The “Big Six” Drivers of Project Productivity and Predictability in Agile Environments

Whitepaper

This is part of a series of abridged whitepapers intended as quick reference sources for busy managers interested in the subject matter and faced with limited time to absorb lengthy research documentation.

It is based on research undertaken by Plandek drawn from anonymised data observed across a range of clients – from small start ups to large corporates with large scale, distributed Agile teams.

Introduction

Purpose of this Paper

The analysis presented below is focused on the software development process and is particularly relevant for Scrum Agile environments.

It is designed to give delivery managers a practical guide to the KPIs & metrics that directly drive project productivity and hence are strong predictors of project velocity and delivery timing.

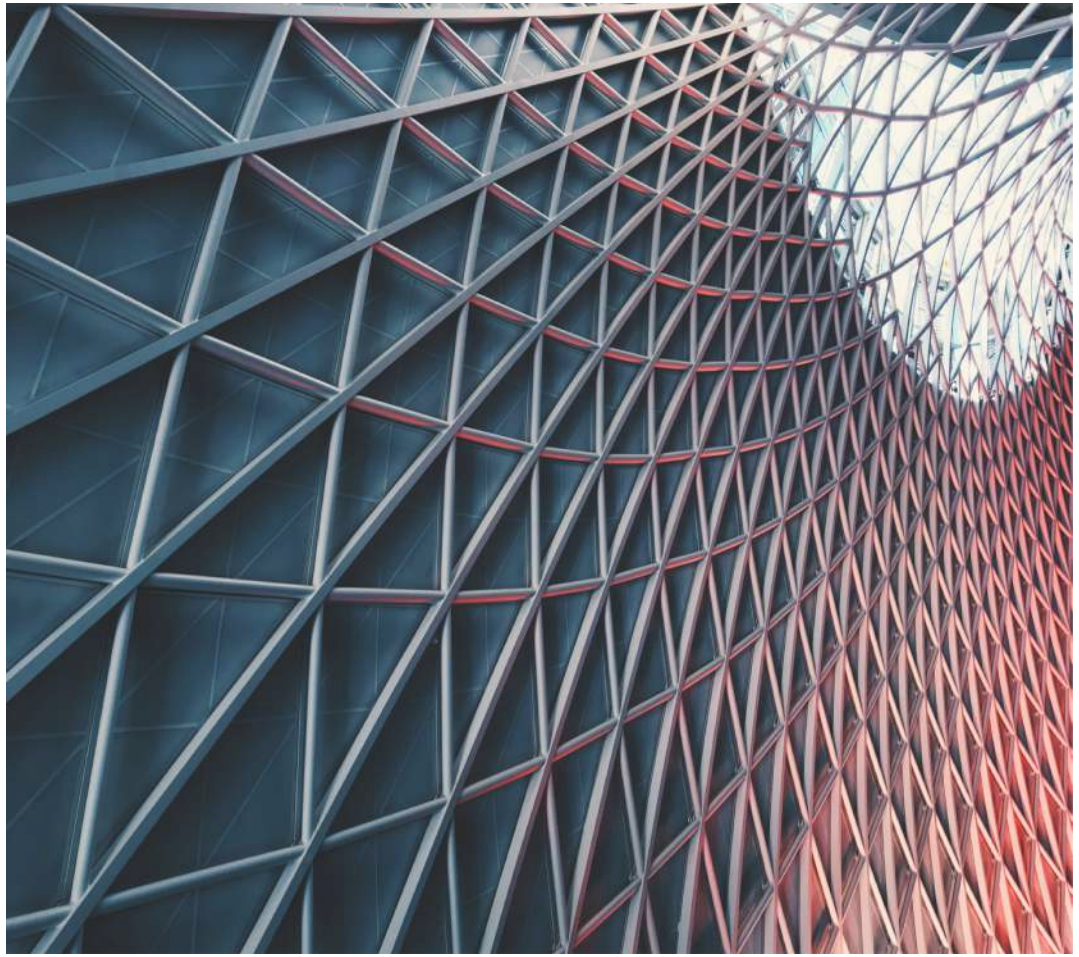
If actively managed, these metrics have been shown to demonstrably improve project productivity and delivery predictability.

Nature of the Analysis

The proprietary analysis is based on experience of working with clients and analysing anonymised data from strongly performing and poorly performing Agile software development projects observed between January 2017 and June 2018.

This paper is based on the findings of this research, undertaken in-house and in consultation with a number of clients, partners and interviewees principally based in the UK, consulted between September 2017 and June 2018.

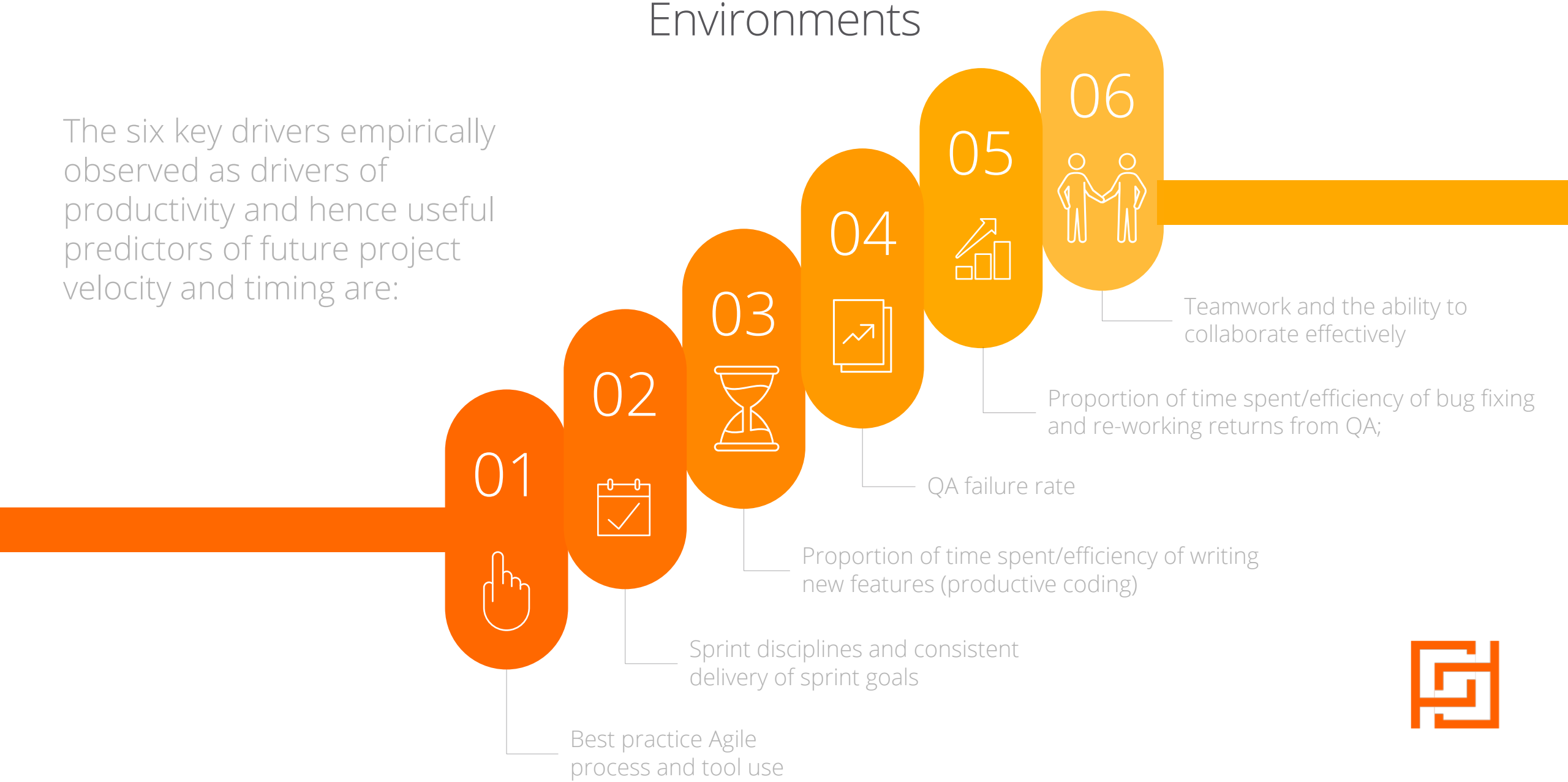
The Scope



- The scope of this Whitepaper is the software development stage of the process only, excluding pre-development and post-development (integration and deployment).
- The data collected within the Plandek tool are analysed to look for the underlying drivers of productivity and predictors of future problems (early warnings).
- Problem projects are defined as projects with rapidly declining velocity, late delivery and unplanned overspend.

The “Big Six” Drivers of Project Productivity and Predictability in Agile Environments

The six key drivers empirically observed as drivers of productivity and hence useful predictors of future project velocity and timing are:



The Critical Enabler – Best Practice Agile Process and Tool Use

- As Agile is based on a philosophy of continuous improvement, all companies are by definition “on an Agile journey towards engineering excellence”. Quite rightly, no client claims to be at the end of that journey. As a result, clients see big differences in process, workflows and behaviours across teams and projects.
- Many of these differences are healthy, reflecting the varied nature of projects and teams. However, there are certain disciplines that if not followed mean that any sensible analysis of the development process across and within teams becomes impossible. It is vital therefore that these basic disciplines are in place, in order to give visibility across the Agile environment, so projects and processes can be effectively managed.
- The most important driver of project productivity and predictability is this first category – this is especially true of larger Agile environments which quickly become very complex to manage.

Key metrics within the category include:

01

“Speeding tickets”

The ability to track by team and individual within team those tickets for which ticket status is not updated correctly – this often involves “speeding tickets”, where tickets are clicked through statuses after the event. It is not uncommon for 90% of all tickets to be treated in this way. Speeding tickets mean that there is no possibility of really understanding the time taken for tickets to progress through the development process, when the work really happened and where the bottlenecks are to be found.

02

Ticket shepherding

The common occurrence where one member of the team updates all team members’ ticket status (often after the event). Again, this prevents any meaningful analysis of real workflows and visibility of the individual contributors.

03

Commits without a ticket reference

The poor practice of not including a (Jira) ticket reference in the commit message or branch name. This prevents the ability to really understand the impact of code effort relating to individual tasks.

04

Parallel work done outside the Sprint

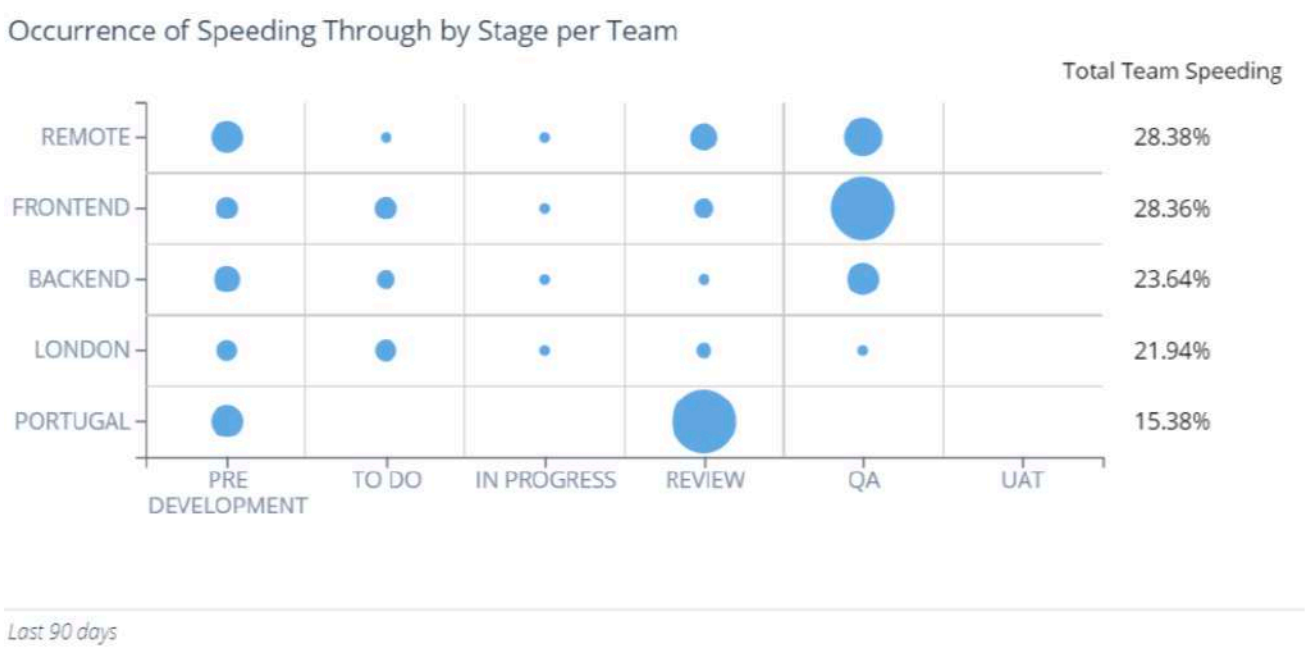
This is a common occurrence where teams are trying to run a disciplined scrum Agile methodology, but undertake work that is not within the agreed sprint targets (e.g. “undeclared” work left over from a past sprint or work from other stakeholders). Clearly if a large amount of work is being undertaken in this way, it is not possible to effectively predict sprint output and project velocity.

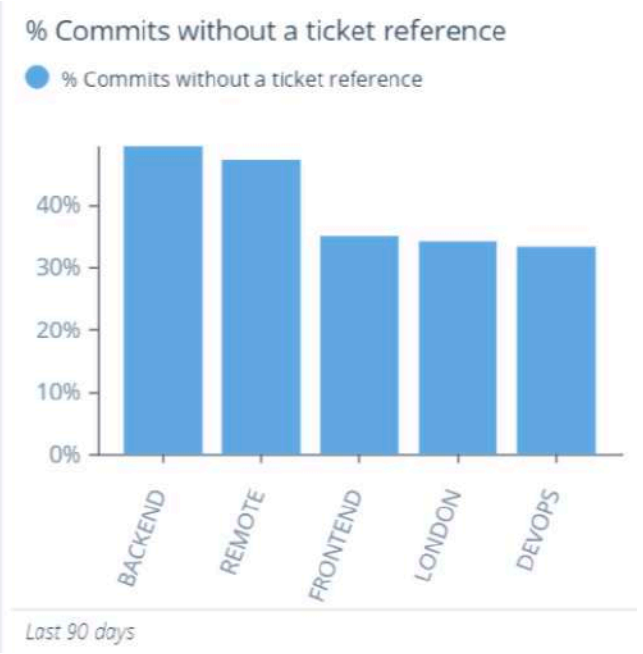
05

Proportion of code commits without a pull request

Effective code review before merging the code is a core discipline, but it may vary greatly by team and team type (e.g. outsourced teams). Code committed without a pull request is increasingly seen as an infosec breach as it should not be possible for individuals to unilaterally update the code base.

Example Agile process and tool use best practice analysis





Plandek, example Project Grace

Sprint Disciplines and Consistent Delivery of Sprint Goals (Scrum Agile)

- The empirical analysis across projects strongly supports the use of a scrum Agile methodology for the delivery of complex, time sensitive projects (with a critical go-live date). The practice of working in two-week sprints and getting into a rhythm of delivering accurately during that period generally increases the likelihood of on-time delivery at project end.
- Tracking teams' ability to work effectively within the structure that sprints provide and to consistently deliver their sprint goals, is the primary early-warning sign of future problems, being a very good measure of current productivity, teams' ability to estimate and plan tasks, and hence future velocity.

Key metrics within the category include:

01

Sprint Target Completion

This is a fundamental measure of sprint effectiveness and a definitive early warning of trouble to come if it deteriorates. Indeed, teams' ability to achieve on time what they commit to at the start of a sprint is absolutely vital if there is to be any visibility of future delivery at project level. It is best viewed therefore as a trend over time, by team. It measures the proportion of committed story points that are actually delivered within the sprint time period by team.

02

Average sprint delay

Looks at the time taken to complete any outstanding story points after sprint end. It is measured by team, in days. A two-day delay may not sound critical, but it represents a 20% time over-run on a ten working day sprint. Lengthening delays is a clear sign of existing velocity problems and is also an early warning sign therefore of future problems.

03

WIP queue size

Another potential sign of stress within teams. WIP queues will always exist, but are a sign of team stress when they breach a critical threshold and become unrecoverable without resorting to unplanned sprints to catch up. WIP queues tracked overtime (relative to the resource available within the team), can therefore be another good early warning sign of future velocity problems.

04

Ticket Overdues

These can be an interesting indicator of growing project problems. Teams can struggle under the pressures of overdues (as they impact each following sprint) and may need to insert additional sprints to clear them.

Proportion of time spent/efficiency of writing new features

- It is clear that the more time teams spend (efficiently) writing new features, the more productive they will be and the quicker a project will be delivered. Our empirical research shows that this simple metric is indeed a key determinant of project productivity and predictability.

- Indeed, it is a common occurrence to see teams with declining velocity due to an ever-increasing burden of non-productive time (e.g. fixing bugs, undertaking essential maintenance, crisis managing stakeholders etc).

Key metrics within the category include:

01

Proportion of time expended on feature contribution

A simple measure by team, tracked over time to reveal what proportion of time is spent on “writing new features”. This may well vary greatly between teams. Clearly timely project completion is dependent on the consistent health of this metric, it is therefore a key measure of productivity and predictability.

02

Analysis of bottlenecks within new feature development cycles

Regular analysis of the key stages within the cycle (from Jira statuses for example), shows the key bottlenecks and if these are increasing the overall cycle time, typical bottlenecks visible include:

Pre-development “On-hold” Time – where tickets are held waiting to be actioned as a result of input required (and not forthcoming) from a stakeholder or engineer

Development Cycle “On-hold” Time – where tickets are frozen pending a dependency within the team or on another feature

QA delays - potentially as a result of environment setup or resource issues within QA.

03

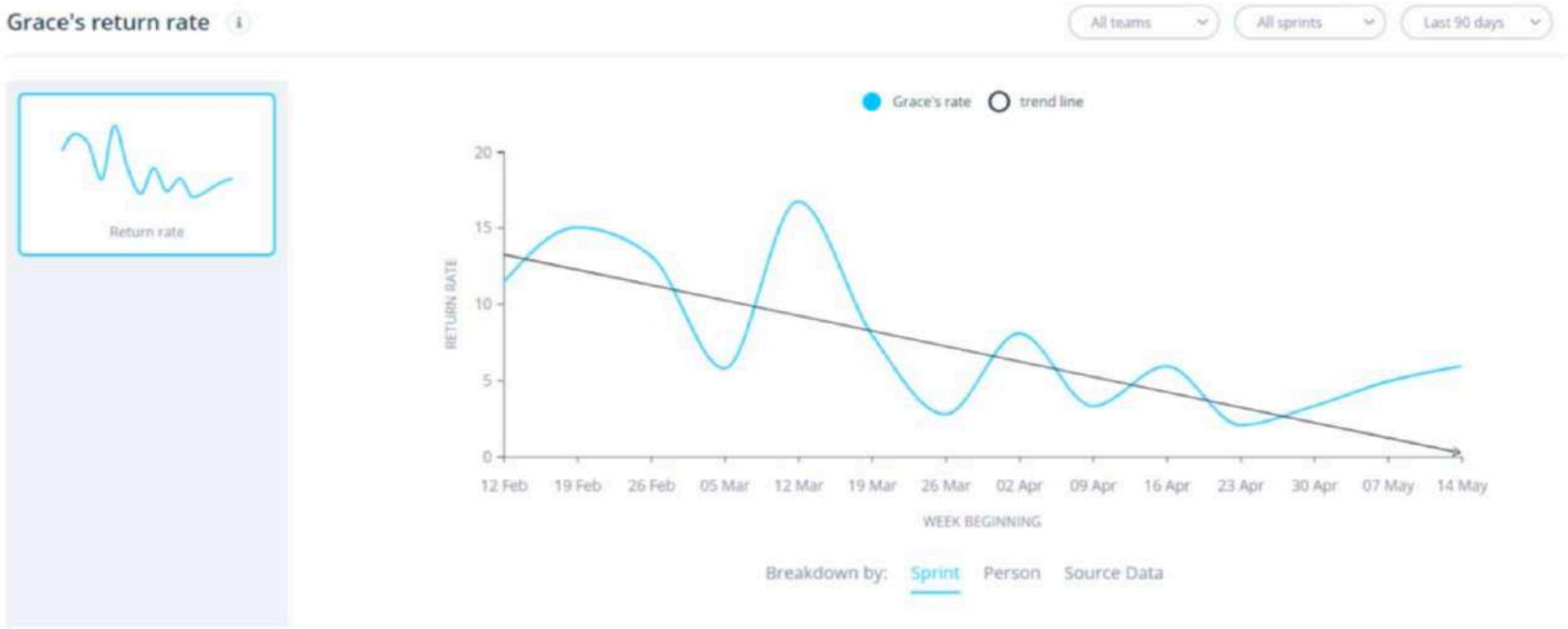
Lengthening cycle times within new feature development

Cycle times in both delivery of new features and upkeep vary for many different reasons. They may lengthen as teams tackle complex project areas - but teams which show a persistent trend of lengthening cycle times require investigation and potential early intervention by PMs.

QA failure (return) rate

- Too few development teams take time to really quantify and understand the impact of rising return rates on overall velocity. We define a “return” as any ticket that is returned to the development team following review by QA or UAT (for whatever reason). Returned tickets need reworking and therefore cause friction and inefficiency. High return rates maybe symptomatic of the brightest engineers working on the most complex of coding issues, but it can also be indicative of problems within a development team
- Plandek’s research has shown a clearly observable 80:20 rule – with on average 80% of returns accounted for by c20% of engineers. These data need to be reviewed judiciously and in context of the teams and projects in question. However, on deeper analysis it is often true that the data reveals engineers who may be new to the team or code base, who would benefit greatly from mentoring and collaboration with peers.

QA failure (return) rate



In this example, the return rate is declining (improving), showing a likely healthy team environment. The graphic below shows an analysis of individual return rates, used to guide Team Leaders' dialogue with individual engineers and to ensure that support and guidance is given to those engineers that require it.

Example Agile process and tool use best practice analysis

Breakdown by: Sprint Person Source Data

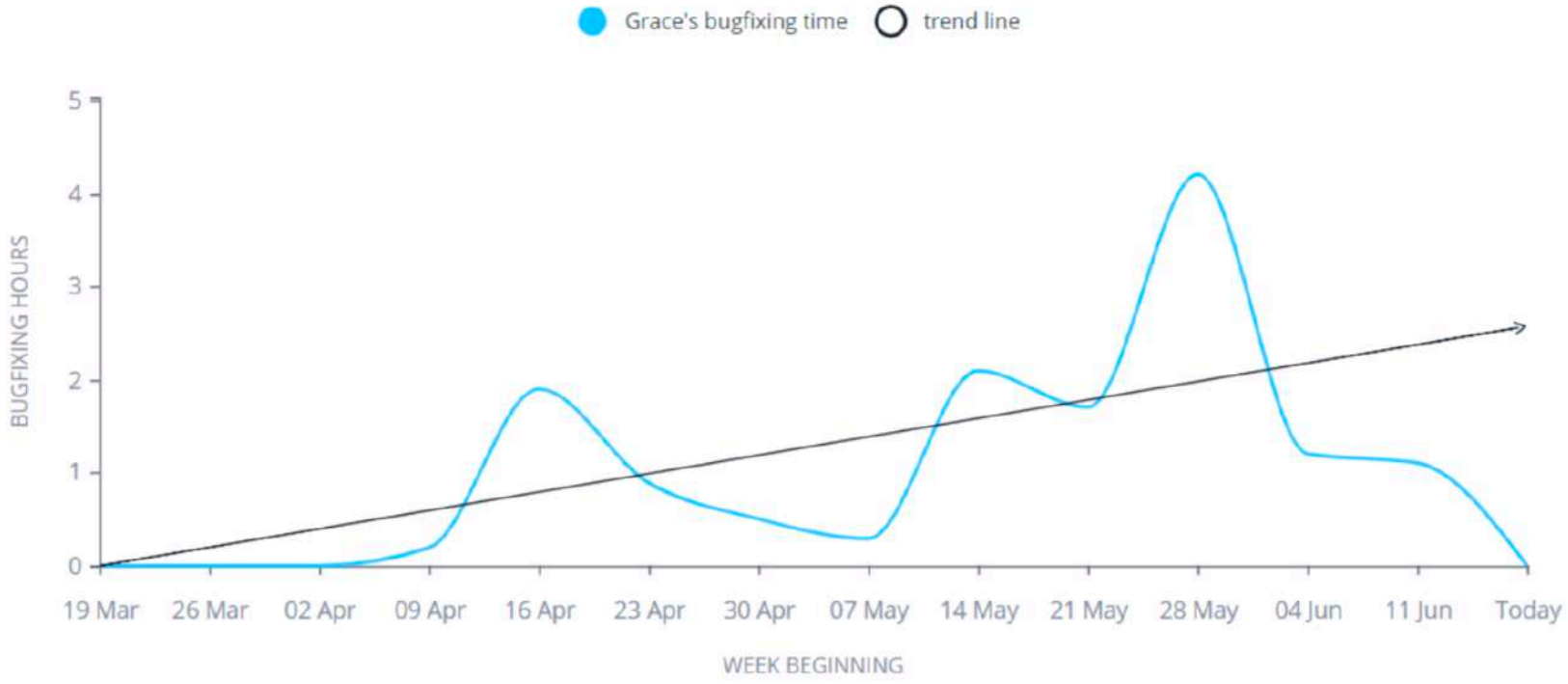
Name	Return Rate (%)	Number of Returns	Tickets Worked On	
Grace's return rate	*8.5*	*34*	*398*	
RM Ricky Martines DEVELOPER IN GRACE	28.6	2	7	
AD Amanda Davis DEVELOPER IN GRACE + 2 OTHER ROLES	22.7	5	22	
ML Marco Lanscioni DEVELOPER IN GRACE	20.4	11	54	
TF Teodor Ferra DEVELOPER IN GRACE	20	1	5	
HZ Hu Zheng DEVELOPER IN ADA + 2 OTHER ROLES	14.3	1	7	

Plandek, example Project Grace

Proportion of time spent/efficiency of bug fixing and re-working returns from QA

- This category of metrics is directly correlated to Category 4 (QA failure and return rates). It is common sense that the more time expended ‘fixing stuff’, will mean less time available to write new features and progress the project. Hence it is a vitally important set of metrics to track. They will vary over time and between team - and directly impacts project velocity.
- The two key groups of metrics in this category relate to:
  - time spent bug fixing;
  - time spent reworking tickets returned from QA during the sprint.
- If teams use a time tracker, it is easy to see the proportion of time expended bug fixing. Even without the use of a time tracker, it can be calculated by analysing bug fix cycle times.

Proportion of time spent/efficiency of bug fixing and re-working returns from QA



Plandek, example Project Grace

Teamwork and the ability to collaborate effectively

- Our research shows this to be a vital driver of team (and hence project) productivity. Good teams naturally work very closely together, supporting those members of the team who need help to improve their productivity (e.g. team members unfamiliar with the code base/technology/project).
- However, in many teams, teamwork does not operate as effectively as it could. The three most common reasons are:
  - unstable teams, with higher team member turnover,
  - inexperienced and introverted Team Leaders
  - teams under pressure with a lack of perceived time to help others.

Teamwork and the ability to collaborate effectively

- For these reasons, we believe that the consistent and objective measurement of teamwork is vital. It is particularly true in the light of the fact that empirical evidence shows that 80% of returns from QA are likely to be created by c20% of engineers (see category 4). These c20% of engineers should be mentored and assisted in a structured and consistent way.
- Indeed, our research has shown that very often the appraisal and development of engineers tends to be informal, inconsistent and often ineffective, as a result of:
  - a lack of ability to consistently measure certain important team behaviours;
  - a lack of meaningful HR processes (often unlike other departments within the same company);
  - inexperienced Team Leaders, who may have been promoted into the role with limited people management experience, skills and training.

Key metrics within the category include:

01

Comments in pull requests

To see who within teams are providing feedback in peer reviews and to encourage senior members of the team to get more involved if necessary

02

Collaborative lines

Tracked by individual and over time, to see how often the same code area is worked on by multiple people, which has been shown to be healthy to maintain code quality, to share code knowledge within teams and to reduce risk

03

Helping out and Pair Programming

These two metrics require very little effort from the engineers. All that's needed to register those is a short comment in Jira, like: "I helped John". Similarly, pairing on a particular task shows a different side of the contributions that an individual makes to the team and project and is also tracked via a short comment in Jira.

Teamwork and the ability to collaborate effectively

Breakdown by: Sprint Person Source Data

Grace's helping out	*6*	*6*
Name	Help given	Help Received
ML Marco Lanscioni DEVELOPER IN GRACE	5	
RS Rory Sackville TECHNICAL LEAD IN GRACE	1	
CE Callum Eaves DEVELOPER IN GRACE		4
Drake Newman DEVELOPER IN GRACE		1
TF Teodor Ferra DEVELOPER IN GRACE		1

Plandek, example Project Grace

Conclusions

- We have been struck by a growing concern among CTOs that the move to Agile among large engineering teams is not consistently delivering the promised productivity improvements and can make delivery forecasting less predictable.
- In our view, the promised benefits are there to be realised, if the Big Six levers of productivity and predictability are strongly and consistently managed at team and sprint level. Too often however, they are not regularly scrutinised by delivery managers.
- The empirical evidence shows that if the Big Six are managed in this way, productivity should immediately improve; and early mitigation of the early-warning signs will greatly reduce the likelihood of a project becoming a ‘problem project’.

What we do

Plandek is the leading Agile and delivery metrics BI platform, providing an end-to-end view of your software delivery cycle.

Our SaaS solution allows mining the data history from the toolsets that engineers use for actionable insights.

For more content, drop us a message at

hello@plandek.com or visit our website plandek.com

Reed Elsevier worldpay News Corp autoTRADER.ca @preqin secret escapes

Follow Plandek

facebook.com/plandekuk

@_Plandek_

www.linkedin.com/company/plandek

Plandek.com