures (TBF) is the average time between system failures or incidents. This KPI evaluates the stability and reliability of your software.

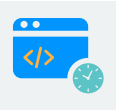


Time to Deploy (TTD) is the duration it takes to deploy a new release, update, or bug fix to production. This KPI evaluates the efficiency of your deployment process.

FEEDBACK PHASE KPIS:



Mean Time to Resolution (MTTR) measures the average time it takes for the team to resolve issues, such as bugs or incidents, from the moment they're reported. A shorter MTTR indicates an efficient incident response process, which in turn, minimizes downtime.



Deployment frequency is the number of times the team deploys new features, updates, or bug fixes to production within a specific time period, such as per week or month. This KPI helps you gauge the team's agility and responsiveness to customer needs.

Time to Feedback (TTF) measures the duration from releasing a new feature or update to receiving feedback from users or customers. This KPI assesses how quickly your team gathers user insights and makes data-driven decisions.

Lean reporting provides valuable data and insights on your team's productivity, agility, and overall performance.

HOW TO IMPLEMENT LEAN REPORTING

Lean reporting relies on using the right tools to collect critical data when, where, and how work is already happening.

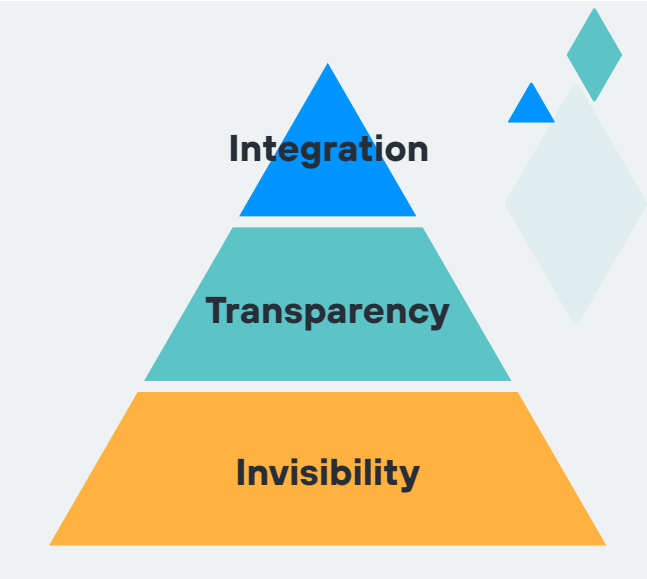
In other words, the holy grail is a reporting system that's fully automated.

No one on your team needs to pause or think or act in order for the relevant data to be collected. But that data is readily available whenever management or the team needs it to make decisions about the project.

This framework relies on 3 core components:

1. Invisibility
2. Transparency
3. Integration

Each one is critical for implementing a Lean Reporting system that provides important data without creating extra headaches.



INVISIBILITY

As we know, reporting is challenging because it requires work. It feels like wasted time. Members of your team loathe to take time away from "real work" to fill out numbers in a spreadsheet.

That's why the non-negotiable element of Lean Reporting is **invisibility**.

The data collection itself must be invisible. Or near invisible.

Data must be collected through a deep integration with the existing tools and systems your team already uses.

TRANSPARENCY

Just because the mechanism by which data is collected is invisible doesn't mean the data collection itself should be a *secret*.

If the team feels their work is being monitored in a clandestine way, that can breed resentment and distrust.

Work and project data shouldn't be a weapon used against the team.

It should be a weapon used ***by*** the team.

That leads us to element number three.

INTEGRATION

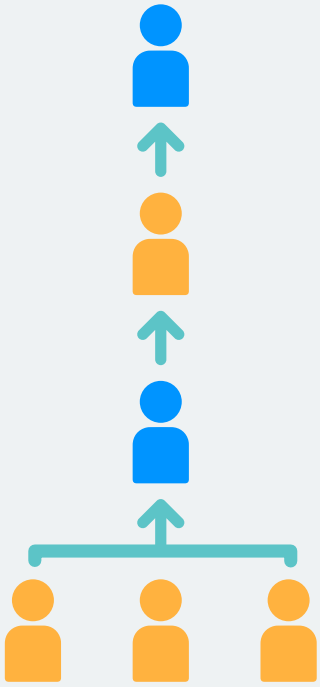
Finally, the data needs to *feel* valuable.

That means it needs to become an integrated part of the work planning process.

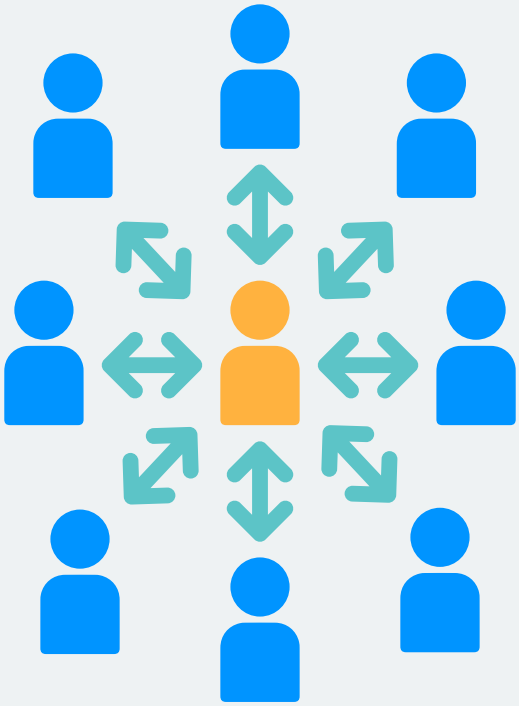
Again, the data is a weapon used by the team to answer important questions like, "how much time will this feature take?" or "how was the last release's stability versus previous releases?"

This means creating space within the existing work structure to analyze and apply the data that's being collected. Reporting shouldn't be purely a tool for top-down management. It's also a tool for self-management and informed decision making.

Traditional reporting



Integrated lean reporting



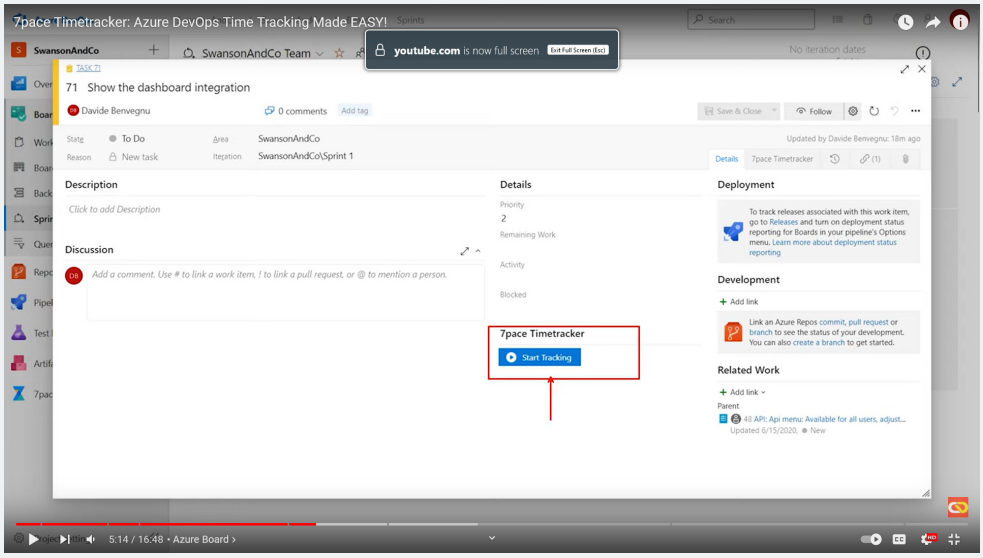
LEAN REPORTING IN ACTION: TRACKING TIME DATA WITH 7PACE

An effective, developer-friendly, and customizable time-tracking solution provides the ideal foundation to implement lean reporting systems.

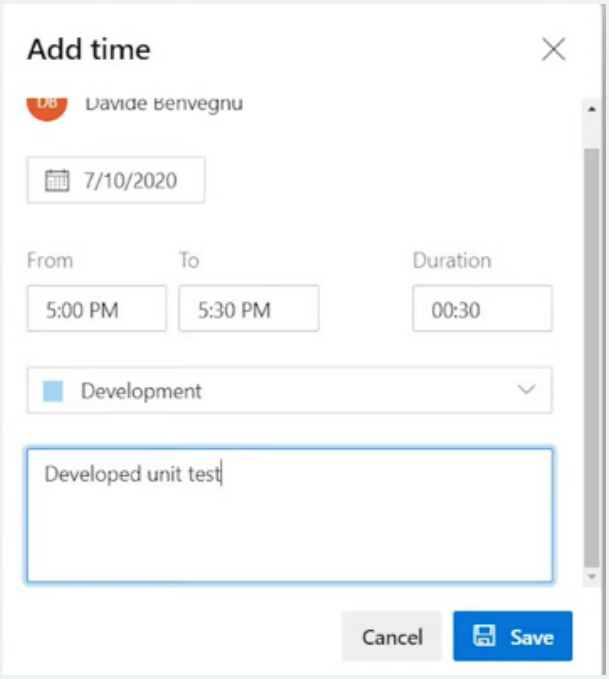
7pace is a developer-friendly and customizable time-tracking solution. It provides actionable project data without manual effort. And it helps you create customized reporting workflows.

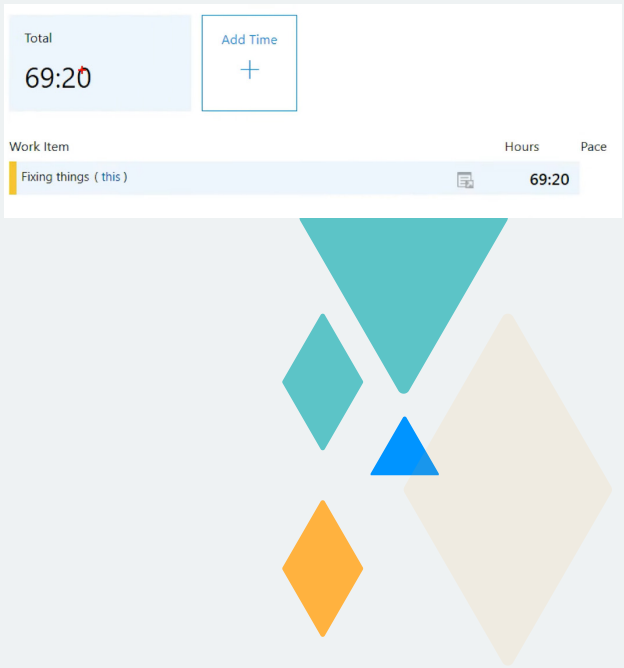
7pace differs from generic time-tracking or reporting tools in the following 4 ways:

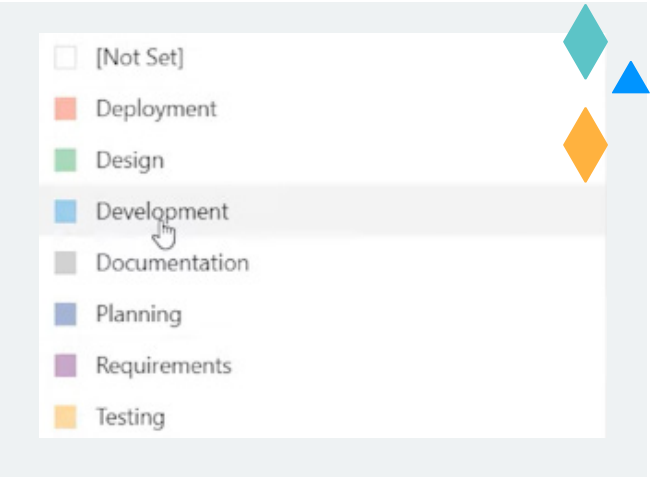
1. 7pace Integrates Into Your Development Environment
2. 7pace integrates into your Azure DevOps and GitHub environments. Instead of putting the time-tracking onus on your developers, you can use 7pace to automate time-tracking.
3. For example, Azure DevOps comes with built-in features to define area paths, query by iteration paths, and assign roles to people in your organization. But it doesn't offer a simple one-click option to track time across your team's development activities.
4. 7pace sits as an extension inside your Azure DevOps environment and offers an intuitive, one-click time tracking button for each task, right from your Azure Boards dashboard.



Starting and stopping the time tracker automatically adds the development effort of each task to your overall project data.







7pace also gives your developers the option to manually add time. It also helps categorize time into various activity types such as research, design, development, testing, and documentation.

7PACE AUTOMATICALLY EXTRACTS KEY PROJECT INSIGHTS

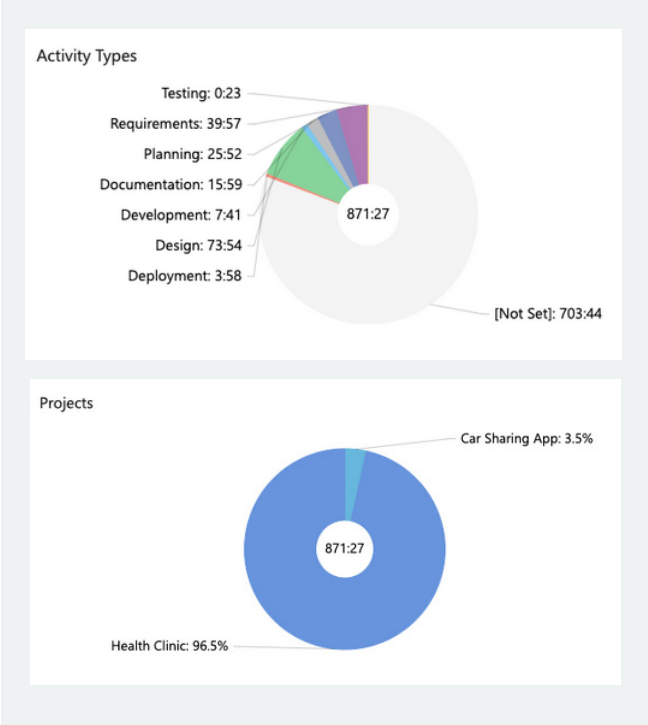
In addition to simplifying time-tracking, 7pace helps you to extract key project insights without writing any manual SQL or XML queries.

For instance, the tool automatically gives you a time split up by activity type.

You can use this to spot patterns. For example, to budget the design phase of an upcoming project, you could refer to the actual time spent on designing a similar project in the past.

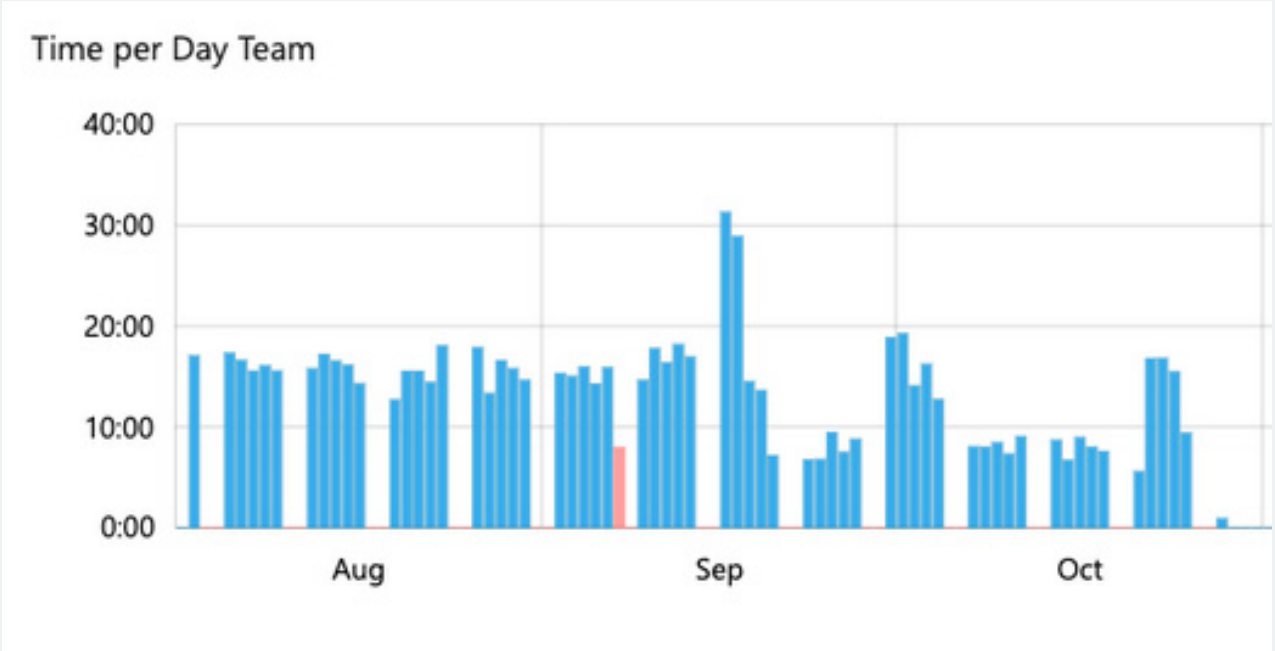
You can use this project data to identify resource constraints and opportunities for optimizing workflows. For instance, if the actual design phase costed twice the budgeted effort, then further investigation could reveal unknown resource unavailability or inaccurate budgeting problems.

7pace also automatically summarizes the total time spent across various projects.



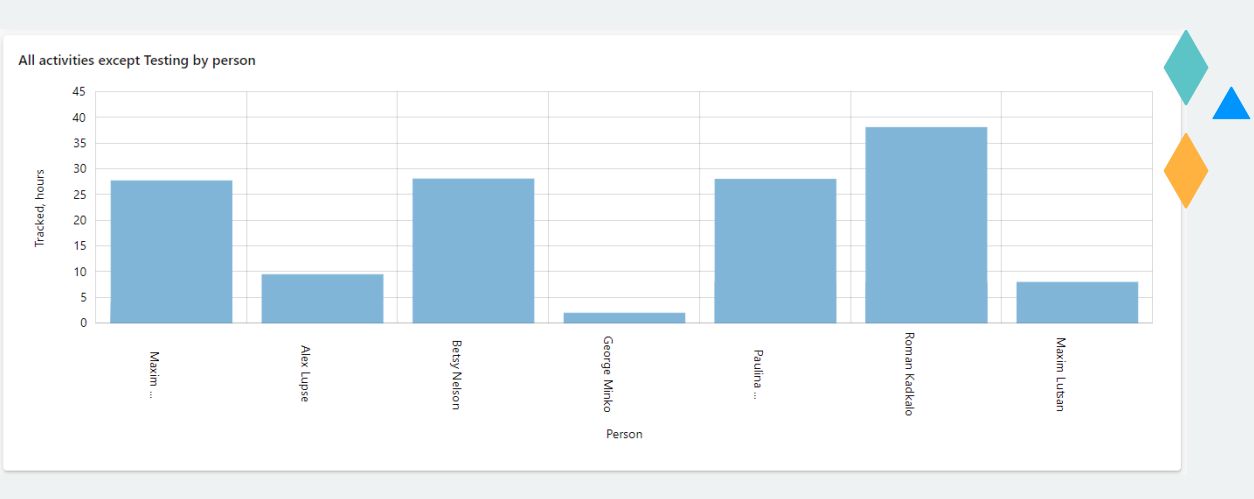
You can use this information to discern the profitable projects from the ones that become resource drains.

The tool helps you understand your team's workload spikes across a given time period.



For example, the above time per day team chart shows that a few team members have worked overtime in September. So you'll need to investigate if this was caused due to resource overload, improper planning, or unforeseen changes.

7pace gives you a visual breakup of team members' time breakdown mapped against specific project activities. For instance, the below chart shows the number of tracked hours for each team member across all activities except testing.

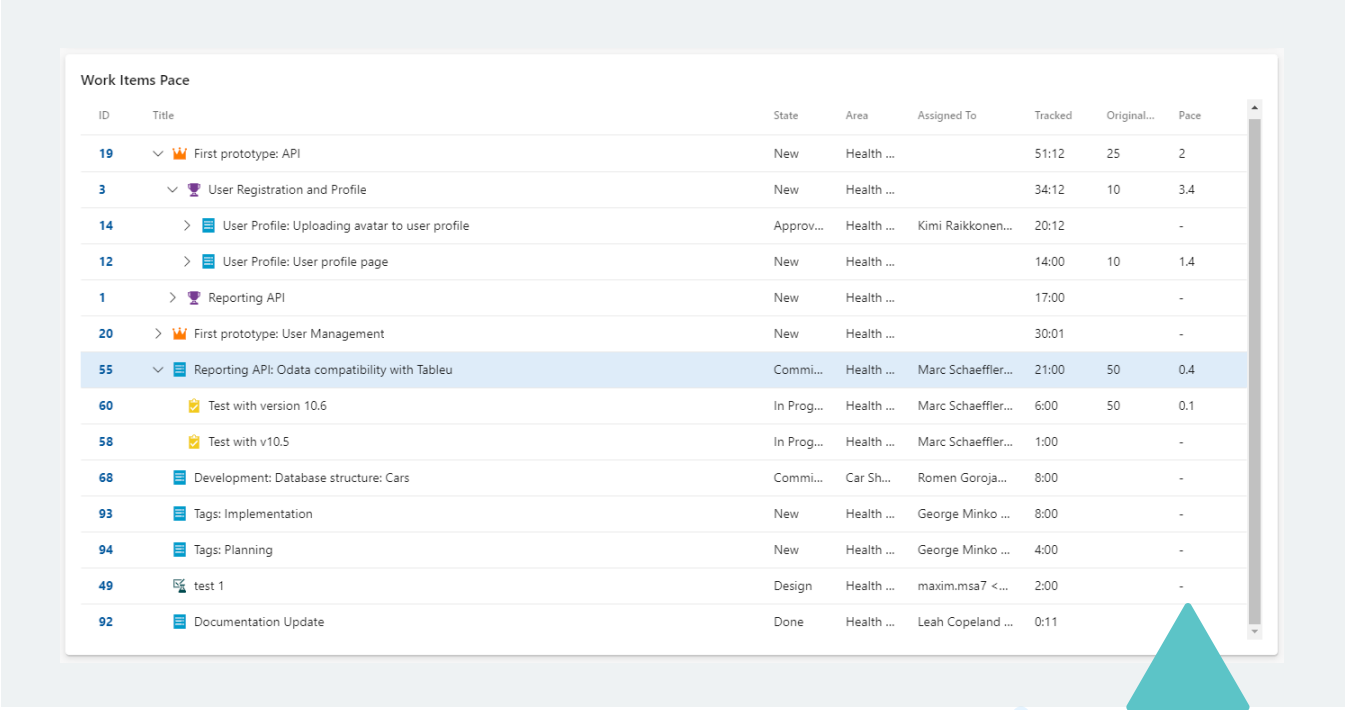


7PACE IMPROVES DECISIONS AND AUTOMATES BILLING

Because 7pace keeps track of all project data, you can use historical data to better estimate, forecast, and plan upcoming projects.

"Tracking historical plans and actuals is the fundamental first step in overcoming the planning fallacy", says Prof. Grushka-Cockayne, from the Darden School of Business, University of Virginia. "You should track your performance because if you start with that – let alone anything more sophisticated – you will improve."

7pace maintains a repository of all your budgeted plans and actual project data. You can understand the health of any sprint of a particular project by clicking the **Iteration** menu option.



Work Items Pace

ID	Title	State	Area	Assigned To	Tracked	Original...	Pace
19	First prototype: API	New	Health ...		51:12	25	2
3	User Registration and Profile	New	Health ...		34:12	10	3.4
14	User Profile: Uploading avatar to user profile	Approv...	Health ...	Kimi Raikkonen...	20:12		-
12	User Profile: User profile page	New	Health ...		14:00	10	1.4
1	Reporting API	New	Health ...		17:00		-
20	First prototype: User Management	New	Health ...		30:01		-
55	Reporting API: Odata compatibility with Tableu	Commi...	Health ...	Marc Schaeffler...	21:00	50	0.4
60	Test with version 10.6	In Prog...	Health ...	Marc Schaeffler...	6:00	50	0.1
58	Test with v10.5	In Prog...	Health ...	Marc Schaeffler...	1:00		-
68	Development: Database structure: Cars	Commi...	Car Sh...	Romen Goroja...	8:00		-
93	Tags: Implementation	New	Health ...	George Minko ...	8:00		-
94	Tags: Planning	New	Health ...	George Minko ...	4:00		-
49	test 1	Design	Health ...	maxim.msa7 <...	2:00		-
92	Documentation Update	Done	Health ...	Leah Copeland ...	0:11		

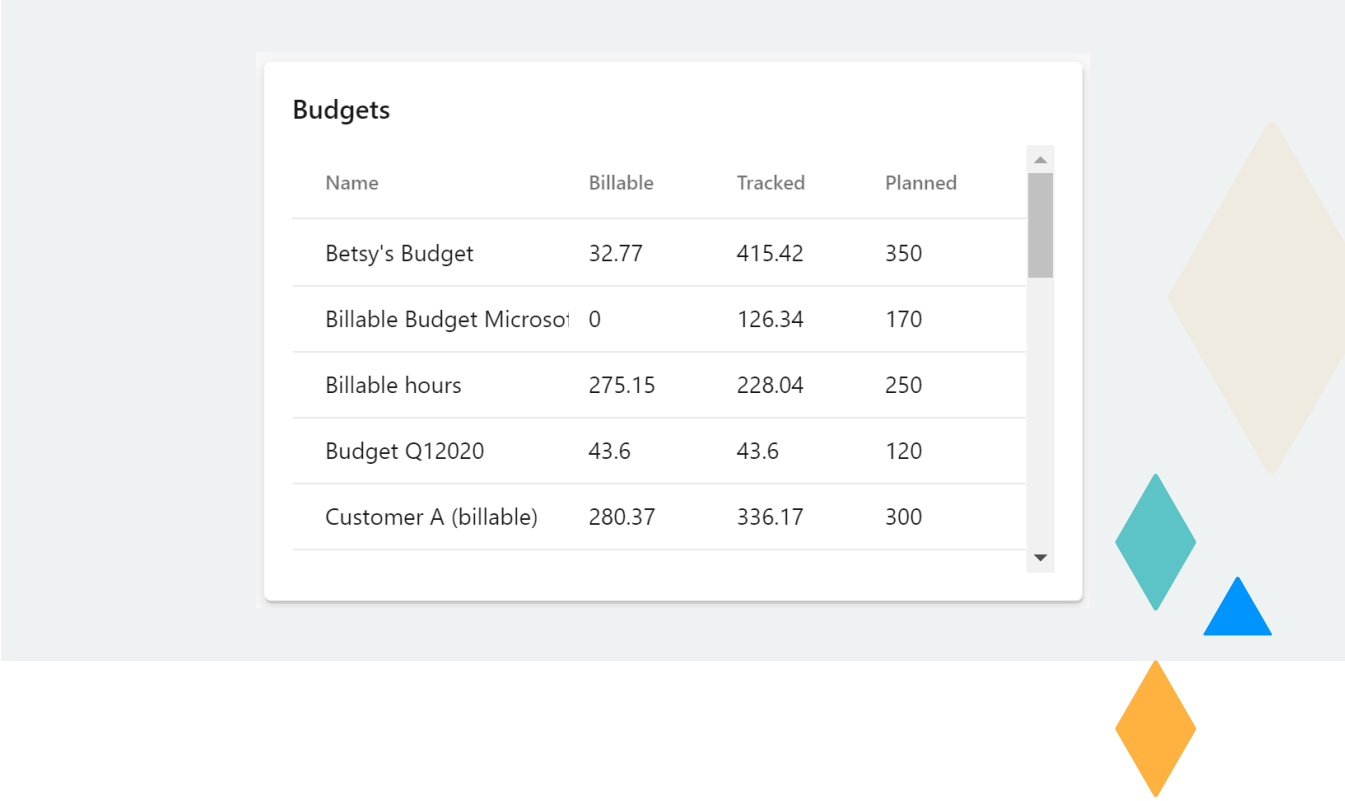
The corresponding dashboard provides the following information:

- Most recent **Iteration Path**
- Person **assigned** to each work item
- Current **state** of each work item
- **Actual time tracked** for each work item
- **Original time estimated** for each work item
- Number of **effort** points estimated for each work item
- **Pace** (number of hours divided by estimated effort) of each task

These data points help you understand discrepancies between your estimations and actuals. Over time, this improves forecasting for upcoming sprints, which in turn, minimizes budget and time overruns.

7pace enables you to make efficient resource allocation decisions based on the time tracked per person per project and shortfalls per person data.

7pace also simplifies your billing workflows by summarizing the billable hours, actual time spent, and original estimate for each project.



7PACE API IS FULLY CUSTOMIZABLE

7pace offers a secure and fully customizable API that delivers all the flexibility you need.

For instance, you can connect the 7pace reporting API to Power BI, C#, Python, Javascript, or Postman. Some use cases of the 7pace API include:

Creating a pivot table in Excel

Connecting to the Postman API platform

Sending project data to a Python or Javascript app

You could use the API to create custom fields in your reporting using the OData syntax.

7pace provides multiple API endpoints that you can hook with a trigger or event to customize your time-tracking workflows. The 7pace API also allows you to connect project data to a visualization tool such as Tableau, Qlikview, Microsoft PowerPivot, or Microsoft Excel.

START PRACTICING LEAN REPORTING WITH 7PACE

7pace delivers actionable project insights and alleviates your time-strapped developers from the mundane manual tasks of updating timesheets every day.

7pace automatically pulls data from your Azure DevOps and GitHub environments to deliver granular project reports. It comes with built-in reporting features. It comes with a customizable API that can create event-based reporting workflows and data visualizations using external tools.

Help improve your team's productivity without micromanaging your developers.

Free trial CTA? Any other CTA (eg: demo) that 7pace wants to use?