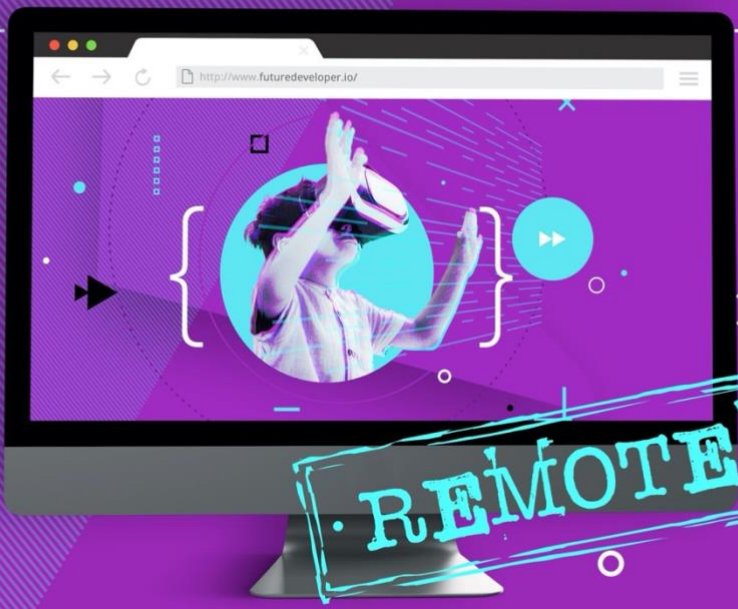
/DATA



BROUGHT TO YOU BY THE

20 future_ developer </summit>

BRINGING TOGETHER INDUSTRY THOUGHT LEADERS & DEVREL COMMUNITY

29 - 30 SEP
COMMUNITY EDITION

6 - 7 OCT
EXCLUSIVE EDITION

ACCESS FROM **ANYWHERE**

www.futuredeveloper.io

WHY HAVE WE WRITTEN THIS BOOK?

Welcome to the third edition of "Developer Marketing and Relations: The Essential Guide". The history of this book goes back to October 2017, during the Future Developer Summit. There, Andreas Constantinou and Nicolas Sauvage fully recognized the fragmented nature of developer relations or DevRel – from the types of companies, the products they represented, and the knowledge of the practitioners. It was there we witnessed that the best practices were often locked behind the doors of the companies that mastered them. We knew we wanted to work with these leaders and develop an essential guide to share this knowledge with a broader audience of developer relations, evangelists and advocates, developer marketing practitioners and beyond.

As we have watched the practice of DevRel grow and evolve over the last three years, there is a continued need for education of what DevRel is, along with the strategy and tactics needed to run a successful program. The good news is, many of the leading practitioners from the best companies agreed to share their knowledge, stories, learnings, and best practices in this guide! We think you'll find the information insightful, whether you are a seasoned professional in developer relations or you are just getting started.

A question we often get asked at SlashData is: "Can you help us understand how Mozilla, Google, or Microsoft practice developer marketing?" (replace names with your favorite tech brands). Well that's exactly what this book aims to accomplish.

This third edition includes the original chapters from the first edition, 9 new chapters and one updated. This guide is arranged in an order that takes you from strategic issues to more tactical issues. You can read from start to finish, or jump into a particular chapter that focuses on what you need to know right now. At a strategic level, you may want to read "Using Developer Personas to Stay Customer-Obsessed" from Cliff Simpkins of Microsoft, or if you are building out a program you might try "Structuring Developer Relations", by Dirk Primbs of Google. If you are just starting out, be sure to read, "Starting from Scratch: How to Build a Developer Marketing Program", by Luke Kilpatrick of Nutanix. If you need to get many stakeholders together in a large

organization, the “The Developer Relations Council: Leading and Aligning Developer Marketing within Large Companies” by Arabella David of Salesforce - a new chapter for the third edition- is a must. Then, learn how to understand numbers and KPIs in our new chapter “Measuring the success of a developer communications strategy” by our very own Rich Muir of SlashData.

We have a few chapters on events which have become a staple in the industry, such as “Growing Up By Scaling Down: How A Small Developer Event Can Make Big Impacts On Your Ecosystem”, co-authored by Neil Mansilla of Atlassian, and “Behind the Scenes Of Great Developer Events”, by Katherine Miller of Google. We must note that this edition is being launched during the COVID-19 pandemic, which has made the planners behind many programs re-evaluate their tactics, so you may want to read “How To Connect With Developers When You Can’t Meet Them”, by Pablo Fraile and Rex St. John of Arm.

As mentioned, developer programs exist in many types, as different companies are marketing different types of products to developers. Ana Schafer and Christine Jorgensen of Qualcomm describe their experiences with communities around hardware in “Hardware Is the New Software - Building A Developer Community Around A Chip Instead Of An SDK”. APIs are well known as a key product in DevRel so we are pleased to bring you a new chapter by Mehdi Medjaoui, founder of GDPR.DEV and APIdays conferences, on “Developer Relations and APIs”.

We can’t list all of the great chapters here, but we would be remiss if we didn’t point out the chapters on community, the heart and soul of any leading developer relations program. Be sure to read “The Power Of Community” by Jacob Lehrbaum of Salesforce, and the new chapter “Building an Inclusive Developer Community” by Leandro Margulis, based on his award-winning days at TomTom.

We always love to hear your comments – perhaps you read something that made a difference for your program, or you have your own story to tell. Please reach out to us at sdata.me/dev-marketing-book and tell us about it

All profits from this book’s sales will be donated to coding organisations that encourage and assist the developers of tomorrow. Your support is much appreciated in helping out these

worthy causes – more details can be found at sdata.me/dev-marketing-guide.

Andreas Constantinou, Founder & CEO, SlashData
Nicolas Sauvage, President & Managing Director, TDK Ventures
Caroline Lewko and Dana Fujiwawa, Editors of the third edition, WIP

September 2020

All profits from book sales will be donated to coding charities.

Introduction | Measuring the success of a developer communications strategy

Richard Muir: Data Journalist - SlashData

In the previous edition of this book, we introduced the idea of the developer as a decision-maker. The developer as an influencer. The developer as a powerful agent within an influential team, steering important technological decisions and being personally invested in the outcome.

With this in mind, successful communication is crucial, especially communicating with them in their language and through the channels they use. With developers being such a diverse (and sometimes challenging group with which to engage), the stakes have never been higher for getting your communications right. Here we'll help you to decide what to measure, and how to define the success of your team and your business.

All profits from book sales will be donated to coding charities.

How do you measure success and why does it matter?

Tl;dr: Measuring success lets you:

- Understand what works, so you can do more of it
- Communicate successes and failures
- Justify your decisions
- Allocate resources effectively
- Track your progress
- Time travel (no, seriously)

Before we get into the how, let's dip our toes into the why. Why does measuring success matter? Here at SlashData, we believe that data beats opinions. Every time. The short answer, therefore, is that you should measure success because it's the only way of understanding the truth of a situation. Measuring the success of your developer relations program is the only way to know if what you're doing is working. Not only that, measuring the results helps you to understand what works well, so you can do more of it, and what works not so well, so you can do less of it.

Another way of looking at this is that measuring the impact of your efforts is the best way to allocate resources efficiently, and for maximum effectiveness. For example, we see time after time that developers want technology companies to spend more time and effort answering questions on forums, and less on organizing conferences. Getting answers in public forums is the fourth most important feature for developers, yet it receives a satisfaction score ten points lower than conferences, the 18th most important feature. We know this because we ask developers which developer program features are most important to them, and how satisfied they are with the support that technology companies provide. At this macro level, measuring success (satisfaction, in this case) shines a light on how resources are allocated ineffectively across the industry. Now imagine taking this data-driven approach and applying it to your internal processes. What might you find? What decisions might you take? And how would you justify them?

When you use data to make these decisions, the justification becomes easy. It's right there in front of you. Why should you divert resources away from organizing conferences, and towards answering developers' questions? Easy, because that's what developers want, and they're currently not very happy with what

they're getting. So measuring success lets you justify your decision. Seen another way, if you can't back up your decision with data, are you sure it's the right move?

But sometimes even the most data-driven decisions don't turn out how you expected - the ground shifts under your feet or a competitor makes an unexpected move, or maybe a global pandemic shakes things up a little. That's OK, these things happen. But by using data to drive your decisions, and measuring the outcome, you can track your progress. You can communicate your successes and failures. Most importantly, you can *quantify* the impact of any changes. Not only changes that you effect, but also changes imposed upon you. Because here's the thing: data begets more data. The next time you sense trouble brewing on the horizon, you can look back in time through your data and see what happened the last time the ground moved under you. You can anticipate the scale of any impact, and take steps accordingly, influencing the outcome. Trust us, your boss will thank you.

In short, measuring success lets you decide what, exactly, you should be doing, and how you should be doing it. It also lets you make informed decisions and facilitates meaningful communication with your team and with stakeholders.

Deciding what to measure

Hopefully, I've covered the why, and hopefully, you're convinced enough to carry on reading to find out the how. But you'll have to wait a little longer for that. Because before knowing how to measure success, you need to know what you're going to measure. And before you even start to think about what to measure, you need to understand what you want to change.

Because that's what developer relations and marketing is all about. It's about creating and effecting the change that works best for you and your developer community. The world is going to change anyway, so why not get a hand on the steering wheel?

In order to know what changes you want to make, you need to understand what your business goals are. Fundamentally, any commercial developer relations program lives and dies by its ability to increase usage. To increase usage by developers already engaged with the platform, or to attract new developers to the platform. But we need to be more specific; of *course* the goal is to increase usage - that's the direct (or often indirect) path

to putting money in the bank. But what is it that you specifically want to achieve? Do you want to increase the frequency of usage for a specific product or API? Do you want to inform developers about an imminent major version and a potential breaking change? (Preventing a decrease in usage is often easier than driving an increase). Or do you simply want your developers to know that you're thinking of them?

Now, let's zoom right in onto the developer. What do you want them to do? What behavior do you want them to start? To stop? To change? Fundamentally, what is the purpose of your developer relations program? Nobody cares for meaningless marketing, least of all busy software developers. So let's get the message clear, and consistent, and make sure it's relevant.

In order to know the message, you must first understand your audience. Are you marketing to existing customers, or are you running an acquisition campaign? If you're marketing to developers that already use your product, it's likely that you already have loads of information about them. You'll have usage patterns, an email address, hopefully, some geodemographics, and maybe even a company name. All of this data helps to build your picture of what that developer does, what you want them to do, how you talk to them, and how successful you can expect to be.

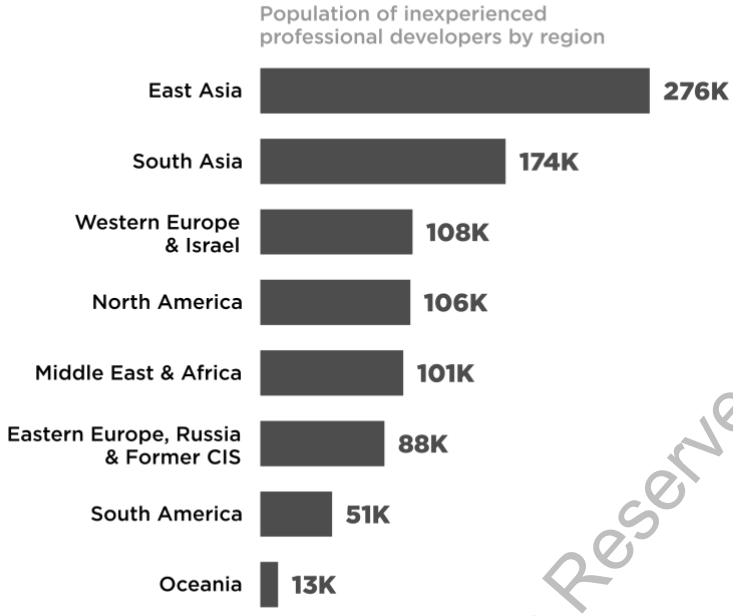
But what about an acquisition campaign? Maybe you're using Facebook audiences or advertising on more specialist platforms. Well, first of all, you want to know how many people you can reasonably expect to reach. This data helps you answer that all-important question, even before you begin. Is it worth it?

Is it worth it? Using data to inform and appraise strategy

With our developer population calculator (sdata.me/Calculator), you can figure out just how many developers fit your target audience and where you can find them. Looking for inexperienced professional developers? Then look no further. Of the 917K professional developers with less than a year's experience, East Asia has 276K of them.

East Asia has the largest population of inexperienced professionals

Q4 2019 (n=775)



Source: *Global Developer Population Calculator www.developerpopulation.com | © SlashData 2020*

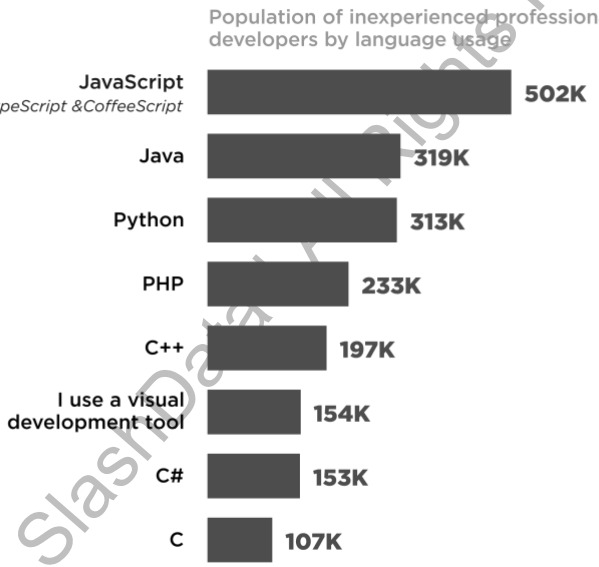
Developing a regional strategy is one surefire way of getting better at communicating with developers on their terms. We find that not only do the socioeconomic factors in different regions affect developers' immediate views and desires, but also that these differences echo at a macro level, shaping the way technologies, and even vendors, are adopted in different regions. For example, developers in Asian regions are consistently less-satisfied with having access to resources in their native language; unsurprisingly, given the anglocentric nature of technological change to date. But this also means that developers in these regions are quicker to take up emerging technologies as they are less burdened by convention and existing structures. So in this way, understanding your developer audience through a regional lens would not only help shape your short term tactical approach (translate resources aimed at inexperienced professionals into popular languages in East Asia,

say) but would also contribute to your longer-term strategy by adding another perspective to your expectations of your product's adoption rate.

One other important differentiating factor for developers is what language they use. Understanding language usage patterns amongst your developer audience will also help you to optimise your short-term decision making, with an eye on the longer-term strategy. Over half of our group of 910K inexperienced professional developers are using JavaScript with a third using Java or Python. This is interesting enough, but when compared with experienced professionals (of which there are 13.6M), we see that developers collect languages as their career progresses. Two-thirds of these more experienced professionals are using JavaScript and around 40% are using Java or Python. It's not surprising that developers are learning constantly, but armed with the information that JavaScript, Python and Java are used widely amongst early-career *and* experienced developers (for now, at least!), you can safely invest in supporting these language communities, knowing that your effort is unlikely to be wasted.

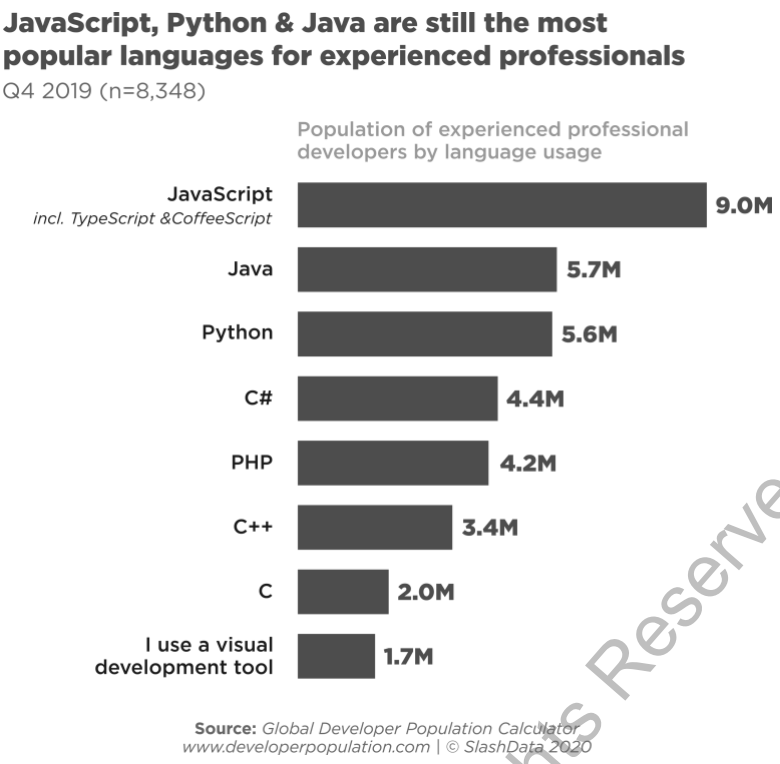
Over half of inexperienced professionals use JavaScript

Q4 2019 (n=509)



Source: *Global Developer Population Calculator*
www.developerpopulation.com | © SlashData 2020

All profits from book sales will be donated to coding charities.



If you've created a regional strategy and are segmenting developers by language, you're doing much better than most. These two data points are easy to collect, easy to interpret, and very powerful indeed. But there are as many ways of segmenting your developer community as there are developers, and you can use many different data points to create your developer personas (though please don't segment by technology! - https://sdata.me/segment). Creating developer personas is another common, and powerful, way of gaining a deeper understanding into developers' motivations and needs, and, just as importantly, of sizing your potential audience.

Knowing how many developers there are helps you to understand your penetration into a particular market. Will this developer marketing campaign be looking to get those difficult, marginal gains in a territory where you have already captured a

lot of the market, or are you entering a market with lots of headroom to grow? Knowing the size of your target audience helps you to calibrate your expectations for your developer marketing efforts. This information is especially powerful when you combine it with first-hand experience on the success of your efforts (here we're referring to click-through rates, signups, downloads and page views), and moves us nicely onto the next stage of this approach, actually choosing which KPIs to track, and who will be interested in them.

Knowing your audience

Developer marketing is a cross-disciplinary subject with many stakeholders. On one side, you have the marketeers, those responsible for interpreting the developer personas or segments (you do segment your audience, right?), and creating the messaging and branding that will draw these developers in. On another side, you have the developer advocate. These are the people who represent the developers' point of view, because they often are one. They understand how developers want to be talked to and they have an ear to the ground for direct, qualitative feedback from developers. Next, you have product managers. At the core of any developer relations program is a product, be it an API, a managed service, an IDE, or a framework. One of the most surefire ways of ensuring that developers use your product is to listen to them and act on that feedback. Finally, you have the top-level managers and executives who take a more holistic approach, looking at success from several perspectives.

Each of these groups is going to want a slightly different view of what developers do and think. Your marketeers want to know how well developers engage with the communications, messaging and campaigns. Your developer advocates might be trying to understand the community dynamics or particular pain points that developers face. And your product managers want to know which features are being used, and what developers are requesting next. In the next section, we'll talk about what KPIs we use at SlashData to give a nuanced but powerful view of the success of a developer relations program that can be applied to all these use cases and more.

Don't forget the developers, either. Although they might not see the results, they're still important stakeholders in the success of your developer relations efforts, maybe the most important.

Whichever metrics you choose will impact them more than any other group. So choose KPIs that benefit the developer - KPIs that value them like the asset they are.

Choosing a set of KPIs

OK, so you've got an idea of the art of the possible. Now it's time to look at what you can actually do with this knowledge. Here's where you decide which KPIs you should use to track success, and this happens to be our particular area of expertise. At SlashData, we measure success in developer relations in all kinds of different ways. We measure developer satisfaction with resources or with products; we measure engagement, adoption and rejection. We understand what developers think is lacking, and what they love. Most importantly, we back up our decisions with hard science and contextualise them with experience and expertise.

We use four metrics to give a 360-degree view of the performance of a developer relations program. We selected these metrics not only because they cover many different aspects of measuring the success of developers relations, and do so with very little overlap. In short, they are efficient and effective:

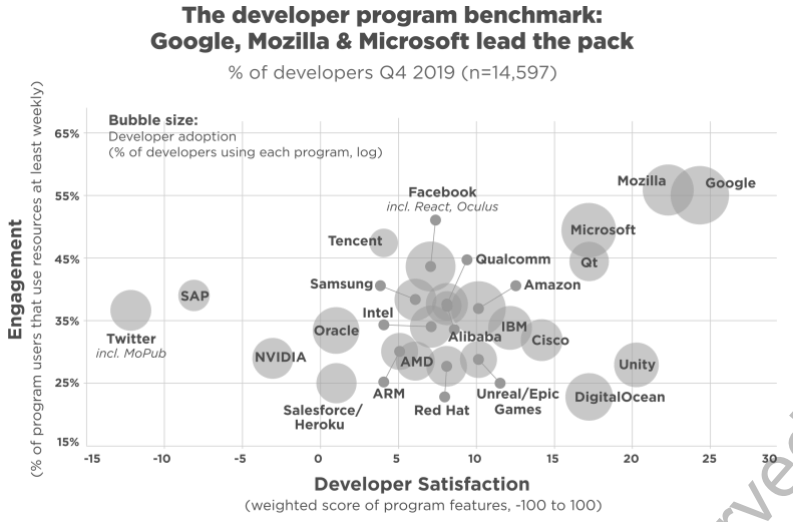
Adoption rate - How many developers use a product or vendor

Engagement rate - How often developers use a vendor's resources

Satisfaction score - What developers think about a product or its features

Reasons for adoption or rejection - Why developers chose (or rejected) a product

These metrics work together to give a nuanced view of success that's not only relevant for people with different interests (such as marketeers or product managers) but is also flexible enough to measure the impact of many different changes to a developer relations program. Here's an example from our Developer Program Benchmarking report that shows how the top developer programs in the world compare against each other for adoption, engagement and satisfaction:



Source: *Developer Economics | State of the Developer Nation Q4 2019 | © SlashData 2020*

You can clearly see how, when combined, these three metrics allow for very fine differentiation between otherwise similar developer programs. Take IBM and Intel, for example. When comparing these two vendors by engagement or adoption, they appear to be almost identical, however, the difference in satisfaction between them allows us to tease out this important difference. And when you can understand in fine detail the differences between two developer programs (or products, or regions etc.), then you can take targeted and meaningful action to change the situation to your benefit. These KPIs, therefore, allow for developer relations practitioners with different interests to take meaningful and effective actions at a variety of levels.

Are you an executive interested in growing the developer community in South Asia?

Consider looking at the changing adoption rates there. The wide-angle view will give you an insight into the cumulative impact of the different pillars of your developer relations strategy and efforts, while a deeper dive into the data will help you to hone this strategy and focus on what works for different segments. Because our metrics are calculated from the opinions of individual developers, we can look at the data from different

angles and understand who is using a product, what they think of it, and how you can reach these developers.

Are you a product manager looking to optimize your feature development budget?

Then you'll be especially interested in why developers reject or adopt a product. By looking at the adoption and rejection reasons developers give for your product *and* your competitors, you'll be better able to carve out your niche in the product space. You'll understand why developers choose your product and be able to take steps to protect this advantage, whilst at the same time capitalizing on your competitors' weaknesses.

Are you a marketer looking to understand the success of a recent email campaign?

Developers' engagement behavior will be especially interesting for you. Understanding which of your products have the most engaged developer community will shine a light on what you can hope to achieve, and tracking changing engagement rates will help you to understand what works best for the different segments of your community.

Are you a developer advocate who wants to target improvements to a product's documentation?

Documentation satisfaction scores will help you to understand where the gaps are and point towards those vendors leading the way. You'll be able to zoom in on users of a particular product or language, or developers in a particular region, and make micro-improvements to squeeze out those last marginal gains.

Our wealth of experience in measuring the success of developer relations has led us to create this framework for selecting metrics that will help you to effect positive change. The KPIs we use were born from this framework. When you're choosing a suite of KPIs (and we do recommend selecting a few) to measure the success of your developer relations program, here are a few things to keep in mind:

Be realistic - this is where the groundwork of understanding the market, and your performance within it, helps you to select SMART objectives, with the emphasis here on 'Achievable' - what can you actually influence.

Be considerate - and by that I mean, choose KPIs that are easy to understand. Be considerate to your colleagues by not hiding KPIs behind definitions and acronyms.

Know your data - you want to choose KPIs that are sensitive to change, but not so sensitive that they fluctuate wildly. Choose KPIs that you can calculate with the data you have, and remember that you can always get someone else to help ;-)

Tell a story - remember, you're trying to change behaviour. You're trying to change not only developers' behaviour, but also your colleagues' behaviour. Telling a story is the most compelling way to bring someone round to your way of thinking. Choose KPIs which combine to tell a story and watch your team come together to make it come true.

Think about the ramifications - We already know we're trying to change behaviour, but what about any unexpected changes? Do the KPIs you've selected incentivise the right behaviour amongst your team, or have you unwittingly created perverse incentives for a negative behaviour change?

Let's wrap things up

So here you have it. A window into the SlashData way of selecting KPIs to measure the success of a developer relations program. You've seen why we think that data-driven measurement is so absolutely vital (and I hope you agree!). You've read a few ideas on how to decide what to measure, and how to think about your audience. Finally, you've been introduced to the KPIs that we use to measure the best performing developer programs in the world and you've also seen a framework that will help you to select your own. This is just the tip of the iceberg, you will discover more ways to boost your developer relations program in this book and you can always refer to devrelx.com for more free resources, including Under the Hood of Developer Marketing podcast, webinars, free graphs and more.

All profits from book sales will be donated to coding charities.