Developer Relations & Marketing

Developers have always been an important constituency for technology companies, but they are becoming even more important as key stakeholders and customers for software vendors, cloud service providers and even 'Internet of Things' companies. This report provides the starting point for understanding how to craft a program that addresses developers and ensures that developers help to create thriving ecosystems around your offerings.

KEY FINDINGS

- Developers have played a large part in driving the success of companies like Microsoft and SAP, consumer websites like Facebook, and service providers like Amazon Web Services and DigitalOcean. These companies' offerings can be traditional ISV software, SaaS, infrastructure delivered as a cloud service, or even 'Internet of Things' devices.
- While developers may emote a distaste for traditional marketing, they are actually very responsive to finely tuned developer relations and marketing programs. These programs revolve around understanding how developers fit into the offering's strategies and plans, creating value propositions that appeal to the needs and desires of various types of developers, and ensuring that you 'show up' to relevant developer communities.
- There are many developer communities, both on the Web and in real life; most of them will be third-party sites that are not under your control. However, as developer relations programs grow, gaining more budget and resources, building your own developer community may be beneficial.
- One of the strongest clusters of developer value propositions centers on speed and increasing productivity: getting software to production faster, ensuring very low barriers to entry to try out new technologies, and otherwise simply moving faster.
- 'Content' (text, video, audio, documentation and even source code) is one of the primary mediums for interacting with developers, and, depending on the type of content, it has many different applications in developer relations programs.

ABOUT 451 RESEARCH

451 Research is a preeminent information technology research and advisory company. With a core focus on technology innovation and market disruption, we provide essential insight for leaders of the digital economy. More than 100 analysts and consultants deliver that insight via syndicated research, advisory services and live events to over 1,000 client organizations in North America, Europe and around the world. Founded in 2000 and headquartered in New York, 451 Research is a division of The 451 Group.

451 Research®

New York
20 West 37th Street, 6th Floor
New York, NY 10018
Phone: 212.505.3030
Fax: 212.505.2630

London
Paxton House (5th floor), 30 Artillery Lane
London, E1 7LS, UK
Phone: +44 (0) 207 426 0219
Fax: +44 (0) 207 426 4698

San Francisco
140 Geary Street, 9th Floor
San Francisco, CA 94108
Phone: 415.989.1555
Fax: 415.989.1558

Boston
125 Broad Street, 4th Floor
Boston, MA 02109
Phone: 617.275.8818
Fax: 617.261.0688

TABLE OF CONTENTS

SECTION 1
Executive Summary

1.1 INTRODUCTION

The ability for software developers to make or break a product or service has grown significantly in recent years. They have always been an important constituency, as organizations like Microsoft and Mozilla have demonstrated, but because of the rise of digital enterprises, cloud technologies that ease infrastructure needs, and new form factors like mobile and tablets, developers are now often included in the critical paths for success in many business models.

These businesses may be 'traditional' technology companies selling software, cloud providers offering SaaS or more infrastructure-oriented services like PaaS and IaaS (or service providers looking to become cloud providers), and even non-technology companies that are looking to leverage the growing 'Internet of Things,' where previously non-networked or IT-fueled products like cars, appliances and even fashion (like watches) are the 'things' joining the Internet ecosystem.

Once developers become important to your business, you must actively groom an ongoing developer relations program that pulls them into your processes, just like you would any important stakeholder or customer. This report focuses on how to plan for a developer relations program and some of the fundamentals needed to start such a program.

1.2 KEY FINDINGS

- Developers have played a large part in driving the success of companies like Microsoft and SAP, consumer websites like Facebook, and service providers like Amazon Web Services and DigitalOcean. These companies' offerings can be traditional ISV software, SaaS, infrastructure delivered as a cloud service, or even 'Internet of Things' devices.
- While developers may emote a distaste for traditional marketing, they are actually very responsive to finely tuned developer relations and marketing programs. These programs revolve around understanding how developers fit into the offering's strategies and plans, creating value propositions that appeal to the needs and desires of various types of developers, and ensuring that you 'show up' to relevant developer communities.
- There are many developer communities, both on the Web and in real life; most of them will be third-party sites that are not under your control. However, as developer relations programs grow, gaining more budget and resources, building your own developer community may be beneficial.
- One of the strongest clusters of developer value propositions centers on speed and increasing productivity: getting software to production faster, ensuring very low barriers to entry to try out new technologies, and otherwise simply moving faster.

- 'Content' (text, video, audio, documentation, and even source code) is one of the primary mediums for interacting with developers, and, depending on the type of content, it has many different applications in developer relations programs.
- There are many topics for content, but some of the more popular ones are educational materials about using your offering; 'war stories' about how you solved a difficult problem; the story of how you made something (aka 'sausage making'); commenting on industry trends and how your offering fits in; and case studies on how developers have used your offering. Documentation and source code are key types of content for developer relations programs.
- Documentation is a particularly important type of content – both 'manuals' and narrations or tutorials that are in a more natural voice. With rare exception, every developer relations program should look at documentation as a core type of content. Source code and the social networks associated with sites like GitHub are also an important type of content that most developer relations programs should have.
- Conferences continue to be an effective engagement point for developer relations communities. They can range from large, vendor-centric events to small 'unconferences.' Companies have reported to us that both types can be highly successful, depending on the tactics and conference.

1.3 METHODOLOGY

This report on developer relations and marketing is based on a series of in-depth interviews with a variety of stakeholders in the industry, including developers at end-user organizations across multiple sectors, technology vendors, managed service providers, telcos and VCs. This research was supplemented by additional primary research, including attendance at a number of trade shows and industry events and the author's participation in developer communities over the past 20 years.

Reports such as this one represent a holistic perspective on key emerging markets in the enterprise IT space. These markets evolve quickly, though, so 451 Research offers additional services that provide critical marketplace updates. These updated reports and perspectives are presented on a daily basis via the company's core intelligence service – 451 Market Insight. Forward-looking M&A analysis and perspectives on strategic acquisitions and the liquidity environment for technology companies are also updated regularly via 451 Market Insight, which is backed by the industry-leading 451 M&A KnowledgeBase.

Emerging technologies and markets are also covered in additional 451 channels, including Datacenter Technologies; Storage; Enterprise Platforms & Infrastructure Software; Networking; Information Security; Data Platforms & Analytics; Development, DevOps & Middleware; Social Business Applications; Service Providers; Cloud & IT Service Markets; European Services; Multi-Tenant Datacenters; Enterprise Mobility; and Mobile Telecom.

Beyond that, 451 Research has a robust set of quantitative insights covered in 451 products such as ChangeWave, TIP, Market Monitor, the M&A KnowledgeBase and the Datacenter KnowledgeBase.

All of these 451 services, which are accessible via the Web, provide critical and timely analysis specifically focused on the business of enterprise IT innovation.

This report was written by Michael Coté, Research Director, Infrastructure Software (overseeing the Enterprise Platforms and Infrastructure Software channel and the Development, DevOps and Middleware research channel), with assistance from Peter Christy, Research Director, Networks; Jay Lyman, Senior Analyst, Enterprise Software; and Scott Ottaway, Vice President, Services Research. Any questions about this report and its methodology should be addressed to Michael Coté at: *cote@451research.com*.

Michael Coté – Research Director, Infrastructure Software

Michael Coté heads up the Infrastructure Software practice at 451 Research, focusing on everything from the hypervisor to the application layer. In addition to covering application development, systems management, VDI, cloud management and platforms, he also closely follows the business of software – including product management and marketing; strategy and M&A; and the tectonic moves of vendors and communities in the technology world. Michael has worked at several technology companies, from startups – such as the pioneering online banking company FundsXpress – to BMC Software. In 2006 he chased his passion for following and commenting on the IT industry to RedMonk, where he was an industry analyst covering the IT world and helping clients plan how to most effectively prosper in the rapidly changing space. Prior to joining 451 Research, Michael worked at Dell in various strategy roles – in corporate strategy, where he helped launch the software group; heading up the internal incubation program; leading Dell's cross-company cloud strategy; and finally, running the portfolio strategy team for the Dell Software Group.

For more information about 451 Research, please go to: *www.451research.com*.

SECTION 2

Developers Are Driving Sales of Software and Cloud Services

Rarely is a computer purchased to simply be turned on: while blinking lights may be entertaining, especially on pine trees, computers require software to be useful. Developers are responsible for writing that software and, arguably, play a central role in much of the money spent on IT globally. Without developers writing software, computers, datacenters, mobile devices and 'the cloud' are all useless beyond serving as self-heating stacks of metal or footrests.

In the realm of enterprise software, developers did not always play such a key role in the sales process because of undue friction in trying out and using products. Simply getting ahold of, let alone testing, the software being considered was difficult and infamously would require a slew of consultants. The widespread adoption of Web applications and open source in the early 2000s quickly changed this, and the friction of doing PoC testing of various enterprise offerings has been virtually removed in the current day[1]... at least it should be. This has also caused and coincided with the rise in 'shadow IT,' wherein developers are often bypassing official IT department policy and working more directly with line of business (LoB) groups to get applications out the door quickly. The availability of cloud-based infrastructure and services is further bringing developers to the forefront and making them an even more important constituency in IT sales.

Cloud services from the likes of Amazon Web Services (AWS) and Rackspace make getting up and running extremely fast, playing to the desire of developers to get their applications into production as quickly as possible. Cloud service providers like these are not only enabling developers, but are also actively courting them with their own developer relations efforts to drive usage and product direction. For example, DigitalOcean has seen a rise in adoption, *growing at about 30% each month through much of 2013* after switching to SSD drives, which developers love.

SaaS companies like salesforce.com and consumer SaaS platforms like Facebook clearly depend on establishing strong communities of developers to extend their core SaaS platforms with new functionality, integrate with other services, and otherwise increase the value of the overall platform ecosystem by writing new code. While developer relations and marketing used to be important chiefly for technology companies that were primarily ISVs, service providers and cloud providers now must also directly address the developer market.

1. *See Geva Perry's "From Developer Adoption to Enterprise Dollars" presentation for more details on this timeline.*

Back in the software space, Atlassian (which is, granted, a developer-tools-centric business) provides a good example of success with a strong developer relations approach. The company has famously grown revenue with 'zero salespeople.' Of course there is a sales function at Atlassian, but by relying on astute developer marketing and product management, the company spends very little comparatively on sales and marketing. By properly marketing to developers, Atlassian, founded in 2002, grew revenue from $59m in FY2010 to $149m in FY2013 and is widely thought to be ready for an IPO soon.

These examples exclude the obvious major historic example of Microsoft and the Java world, which thrived on attracting and retaining developers to their platforms, as well as the modern-day examples of Apple, Google, Facebook and others in the Web space.

Clearly, when done correctly, creating and maintaining a developer relations and marketing program can be beneficial for business, whether at a technology company or in emerging spaces like the Internet of Things (IoT), where new and traditional companies alike are injecting software into highly networked everyday objects. Much of the work to be done here is in programming the new platforms and applications that will drive IoT devices and networks. Developers could soon be a driving force in refrigeration and toaster technology (finally!).

In our view, technology firms and many non-technology companies (those toaster makers, for example) must start considering how they work with developers. This report is targeted at companies – be they ISVs or SaaS providers, cloud and service providers, IoT companies, or other 'digital enterprises' – that are seeking to instrument as much as their organization as possible with software-driven business services, and thus will likely need to consider developers as new stakeholders and customers.

This report covers some of the critical steps and thinking behind putting together a developer relations and marketing program, including general guidelines for what a program looks like, as well as identifying who developers are exactly and examining a collection of tactics that can be applied. The report merely scratches the surface of developer relations, however; a thorough treatment would fill a book. Where relevant, we have made reference to other resources available for digging deeper. And, of course, 451 Research is always eager to help.

2.1 'RELATIONS' OR 'MARKETING?'

You'll often encounter two terms, which, in this report at least, we treat synonymously: 'developer relations' and 'developer marketing.' Both of them essentially mean the program and activities you pursue in order to curry favor with developers, get your message to them, actively include them in product management plans, or simply hang out with them in the commercial interests of your organization. You may also encounter terms like 'platform evangelism' or simply 'evangelist.'

While the rise of coupon and 'deal' sites like Groupon shows that sometimes people actually do want to be marketed to, most people, let alone developers, do not think of themselves as *enjoying* being the target of marketing. The term 'marketing' therefore gets a bad rap, and some try to avoid using it at all. Borrowing a term from open-source community management (and marketing!)[2], 'relations' is often used instead of 'marketing' to arrive at the softer phrase of 'developer relations.'

But from our perspective, whether it's called marketing or relations, it involves the same goals – and it can be great or poor depending on the execution. Throughout this report, then, we use both terms.

2. See our 2010 report, "Closing the Deal with Community" for more.

SECTION 3
What Is a Developer Relations Program?

The goal of a developer relations and marketing program is to establish your technology, service or product as the de facto standard in its market[3]. Users and buyers in that market may not end up always choosing your platform, but it will become 'the normal' choice. At the start, this means somehow getting developers to choose your technology over competitors (or in-housing their own solution), while ongoing it means maintaining and growing the de facto-standard position. Tactically, this means that the goals of a developer relations program are to encourage developers to consider, try out, advocate for, purchase, use, and even contribute to your product, project or service.

This is part advertising, informing developers, training them, and encouraging them to take a more active role in, for example, using your cloud services' APIs, building extensions on top of your ERP platform, or integrating with your newly networked IoT device. Some developer relations program include building partnerships as well: getting the developers at other companies to adopt, build on top of, and interoperate with your offerings.

With that in hand, the goal is then to convert that good will, knowledge and effort into sales – at least for commercial organizations. Non-commercial organizations, like open-source foundations, use developer relations programs to grow adoption of their projects and attract community members to work said projects and grow membership, paid and otherwise.

More advanced developer relations programs treat developers as an important part of the product management planning process, involving them (individually and in aggregate) in determining what functionality should be included in a platform. Some products require this deep level of interaction; for example, the developer relations around platforms like iOS and Android require directly soliciting input from developers on the future direction and features of those platforms and, when it makes sense, actually doing something with that input that is reflected in the platform.

3.1 CREATING YOUR DEVELOPER RELATIONS PROGRAM

Each developer program will be different depending on the products and services you're offering, the resources you have at your disposal, and the developers you're working with. However, there are some high-level guidelines that can direct the strategy and planning for your developer relations program:

1. **Be clear about your goals and your connection to the sales and product management processes** – Developers have become a key part of the funnel for many IT sales processes – not just a 'nice to have' part. Whether you're an individual marketing to developers (often called an 'evangelist' or *'advocate'*) or a dedicated team

3. A notion that should be credited to Jim Plamondon's 2008 unpublished manuscript on technical evangelism, and echoed in Patrick Chanezon's discussion of Google's developer relations program.

of developer marketers, you should be crystal clear on how your activities fit into and 'feed' the process of acquiring prospects, closing deals, retaining customers and growing accounts. Developers may be so strategic to your offering that you need to start intermeshing your developer relations and product management activities as well, involving developers in the shaping of the offering itself, and not just raising awareness of the offering among developers once it's out the door.

2. **Measure your progress and activities** – Use marketing activity input to constantly assess how you're doing and re-target your efforts. As covered in Section 5, there are many, often subjective, tactics for doing developer relations. Some can be highly measured, such as lead generation at conferences, but others are more difficult to gauge. In either case, finding out what's successful will be an iterative process, especially as you seek to more perfectly align with sales (see guideline #1), which can itself be chaotic and ever-changing.
3. **Find and engage with your target developers** – Much of the activity in a developer marketing program is good old-fashioned engagement and influence-building. As we cover in Section 5, you'll be seeking to generate content and conversations, to not only disseminate information about your platform, but also support the ongoing engagement (or '*conversation*,' if you prefer) with prospective and current customers in the shape of developers.

If successful, as your developer relations program grows, you will likely move from tasks that can be done part-time by individuals to a full-fledged, well-funded and well-staffed program. In Section 6.1, we profile SAP's developer relations program, which is large indeed – supporting over two million developers globally, running four annual conferences and supporting an SAP-hosted community, to name just a few of the activities this large, well-funded developer relations staff performs each year.

SECTION 4

What Is a 'Developer?'

Before discussing how to work with developers in your marketing efforts, it's good to know some general types of developers so that you can tailor your efforts accordingly. While there are many types of developers, all of them, of course, write software. They do this by using at least one programming language to write the code needed to fulfill some set of requirements. These requirements define what the software does – for example, managing the workflow for approving a purchase order, allowing users to share photos of their cats eating sandwiches, or categorizing email as spam.

In an organizational context, a developer's primary goal is write code that fulfills the requirements given to them, most likely with as few errors as possible and as close to an agreed-on delivery date as possible. Cost, while a consideration, is mostly left up to management. Like all people, developers have personal motivations as well: primarily, learning new programming skills, advancing in their careers, avoiding boredom and pride in their work.

4.1 DEVELOPER TYPES

Developers are usually defined by the tools they use and the context, or industry, in which their software is delivered and used. For example, there are 'Web developers' who write in HTML5, PHP, Ruby, etc. and whose applications are delivered on the Web. 'Enterprise developers' are not so much defined by the tools they use as much as by the industry where their software is delivered; it is generally for use by large organizations (corporations, governments, etc.) to fulfill business requirements, either internal or external-facing. 'Mobile developers' typically write apps to be delivered on iOS, Android and other platforms, using a variety of programming languages and tools, and may be writing 'consumer' or 'enterprise' software. Also, as we outlined in our earlier report, *The Rise of Polyglot Programming,* we've found that programmers are increasingly using a wider and wider variety of tools and languages.

While much of our focus is on newer types of developers, there are also developers who work on legacy platforms such as mainframes or mini-computers, and in older languages like COBOL, or even old instances of Java. These developers are typically found in huge financial companies and government organizations that have been using IT for decades and have found little (usually financial) motivation to modernize their IT. Finally, there are also 'systems' and 'embedded' developers who write software used in hardware ranging from PCs to cars.

The way developers organize themselves and run their process is also critical to defining a type of a developer. While it may be a cliché, there are many 'coffee shop and hoodie' developers that act as free agents, often alone, on project-based development work. This type of developer is often creating Web and mobile applications. As a team is formed, developers often must introduce process and methodology to ensure that collaboration is beneficial and doesn't damage the end product. Once a team of developers is formed, you can expect them to contemplate using a process like Scrum (a type of Agile software development), Lean or even emerging processes like *DevOps*. In massive organizations – and especially in government – where teams number in the hundreds, if not thousands, you can expect to see developers using derivations of Agile or older methodologies like the Capability Maturity Model (CMM), or non-iterative, 'big-bang' methodologies that are often collectively labeled 'waterfall.'

4.2 SUPPORTING ROLES

In groups, developers are typically joined by several other roles: architects, QA staff, product managers and project managers. The term 'architect' generally means a more senior person who tries to look at the big picture and make sure all of the parts fit together well for the present need, and also ensure that the software being developed will be 'future-proofed.' This notion of future-proofing can essentially be viewed as avoiding debt. Often called technical debt, architects (and hopefully the developers themselves!) try to ensure that the software being written remains flexible and well designed enough to adapt to future needs[4].

QA staff, or testers, are responsible for devising ways to verify that the software is functioning correctly (i.e., satisfying requirements) and also to find bugs and errors in the software. QA staff do this by coming up with tests and executing them, either manually or, increasingly over the past decade, with automated testing. As our recent DevOps Market Survey shows, QA teams are often one of the least influential groups in the development process, sadly.

Product managers are responsible for ensuring that the software being developed is valuable to customers and, thus, that the customers will want to buy it. They do this primarily by specifying requirements, helping developers understand the requirements (features) that go into each release of the product, and, in Agile methodologies at least, being an advocate for the customer's concerns.

Project managers – often a role filled by the managers of development teams – are responsible for keeping things on track and ensuring that the overall group is delivering on time and otherwise functioning properly.

4. See this 2012 interview with Agile expert, Israel Gat for further explanation of technical debt.

A good developer relations program will analyze how these roles are fulfilled in its target market and tune its tactics accordingly. In terms of the specific roles within the development team, sometimes more than one are fulfilled by the same person, or by the group as a whole. For example, much of automated testing is now done by developers as part of their primary development practice instead of afterwards by QA staff. Architecture is often not left up to an individual, but rather spread across (usually) senior members of the team.

In this report, when we say 'developers,' while not entirely accurate, we're often collectively referring to all of these roles – that is, individuals in the overall software development teams that are working together to create and bring their product to market.

SECTION 5

Tactics

Marketing to developers is a tricky business. Most developers believe themselves savvy enough to be immune to marketing. And, indeed, developers are savvy about identifying when they're being 'hustled.' While it's true that nearly every developer is a sucker for a free t-shirt, more sophisticated tactics than snazzy schwag are needed for a good developer relations program. The goal, of course, is to reach the influential developer early in the sales, or general relationship-building, process and ensure they're getting the information and support they need to reach a decision about your offering.

5.1 TOP-DOWN VS. BOTTOM-UP

When introducing technologies, there are two broad methods of getting new technologies into an organization:

1. **Top-down:** Appealing to upper management and architectural needs (still often individuals), who will then mandate the use of your product 'down' through the ranks of the company.
2. **Bottom-up:** Appealing to an individual's needs and desires for their primary projects or even side projects, and then allowing those developers to spread their decisions to use your product 'upward' through the organization.

Most of our tactics deal with the second approach, as does much of developer relations thinking. Historically, the introduction of new developer technologies in organizations started with a bottom-up approach, so we spend most of our attention on this method. However, as noted above, even in top-down decision-making, you will still be appealing to individuals to drive technical decisions[5]. In many of those cases, the same tactics, such as getting proofs of concepts up and running quickly, will still apply.

5.2 VALUE PROPOSITIONS – WHY DEVELOPERS WILL BE ATTRACTED TO YOUR PRODUCTS

Each product and service will have different features or uses that developers find valuable, but in general developers are driven by a core set of value propositions. Think about how you can address some or all of these value props in your developer marketing practices:

- **Productivity** – Developers want to move fast and get more done. While it's true that developers often tend to meander to 'entertain themselves' with coding (as *the gold plating anti-pattern* describes), they definitely want the 'boring' parts of their job to be

5. Sweeping changes in large organizations, like changing the core process and methodologies used to create software, often require a top-down approach. The case of introducing Agile software development at BMC Software in the early 2000s is a good example, as is the platform memo (referenced in the infamous "platform rant") sent by Jeff Bezos to internal development staff.

as speedy as possible. Productivity is not just about speed, but also leverage: the ability to do more things with fewer actions. Think about how your offering can accomplish tasks quickly ('get a simple mobile app running in minutes!') and allow a developer to multiply their efforts ('out of the box, this mobile app integrates with the top 10 social networks!').

- **Speed to market** – While an effect of increased productivity, the ability to get products to market faster can be a strong motivator for developers. The more entrepreneurial developers (those at startups, for example) will have competitive and revenue motivations to get their applications into the market quickly. Other types of developers will be eager to get their code into the hands of end users quickly to take pride in having their application actually used. In general, the culture of developers rewards and respects 'shipping,' so speeding up the process of getting their software out the door will be a strong value proposition.
- **Cool new technology** – For many developers, programming is not only a source of entertainment but a way to advance their careers. These developers are always on the lookout for new ways to solve problems and new technologies to use. Think about how your offering can appeal to new, *'horizon 3'* vision thinking and attract the interest of developers who suffer from 'shiny-object syndrome.' One might theorize that part (although not all, by any means) of what makes *polyglot programming* popular is this desire to have variety in the developer toolbox.
- **Stability** – While many developers like exploring and learning new technologies, a seemingly equal amount of them enjoy the stability that comes with applying their existing learning and skills. It's not that these developers don't want to learn new things, it's that they don't have the time to become as expert in new technologies as they are in existing ones[6]. Think about how you can appeal to the existing skills and knowledge a developer has, increasing the value of their existing skills rather than requiring them to take the time to learn new skills. Another way of looking at it is: why is your offering 'the safe choice?'
- **Solving riddles** – Another subculture of developers values the challenge of solving riddles and 'wicked problems.'[7] They're the types who will spend hours on mental puzzles and try to drive down the seconds spent to solve a Rubik's Cube. Similarly, in their daily work – and, thus, the technologies and vendors they ally themselves with – they'll be driven toward hard problems and technologies that help them. Much of the early respect (and, thus, marketing power) that cloud computing and NoSQL databases received was because these technologies solved large, complex problems – the discussions and marketing around them therefore played toward this interest in solving riddles. Think about how you can highlight a clever way you solved a complex technology problem, not only to display your intellectual tenacity, but also to highlight the similar spirit of your offering[8].

6. For an extended discussion of lack of time leading to learning avoidance, see the discussion in episode 62 of the Accidental Tech Podcast.

7. See the 2009 On the Media interview with the creator of Tetris, Alexey Pajitnov, whose interest in puzzles and riddles drove him to develop the famous game.

8. One of the more interesting examples of this – the writing of which is pure marketing – is Backblaze's ingenious reaction to the 2012 flooding in Thailand that prohibitively increased the price of hard drives.

- **Satisfying requirements** – While solving riddles and learning new technologies are some of the fun, even entertaining, motivations that drive developers, much of their daily life is consumed in searching for ways to solve relatively boring problems, or 'business requirements' that they're beset with. These requirements could be things like integrating with Active Directory, complying with PCI audit needs, properly ETL'ing data sources for further processing, or meeting scaling-up needs for retail applications. There might also be top-down mandates like 'move to cloud technologies to better control costs.' Think about how you can quickly explain how your offering satisfies common technology and business requirements so that developers can quickly see if it will meet their needs.
- **Getting paid** – Many developers actually enjoy 'hacking' business more than one would think, and are eager to learn how they can monetize their ideas and development projects. The iTunes App Store, Android Marketplace and other platforms, and the perceived promise of easy money, have received much attention from developers in recent years and have stoked the fires of 'I may get rich! (or at least be able to be self-employed)' thinking that runs through many, though not all, developers. A specialized – and over-represented – subgroup of startup developers are especially interested in the business angles of development.

There are other value propositions, but the set described here provides a good starting point for the value props unique to your offerings and the developers you want to reach.

5.3 SPEED, OR LOW BARRIERS TO ENTRY

Speed is always a good guiding light for planning how to market to developers. How fast can they get your framework installed and ready for coding? How quickly can they find the documentation that explains how to authenticate to your APIs? These considerations are often called 'low barriers to entry,' which is the easiest metaphor for thinking about speed. Do developers need to sign up for an account to view your documentation? Do they need to pay to test out your framework or learn it? Obviously, if the answer is 'yes' to those questions, you're not helping your marketing efforts.

Here are a few common areas to focus on in terms of speed, or low barriers to entry: ensuring you provide free use of development tools (unless that's the core product), without even requiring creating an account; doing proofs of concept and tire-kicking, as profiled in the SAP case study (see Section 6.1) and done with the Docker try-it online demo; and avoiding requiring users to create and sign into an account to read documents or view your community content.

5.4 SHOWING UP: FINDING YOUR COMMUNITY

While often portrayed as taciturn, introverted loners, most developers are actually very chatty and enjoy being in groups, especially groups that discuss programming. From a developer marketing perspective, online communities are largely about driving attention with content. That content will fit in various mediums like support forums, docs, chat rooms like IRC, actual code for extensions and projects, videos, screencasts and blogs. Developers find these communities and sites useful for both technical and social reasons – usually to help them do their job with resources for learning, troubleshooting and information. There are also benefits in the areas of reputation (which can help with careers or simply self-satisfaction), business development and entertainment.

5.4.1 WHY DO DEVELOPERS SEEK OUT COMMUNITIES?

Developers spend much of their time researching and learning in order to get their job done, so content that helps educate developers and solve problems is highly prized and a natural form of content for developer communities. Researching is such a large part of a developer's life that *some may even half-jokingly say they're actually experts at Googling*, not programming.

The social aspect of communities is important as well. Software development is a highly collaborative activity, with much of the intermediate- to expert-level skills learned word-of-mouth as craft (rather than with rigorous, well documented procedures). More senior developers discuss these craft learnings, and they need forums for this discussion.

Also, while it may seem far-fetched, developers also look to communities for entertainment, especially in the form of podcasts and blog posts that are often transmitting developer-world news and craft, while at the same time being jokey and human enough to simply help the developer pass the time in commutes or during other boring spans of time, like all-hands meetings, filing expenses or waiting for test suites to run.

For developer marketers, the important point is that developers 'show up' to communities and are there for you to interact with, almost waiting for you to do something helpful and interesting.

5.4.2 WHERE ARE YOUR DEVELOPERS?

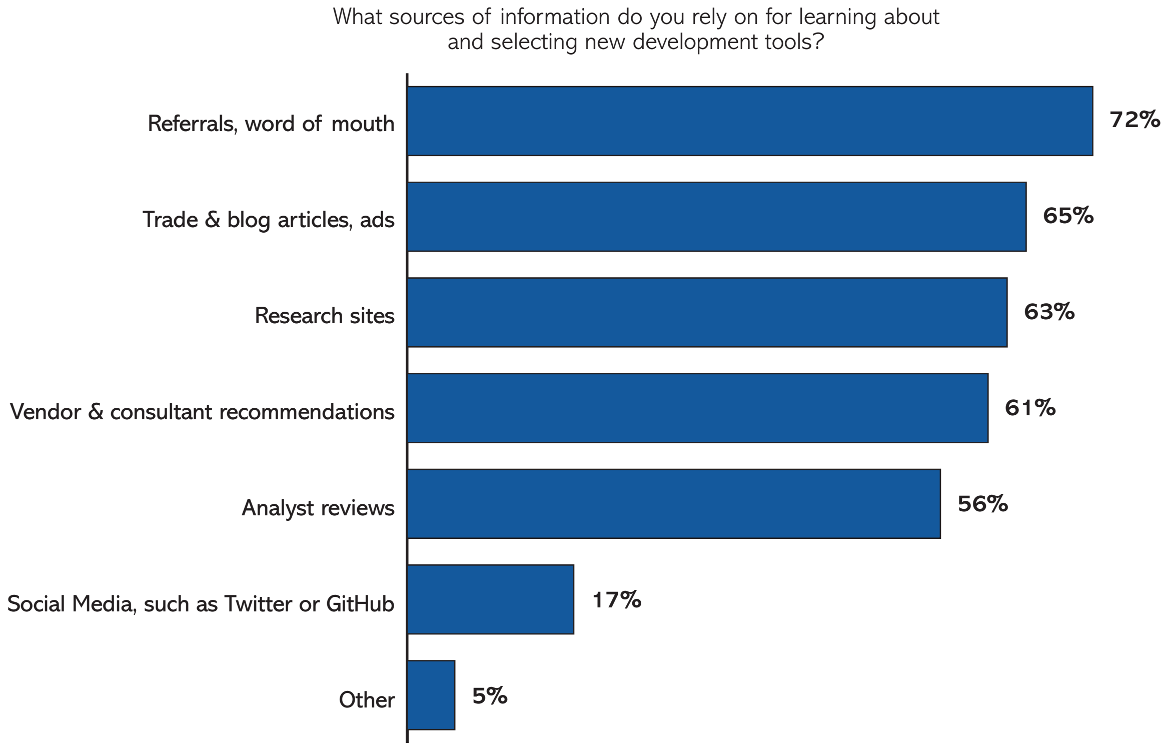
The 'places' that developers end up congregating are rarely the vendor's avenue of choice. It's little wonder, since vendor-created communities are often dedicated to one topic (the vendor!) and fall into neglect all too quickly, as evidenced by ancient content-editing tools and only a trickle of new features. With rare exceptions, the developer marketer will have to go out into the world (often the Web) to find and interact with their community. As things progress, you may be able to rein the community in, perhaps through a blog or a highly maintained community site centered on docs or support. The first step is to find out where your community is loitering, so to speak, then go there and interact with them.

Finding where developers cast their attention and, thus, where you should spend your time to 'hang out' with them is a tricky task. Much of it involves controlling for your time and money spent. While you can spend a lot of your resources, for example, traveling to regional user groups and meetups, this may not pay off as well as building up a strong Twitter program coupled with good blog content or, even better, ensuring that your company shows a good face, through code, in GitHub. The online sites and conferences where developers congregate change over time, and you should continuously monitor the effectiveness of any given Web-based channel.

Traditionally, email lists were used for this, as well as now ancient sites like Ward Cunningham's *c2 wiki*, popular blogs like Joel Spolsky's *Joel on Software*, or even in-person meetups. These traditional methods are still used, but dedicated developer communities, such as Stack Overflow and even the hosted version-control site GitHub, are increasingly used for this discussion as well. Developer marketing professionals tell us that they also see a material amount of traction from sites like Reddit, Hacker News and even Facebook. For real-time interaction, Google Hangouts is being used more. And, despite IRC decreasing in general popularity, the free-wheeling chat-room world of IRC has *risen in popularity with developers in recent years*. As we profile in our case study of SAP, developers also do show up at vendor-dedicated communities.

In addition to all of this churn and the various hashtag-festooned social channels, traditional channels are still effective. In one of our recent DevOps Surveys, we asked participants where they got recommendations and research for their tools selection:

FIGURE 1: TOP INFORMATION SOURCES FOR DEVELOPMENT TOOLS



While the DevOps market does not map perfectly to the developer market as a whole, the overlap is significant enough that it does provide useful input (after all, the 'Dev' in DevOps stands for developers). This study was conducted in 1Q 2014 and included 200 North America-based participants in the 'mainstream DevOps market' – and it is thus skewed *away* from the technology industry, focusing instead on the developers and operations teams in 'mainstream' industries.

Many of the 'elite,' especially chatty developers we speak with often say they get all of their news from Twitter and GitHub. Obviously developers don't learn in 140-character increments, but rather discover larger pieces through Twitter. Anecdotally, this seems to be the case for that niche of influential developers. However, in the wider market, as our DevOps Survey shows, more traditional information sources are used. Unexpectedly, word-of-mouth is strong, pointing toward the need to build up an effective referral network or, more colloquially, ongoing 'conversation' in the market. So-called 'viral hooks' in your product, like the ability to invite users or add administrators[9], can also help promote word-of-mouth as a channel. The other sources cited by survey respondents point toward methods of helping developers understand and learn new technologies, with research sites and analyst input with trade magazines (likely online versions thereof) and blogs ranking high as well.

Developer research sites like Stack Overflow and Super User are incredibly popular for this type of research. And despite being dead last in the DevOps Survey, social media channels are still important sources for information discovery; again, anecdotally, this is especially true for the more highly influential, 'elite' developers that can help spread your developer marketing to their vast array of followers.

5.4.3 IN-HOUSE COMMUNITIES: AVOID BUILDING A GHOST TOWN

If your company can properly fund starting and maintaining its own community, you have a shot at getting even more developer marketing power than if you have to 'go out' into the Web. Large companies like Microsoft, IBM and SAP have the will and resources to fuel their own communities, as do some companies whose portfolio is small enough to make in-house communities sustainable.

In-house communities are websites that are owned and operated by a vendor, such as SAP's SCN. They tend to replicate what you'd see on other sites (support forums, how-to info and even some 'off topic' philosophizing) and are often centered on the vendor's projects and content (like docs).

The most important marketing-centric activity for an in-house community is simply keeping alive and active. Vendors often have a 'if you build it, they will come' stance on community, assuming that developers will gleefully come and contribute content. Nothing could be further from the truth: early on, the developer marketer must painfully

9. Many products that follow a "bottom-up" spread have functionality to add in administrators and otherwise integrate with "official" IT department processes long after the product has virtually been picked up by developers.

bootstrap the content and activity, and keep up with it to get over the hump. A developer will look at an in-house community and if it appears to be a 'ghost town,' will quickly move on and likely never return.

The ultimate, sustainable goal is to get the community to generate its own content, but that's a far-off goal at first. This means that the in-house community managers must relentlessly fill it with content and become virtual barkers to draw people in. (Types of content and topics for the above are covered in Section 5.7.)

5.4.4 GAMIFICATION & PRODUCT MANAGEMENT

Many communities use 'gamification' techniques, which track the activity of community members and publicly lists their rankings as awards. This creates another type of content – reputation – that is unique to your site and valuable to the developers you're trying to attract. More developer-centric communities, such as Stack Overflow, Hacker News and GitHub, all use gamification techniques to encourage and reward engagement, and even tune content listings[10]. Communities like Spiceworks and SolarWinds' Thwack also thrive on this gamification system, and serve as a good, non-developer examples.

These communities also showcase another positive effect of grooming and growing your community – namely, mining the community for production management direction. If a certain area of your product is especially irksome or popular, your community will likely talk about it, giving you valuable input into product direction. Equally valuable for developer marketers is to point this feedback loop out to the developers themselves: your community wants to know you're paying attention to them and incorporating their feedback into your product direction. So, when you do so, tell them! Spiceworks frequently does this with its technical community: at its annual conference, it lists the top features requested by users and honestly walks through the ones that it decided to implement that year. The features are not always the 'top five,' and they span the charts, but the community marketing message is clear: we're listening and responding.[11]

5.5 DOCUMENTATION IS A STOREFRONT

Getting access to documentation, manuals, best practices, common templates and other forms of 'docs' should be as free and easy to access as possible. There's little reason to lock up manuals behind logins or paywalls. Once a developer learns about your offering, they'll want to learn how it works and start investigating how they can use it. Documents are paramount here – most importantly documents that are 'living,' not only with comments but replies from you, the owner of the document, correcting and modifying the docs accordingly. Developers will also respond well to seeing all the past versions of your documents, and even the version-control logs so they can compare the docs. You

10. See this 2011 commentary from Jeff Atwood on introducing gamification to Stack Overflow for more background and guidance on how you might introduce it into your community.

11. While Spiceworks and Thwack are not developer communities, the 'IT Pro' crowd it supports has very similar patterns and practices as developers, and this provides for good examples.

should consider checking the docs into public version-control systems like GitHub, which will layer in additional social value that can be used for raw promotion and marketing data.

As with most developer marketing, careful gardening of your documentation also generates content that can be used for marketing. You should think of your docs as one of your storefronts – if not the primary storefront – for your offerings. This means instrumenting your documents with marketing automation and tracking. While you may not be tracking who is accessing the docs at first, you may eventually be able to identify the individuals and then retroactively see a history of docs they've accessed. Rather than being spooky spying, the intention is to (once again) speed up the developer's interaction with your company, in this case when it comes to sales. Knowing what a developer (or group) at a prospective company has looked at will help your salespeople identify what they're interested in and may want to pay for. The group may be interested in your storage-as-a-service offerings, or perhaps they are interested in how to integrate their queue with an ERP back-end.

Knowing why developers are using your technology will help speed up the process of forging a commercial relationship. Even more important – from the developer's perspective – it will allow you to identify developers who are ready for a sales conversation and those who are not. For example, just because I downloaded one white paper on your offering's architecture does not mean I'm interested. But if I also read through the install instructions and walked through an online tutorial, I've demonstrated that I'm more serious because I've committed so much time to learning your system... or I'm just really bored.

5.6 CODE AS MARKETING

It's little wonder that one of the better ways to interact with developers is through actual code. Open-source-based companies tend to do this as a by-product of their open development model, but cloud services and more 'closed' companies can often do more to use code as marketing. In this day and age, the code doesn't even need to be open-sourced; it just needs to be available for free use. Being open source is still important and drives marketing and product benefits, however.

While admittedly biased toward open source, *the 2014 Eclipse Foundation community survey* shows a steady interest in not only using open-source software, but contributing to open software. In that same survey, developers cite the sheer desire to 'give back' as the top reason to contribute, along with learning, providing bug fixes and even advancing their careers. While it's unreasonable to expect every developer, or even a majority of developers, to contribute code, almost every developer will expect you to show your code and provide plenty of it to the community. From a marketing standpoint, then, code can be thought of as a highly specialized type of 'content' used in your developer marketing efforts.

5.6.1 MINI CASE STUDY: CLOUD PROVIDER USES GITHUB TO JOIN THE CONVERSATION

One cloud provider we know of offers a good 'mini' case study of using code as marketing. Seeking to compete with Amazon Web Services' seemingly iron grip on developer relations in the cloud world, this cloud provider initiated a program to raise its profile in GitHub. To do this, the provider created several command-line-driven projects for interacting with its API, as well as frameworks for different types of application delivery, like Web and mobile.

The company worked with an independent development house in mobile and Web development that had already established a good reputation in GitHub. Not only did this result in actual working projects and products that were beneficial to developers using the cloud provider, but it also allowed the provider to coast off the reputation of the contractor: as the developers at the contractor would check-in code, notifications of each check-in were added to their activity streams in GitHub, spreading to their followers. Similar efforts are done at SAP, where the developer relations program involves activity encouraging internal SAP developers to contribute more and more to GitHub.

5.6.2 MINI CASE STUDY: THE JAVA PET STORE

One of the better historic examples of code as marketing is the J2EE Pet Store. The enterprise bundling of Java, then called J2EE, was necessarily complex and hard to get started with; there was simply too much functionality to understand at first – a real developer marketing challenge! In addition to numerous books, docs and human interaction such as training, Sun Microsystems released the Java Pet Store, which was a simplistic example of building the software stack used by a fictional pet store, all based on J2EE. This 'code as marketing' effort allowed developers to see J2EE in action and, despite warnings that the code was not intended for production use, would often form the basis for real projects!

The Pet Store became such a popular mental model that Microsoft offered its own Pet Store for .NET as well as other popular frameworks like Spring[12]. Similarly, the example code put out by vendors is often the 'starter' used by developers to begin their own projects. Undoubtedly, the Apache Web server modules and recipes shipped by Puppet Labs and Chef have been used by developers as the starter for their automation projects.

5.6.3 CODE AS MARKETING FOR HR

One special application of code as marketing that's worth pointing out is the role of code for recruiting and retaining your own developer talent. As any company hiring developers knows, finding the perfect developer is both difficult and costly – and once you have one employed, you'd like to keep them. Increasingly, top-tier developers know they must maintain their reputation in the overall community. One of the easier, more effective ways to do that is open-sourcing code, or at least frequently talking about it. The 'labs'

12. See the Wikipedia the Java BluePrints entry for more Pet Store history.

blogs you see at sites like *Netflix*, *Etsy*, and even *Volvo* (which have little ostensible reason to do developer marketing) are targeted at exactly this HR task[13], in addition to disseminating good ideas and learning to the community. Encouraging your developers to participate in such labs programs, then, not only helps with marketing but can help with recruiting and retention.

5.7 CONTENT

Many of the tactical suggestions here require content aside from code. More than just a chunk of text, 'content' can be used in multiple channels and recast into different formats. For example, a podcast episode that describes how the current release of your software was designed could show up as a blog post (perhaps edited down from a transcript), be recast into a webinar, or even be put into the form of a talk for conferences.

5.7.1 TYPES OF CONTENT AND MEDIUMS

At a minimum, it's likely that you should have an email newsletter, blog and Twitter account for your company that can used for developer marketing. The email newsletter will be a valuable, fully controlled channel to developers and provide rich analytics about open and click rates, narrowed down to individuals, allowing you track successful content and which individuals are engaged (or not). As covered in Section 5.4, showing up at sites like Stack Overflow and GitHub is also important, and you should always be on the lookout for where your developer audience is hanging out and for new mediums to reach them.

Successful mediums for content can vary from vendor to vendor, but community content portfolios often include some of the following:

- **Blogs** – More than likely, one of the first things you should have is a blog. This blog should have weekly, if not daily, news and info on happenings in the community, including new releases and profiles of community members.
- **Podcasts** – These can serve the same functions as a blog, hopefully with multiple regular hosts from the vendor and the community. Podcasts are notoriously hard to track beyond raw downloads, but are generally easy and quick to produce, with the possibility of establishing dedicated listeners. Coupled with transcripts and rewriting them into blog posts and presentations, podcasts can be a good front-end for generating highly reusable content.
- **Support forums** – A support forum that not only allows for problem solving but, as with the Spiceworks example, continually ranks and lists top bugs and requested features.
- **Code** – This could include the actual source code and extended plugins and configuration for your products. This is not only the code for your product, but also examples and other code-as-marketing activities, as discussed in Section 5.6.

13. See the Netflix "Tech Blog" entry on building the new streaming app framework for an example.

- **Libraries and repositories** – Hosting repositories of extensions and configurations done by your community as the authoritative source, like Docker hosting images made by community members.
- **Documentation** – The manuals and other docs for your offering, covered in Section 5.5.
- **Demos, screencasts, interactive tutorials** – Online, interactive tutorials like *the Docker walk-through*. Docker (the company) says this tutorial is a strong selector in its sales funnel process and is very important for its high-velocity, low-touch go-to-market model.
- **Books** – Books can be a great type of content for an extended treatment of your product and the context it exists in (what problems it solves, new opportunities your product creates when developers use it, etc.). Free books (digital, hard-copy or both) are good for lead generation and helping make the market you're operating in. One example is SmartBear's *Best Kept Secrets of Peer Code Review*, which helps sell its code-reviewing product, Code Collaborator. Not all developers are convinced that code reviews are a worthwhile practice, so this book was published to not only make the case, but explain practices: a classic example of market-making[14].
- **Presentations, webinars and other talks** – Hopefully these will include slides (hosted at Slideshare, for example) and recordings. These are not purely sales pitches; they give you the chance to establish credibility and engage with your community. Many of the major topic ideas in the next section can be recycled into presentations.

As far as frequency goes, 'post' (if that's the appropriate word for the medium) as often as possible without affecting the quality of the content. The important part is to keep the stream of content up, and be on the lookout for new mediums to spread the same content in. As *Danielle Morrill outlined in her discussion of developer marketing at Twillio,* you can (and should) recycle these content items by rewriting or simply reposting them on your blog and other mediums annually.

5.7.2 TOPIC IDEAS

Here we present a list of common, effective topics that we often provide to clients who ask for ideas on what types of content to create – many of which can be used across various mediums. While not an exhaustive list, it provides some good starting ideas:

- **101: Getting started** – While 'getting started' content should clearly be in your documentation, tutorials and other 'recipes' for using your product make for good content. Adding commentary to stock tutorials is good as well.
- **Ongoing educational content** – Beyond getting started, content that walks through how to use other features (from basic to advanced) in your offering can be very valuable, content-wise. Don't rely on the manual and SDK documentation as the sole source for this material, and instead find opportunities for tutorials and walk-throughs. Sites like

14. See also Jason Cohen's brief presentation on how this book was used to fine-tune SmartBear's developer marketing campaigns and some additional commentary on books vs. white papers.

IBM's developerWorks excelled at this during the early enterprise Java days, and user management API company Stormpath reports that this type of content is "super powerful."

- **Wicked problems and war stories** – Write up how you solved interesting problems in gory detail – especially for service providers and others who want developers to join their orbit, like *Backblaze showing off some nerdy thinking* when hard drives become too expensive. This topic might also be the discovery of new coding techniques or '*design patterns*.'
- **Something you tried and discarded**[15] – Simply talking about your success is not sustainable, or always the most interesting topic on hand. In putting together your business, developing your product and otherwise finding your way, you'll go down many false paths. Developers will appreciate hearing not just about wicked problems that were solved, but solutions that ended up not being helpful.
- **Sausage-making** – Many developers enjoy reading stories of how other developers created their products. With each release (large or small), interview the product managers or development team to get the story of how the release was done, find the 'whys' of new features and how you're imagining they'll be used.
- **Philosophizing about methodology, the industry or culture** – This helps attract people to your macro-level concerns and movement, and builds brand awareness. 37signals Ruby on Rails was about simplifying development on a day-to-day basis, and shooting down the notion that complex was good (WS-*, Java, etc.). The company needed much 'air cover' (macro ideas) to tenderize the minds of developers who had been trained to embrace complexity by a decade of 'enterprise development' thinking. This can also take the form of discussing your company's culture, as 37signals (now Basecamp) also does frequently.
- **Case studies** – These can take the form of stock interviews of community members (see *The Setup* for an interesting, engaging approach) or full-blown case studies in PDF form. Obviously, it's important to focus on how your offering benefits the users as much as possible, lacing in instructive narratives of how your technology was applied, if possible. Podcasts are a good medium for case studies as well.

15. Suggested by Stormpath's Claire Hunsaker in her Cloud Marketing 101 talk, which also contains good tips on rethinking positioning for developer marketing.

5.8 CONFERENCES

Conferences also provide a good, though tricky, venue for reaching developers. While large conferences like JavaOne were once a focal point for developer communities, many large conferences have tended to fragment into smaller ones, or just disappear altogether in favor of online communities. However, conferences and smaller events like meetups are still key to a developer relations program.

We group conferences into four buckets for the purposes of developer marketing:

1. **Large vendor conferences** – Conferences centered on the commercial interests of a vendor or a group, such as VMware's VMworld or the OpenStack Summit. These conferences can be good for the sheer volume of attendees, but expensive.
2. **Your vendor conference** – Like the example above, except a conference that your organization funds and puts on, giving you much more control over your interactions there (and much more expense!). Large vendors like salesforce.com, IBM, Oracle and SAP rely on these conferences for key annual get-togethers of their developer community.
3. **Industry conferences** – Put on by trade press groups, 'non-vendors' like O'Reilly, and others focusing on the industry as a whole. These events can be good for disseminating your own ideas and reaching developers who are looking to learn about software development in general, not just the ecosystem of a specific vendor.
4. **'Unconferences' and meetups** – From smaller events like DevOpsDays to weekly meetups and lunches, this is where many of the more interesting 'conferences' have been happening of late. These events tend to be very low-budget (in all senses of the word) but can also be very rewarding for those who have an offering that aligns with the attendees (see Section 6.2 for an example).

In each instance, it's important to value the face-to-face time you have with individual developers. In addition to just attending and carousing with fellow attendees, you should always try to give at least one talk at the conference. Depending on the type of conference, the talk should likely be about a general problem area and how you solved it, rather than a direct product pitch. A case study often fits here as well. Whatever the talk may be, your goal is simply to introduce yourself, your company and, indirectly, your offering to the crowd.

If you have the budget, getting a booth (or 'table' at smaller events) is a good idea. Coupled with speaking and product announcements timed to coincide with the event, you can use the booth to provide deeper dives into your offerings and directly feed your lead-generation pipeline, but also to test out and fine-tune your developer marketing programs.

To that end, establishing clear goals and tactics for a conference is ideal. These goals may revolve around an announcement you make at the conference – for example, releasing a beta of a cloud monitoring tool, which you then give a talk about during the conference ("cloud monitoring is hard; here's how we got burned and ended up solving the problem") and also demo at your booth. Since this is developer marketing, having technical staff on hand is wise because developers may want to dig deeper into the technology beyond a salesperson's abilities. No matter how finely orchestrated your planning is, continually test out and observe what's working in order to fine-tune your approach, especially as your ambitions and budgets expand[16].

16. See Jason Cohen's tips for tradeshows for an exhaustive checklist of things tech companies should consider and plan for when going to conferences.

SECTION 6
Case Studies

The case studies we present here provide real-world examples of developer relations and marketing in action.

6.1 SAP

Founded in 1972, SAP is one of the largest enterprise application vendors in the world, backing the business software needs of its customers. The nature of enterprise software requires most customers to customize and integrate SAP's software with their existing and new business systems, creating the need for developers to tweak and build on the SAP 'platform.' To that end, SAP has its own programming language, ABAP, and a series of frameworks and platforms that developers, systems integrators, ISVs and startups use and extend.

At SAP's scale (2013 revenue was €16.81bn), a very conscious, well-funded developer marketing program is needed. SAP's developer-centric community program started in 2003, then called the SAP Developer Network (SDN), and it was renamed the SAP Community Network (SCN) around 2008 to include business analysts and other non-developers who participate in the SAP community[17]. The program services the estimated two million developers in the overall SAP developer ecosystem.

6.1.1 STRATEGIZING THE DEVELOPER RELATIONS PROGRAM

The goals of SAP's developer program are to educate developers about new features and products coming out and to support platform usage and sales. Any given developer in the SAP world is typically business-savvy (in that their domain is business software, not consumer software) and is looking for the most efficient way to learn how to use existing and new SAP technologies. For example, developers might want to explore writing mobile applications or using SAP's HANA in-memory database.

To structure its approach of working with two million developers, SAP segments developers into four categories, each with customized goals and approaches:

1. **Corporate developers** – Developers working in-house for companies using SAP's portfolio. These corporate developers are the ones who will make their company's overall technology stack function, and do much of the day-to-day work of making the SAP platform valuable to the customer. Keeping these developers happy and abreast of new offerings is important for keeping these accounts humming along nicely, and growing accounts by enticing developers to ask budget-holders for new functionality.

17. SCN Wikipedia entry, June 2014.

2. **ISV developers** – An ecosystem as large as SAP's relies on many third-party application developers to write companion applications and integrations with SAP's software. These developers need quick access to information and running environments to test out their integrations, and also would like to have clear commercial outcomes, such as access to SAP's marketplace.
3. **Systems integrators** – SIs that work in the SAP ecosystem are always looking to increase their billable hours and raise their profile to get new business. To that end, the developer program ensures that SIs learn about new technologies that can drive new business and uses gamification programs in SCN and recognition schemes like the SAP Mentor Program to raise the developer's profile (see below).
4. **Startups** – While startups generally have the same concerns as ISVs, they have additional concerns that warrant their own segmenting. Startups are not only interested in the technical aspects, but the developers are often the core entrepreneurial team at the startup, and are thus interested in business development and partnership activities with SAP around marketing and sales.

While this exact segmentation of developers may not apply directly to your developer program needs, it serves as an example of the degree of up-front strategizing and categorizing that's helpful when putting together a developer relations program.

6.1.2 TACTICS

A developer relations program as large as SAP's has many moving parts, so here we present a few highlights:

- **Eval speed** – One of the critical needs for developers is getting up and running quickly in order to 'kick the tires' of systems. Large portfolios like SAP's can be difficult to set up quickly, so the SAP developer program provides cloud-based labs of the software, such as HANA, and has not only ensured that SAP applications can be run in Amazon Web Services, but it will also run applications for up to 10 hours free in AWS. Regionally, the developer program has often helped fund these virtual labs – for example, in Latin American and India, where developers typically do not have easy access to expense such services.
- **Free tools** – Over the past few years, SAP has greatly simplified the licensing for its developer tool-chain. Developers had grumbled about the cost and fine-print around using developer tools for some time. Now, most of the tools are free, and the licensing is much more liberal, lowering the barriers to entry for tool use.
- **Conferences** – Each year, SAP hosts four developer-centric conferences – SAP d-code (formally called SAP TechEd) – in the US, Europe, India and China to reach its two million developers. Some developers cross over to the SAP business conference – Sapphire – as well. Clearly, gathering developers, and doing it globally, is a vital part of maintaining a thriving community.

- **SAP-hosted community** – The SAP program has its own online community – scn.sap.com – which closely tracks developer profiles and activities, and includes gamification elements that rank and show the 'achievements' of developers, such as helping solve problems in support forums.
- **External community** – The developer relations program has been working with SAP-badged developers to put more and more of their code in GitHub in order to start creating new relationships with various developer communities. For example, SAP recently put HANA drivers for various other platforms and frameworks in GitHub, and very quickly drummed up interest from the node.js community – something that would have been difficult to do without applying this practice of 'code as marketing.'

One of the developer relations efforts, the SAP Mentor Program, warrants more than just the 'bullet point' treatment here. The Mentor Program recognizes prominent SAP community members, with many of them being developers. There are over 140 Mentors – an estimated 70% of them developers, SAP says – with whom the company engages frequently and provides additional services, much like the Microsoft MVP Program.

These Mentors not only act as third-party advocates for SAP in the marketplace, but also give regular input to the company, right up to the highest level: there are monthly meetings with Mentors to test out new product direction, programs and conference topics, and Mentors are also given meetings with SAP executives at various SAP conferences. The Mentors benefit professionally in many ways, from getting previews of new products and other valuable information to free event attendance and, of course, the cachet that comes with being recognized as a valuable developer in the SAP community.

6.2 DEVOPSDAYS AUSTIN

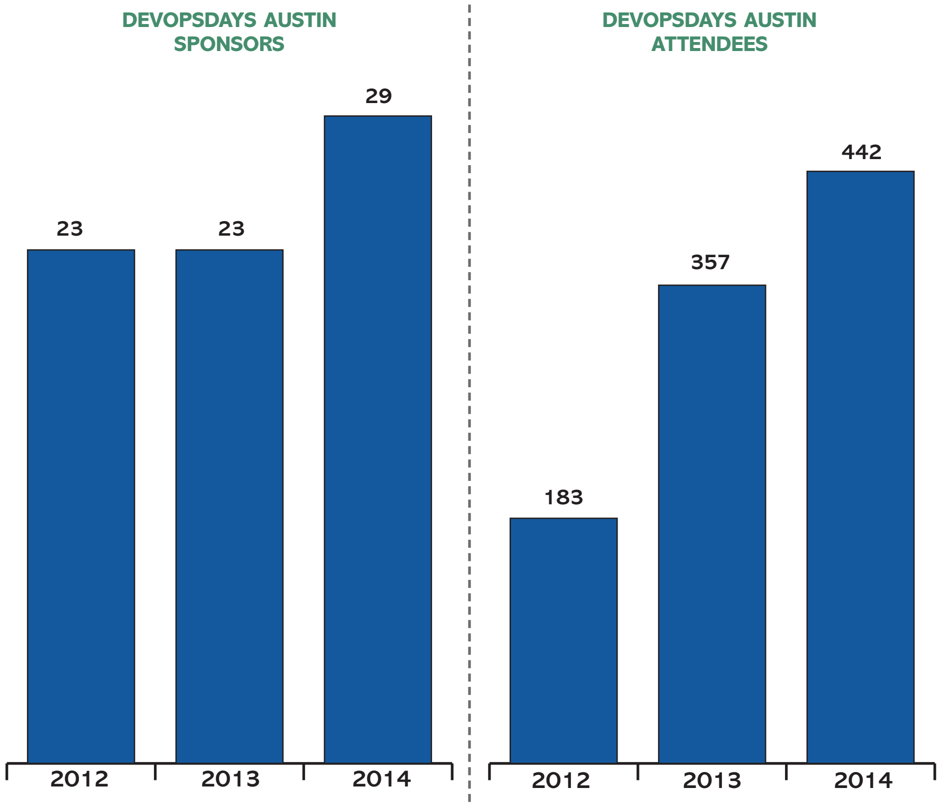
While not a company, looking at the multiyear success of DevOpsDays, an 'unconference,' is useful for understanding developer relations. As we've mentioned, 'DevOps' is not purely about developers; it includes developers and operations staff. More broadly, the thinking and tactics of developer relations described here apply to DevOps as well, in our experience. Equally important, as we're tracking the spread of DevOps in popularity and across industries, reaching this audience is likely important for many readers considering how to engage with this community.

Started in Belgium in 2009, DevOpsDays is truly an unconference run by volunteers. It has spread globally, with events in North America, Europe, India, Australia, the Philippines and many other sites. Many of these regional events are annual, including the Austin, Texas event. Vendors pay for sponsorships, typically ranging from $1,000 to $5,000, to have tables, sponsor the bar and do five-minute 'lightning talks' between keynotes. There's a very minimal ticket fee for attendees.

The format is typically half planned and half 'open spaces': keynotes, accepted talks and then afternoon sessions that are proposed and voted on by the attendees in barcamp style. This format and the 'vendorless' business structure of the conference encourage much participation from the attendees, both as listeners and presenters. Anecdotally, in our experience, the 'quality' of attendees has been very high, matching the depth and practitioner-heavy content. This serves as a good example of some of the tactics for effective developer relations activities: focus on discussing cool new technologies, going over 'war stories' and use cases, and overall disseminating useful information on how to solve wicked, intriguing problems and 'riddles.'

Narrowing the focus to one geography, Austin, the ongoing momentum, numbers and anecdotes about sponsoring vendors confirm that events like DevOpsDays are helpful conferences for developer relations programs. For three years, the conference has sold out and grown in attendance, maxing out its sponsorship opportunities:

FIGURE 2: GROWTH OF SPONSORS AND ATTENDEES AT DEVOPSDAYS AUSTIN



Source: DevOpsDays Austin organizers, June 2014

Sponsors have told us that the quality of leads generated at these events (from setting up tables and doing the usual conversations with attendees) are extremely high. Dell says that for a "minimal amount of sponsorship" and raffling off one Sputnik developer laptop unit, it generated 100 "great leads," according to Barton George, Director, Dell Services. At the same time, the volunteer organizers say that the size of the annual conference has been primarily limited by the size of venue they've been able to book, suggesting that attendee demand is outstripping supply.

Other industry conferences, especially the open-source foundation conferences such as the OpenStack Summit, ApacheCon and Eclipse Summits, have some of the vendorless aspects seen in DevOpsDays.

6.2.1 BMC SOFTWARE AT DEVOPSDAYS

BMC Software's involvement in particular is emblematic of the success a company can have in developer relations at an event like DevOpsDays. For three years, BMC has sponsored and arranged for live video feeds of most DevOpsDays, including the Austin events, associating its brand and management products with the event, in addition to building up goodwill with the DevOps market. BMC's Christopher Little, who works on DevOps marketing for the company, says these BMCtv watermarked videos provide great content for BMC's DevOps community sites, and have generated over 170,000 views. The events have also become a great place to talk with existing BMC customers and prospects – at the same time even – feeding the word-of-mouth and face-to-face components of BMC's DevOps sales process.

While BMC, in our opinion, is not a slam-dunk for being on the metaphoric shortlist for DevOps, the company's involvement in DevOpsDays has earned it an invitation to the party that not all of its vendor peers seem to have.

6.2.2 SUMO LOGIC AT DEVOPS DAYS

Another company, Sumo Logic, has found success in reaching its core community and buyers at DevOpsDays. At the recent DevOpsDays Austin, the company had a booth and also gave an hour-long talk. The booth traffic generated 75-100 leads (out of 442 attendees), with a conversion to active opportunities of 10-15%, and two deals that are highly likely to close. As Kurt Zanca, the Sumo Logic sales manager at the event, said, "You're not gonna get much better than that" at a conference. In addition to raw lead generation, the event was even more valuable to the company for the knowledge exchange with actual practitioners and the opportunity to have Sumo Logic customers talking about Sumo Logic at the event. The company used its speaking slot not as a pitch, but to tell the history of DevOps at Sumo Logic itself – a wise move that engages in knowledge exchange without 'selling.'

INDEX OF COMPANIES