Cutter IT Journal

The Journal of Information Technology Management

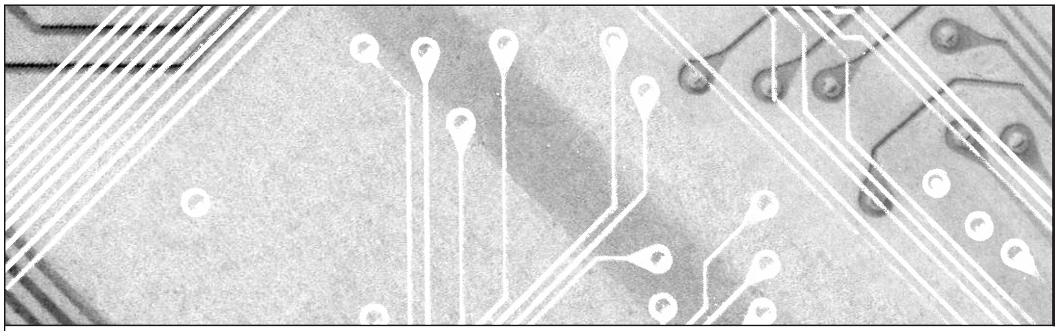
Vol. 26, No. 9
September 2013

"One of the great things about the API economy is that it is based on existing business assets.... What were assets of fixed and known value suddenly become a potential source of seemingly unlimited business opportunities."

— Giancarlo Succi and Tadas Remencius, Guest Editors

Profiting in the API Economy

CUTTER CONSORTIUM



Cutter IT Journal

About Cutter IT Journal

Part of Cutter Consortium's mission is to foster debate and dialogue on the business technology issues challenging enterprises today, helping organizations leverage IT for competitive advantage and business success. Cutter's philosophy is that most of the issues that managers face are complex enough to merit examination that goes beyond simple pronouncements. Founded in 1987 as *American Programmer* by Ed Yourdon, *Cutter IT Journal* is one of Cutter's key venues for debate.

The monthly *Cutter IT Journal* and its companion *Cutter IT Advisor* offer a variety of perspectives on the issues you're dealing with today. Armed with opinion, data, and advice, you'll be able to make the best decisions, employ the best practices, and choose the right strategies for your organization.

Unlike academic journals, *Cutter IT Journal* doesn't water down or delay its coverage of timely issues with lengthy peer reviews. Each month, our expert Guest Editor delivers articles by internationally known IT practitioners that include case studies, research findings, and experience-based opinion on the IT topics enterprises face today — not issues you were dealing with six months ago, or those that are so esoteric you might not ever need to learn from others' experiences. No other journal brings together so many cutting-edge thinkers or lets them speak so bluntly.

Cutter IT Journal subscribers consider the *Journal* a "consultancy in print" and liken each month's issue to the impassioned debates they participate in at the end of a day at a conference.

Every facet of IT — application integration, security, portfolio management, and testing, to name a few — plays a role in the success or failure of your organization's IT efforts. Only *Cutter IT Journal* and *Cutter IT Advisor* deliver a comprehensive treatment of these critical issues and help you make informed decisions about the strategies that can improve IT's performance.

Cutter IT Journal is unique in that it is written by IT professionals — people like you who face the same challenges and are under the same pressures to get the job done. *Cutter IT Journal* brings you frank, honest accounts of what works, what doesn't, and why.

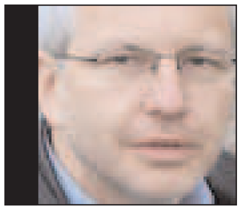
Put your IT concerns in a business context. Discover the best ways to pitch new ideas to executive management. Ensure the success of your IT organization in an economy that encourages outsourcing and intense international competition. Avoid the common pitfalls and work smarter while under tighter constraints. You'll learn how to do all this and more when you subscribe to *Cutter IT Journal*.

Cutter IT Journal®

*Cutter Business Technology Council:
Rob Austin, Ron Blitstein, Tom DeMarco, Lynne Ellyn, Israel Gat, Vince Kellen, Tim Lister, Lou Mazzucchelli, Ken Orr, and Robert D. Scott*

*Editor Emeritus: Ed Yourdon
Publisher: Karen Fine Coburn
Group Publisher: Chris Generali
Managing Editor: Karen Pasley
Production Editor: Linda M. Dias
Client Services: service@cutter.com*

Cutter IT Journal® is published 12 times a year by Cutter Information LLC, 37 Broadway, Suite 1, Arlington, MA 02474-5552, USA (Tel: +1 781 648 8700; Fax: +1 781 648 8707; Email: citjeditorial@cutter.com; Website: www.cutter.com; Twitter: @cuttertweets; Facebook: Cutter Consortium). Print ISSN: 1522-7383; online/electronic ISSN: 1554-5946.



Opening Statement

by Giancarlo Succi and Tadas Remencius, Guest Editors

If you look at history, innovation doesn't come just from giving people incentives; it comes from creating environments where their ideas can connect.

— Steven Johnson

One way of looking at the API economy is to consider it as a combination of technological advances and new social and cultural trends that merged to form an interconnected environment ripe with exciting business opportunities. At the center of this fertile ground are Web APIs, which connect providers and consumers (developers) into a symbiotic ecosystem. In a sense, APIs connect companies much as social networking sites connect people. However, where the latter are driven by social needs, API ecosystems are based on a win-win scenario in which benefit is gained not only by providers and consumers, but also by end users, who receive more and better products and services targeted at context-specific user experiences and expectations.

Innovation is the specific instrument of entrepreneurship … the act that endows resources with a new capacity to create wealth.

— Peter Drucker

One of the great things about the API economy is that it is based on existing business assets. There is no need to design new products or come up with new services — companies can simply capitalize on their existing core business strengths. This potentially allows any company, regardless of its size or actual business, to join in the new economy by exposing some of its assets to its partners or the general public. What were assets of fixed and known value suddenly become a potential source of seemingly unlimited business opportunities.

The role of API ecosystems makes the API economy quite different from a typical business philosophy. Companies are no longer in direct control of the outcomes of their actions — the impact of API consumers on the success of the exposed APIs is simply too great. No matter how well producers prepare or how many investments they make, the ultimate factor in the outcome of an API program is the API consumers and their ability to develop innovative and timely products and services. In fact, both parties depend heavily on each other, while having a relatively low level of direct contact and little control over each other's actions. The advantage of this arrangement is that it offers a lot of flexibility and prevents tie-ins on both sides: providers generally can change or stop an API program quite easily, while consumers have the freedom to switch to alternative APIs from other providers at low or zero overhead cost.

Innovation is taking two things that already exist and putting them together in a new way.

— Tom Freston

On the API consumer side, the API economy is all about innovation and rapid time to market for new solutions. Even fresh startups with zero starting capital can quickly produce new mobile apps by combining multiple existing APIs in innovative ways or applying them in new contexts.

In a sense, APIs connect companies much as social networking sites connect people.

We can see innovation not only in developed solutions and services, but in business relationships as well. New business models centered around APIs are appearing and evolving at a fast pace.

The late Steve Jobs stated in one of his speeches that "innovation distinguishes between a leader and a follower." It is interesting to consider how this applies in the context of the API economy. Here innovation becomes a shared commodity that goes in a continuous cycle from producer to consumers and back again. So who is a leader and who is a follower in this case?

If anything can be said for sure, it is that the time of the API economy is now, and an interesting time it is. The potential benefits are many, and there are plenty of success stories out there. However, which of those benefits are achievable by specific companies and in what ways remains unclear. This issue of *Cutter IT Journal* explores these topics and aims to help companies answer the question of how the various benefits offered by the API economy can be unlocked.

If anything can be said for sure, it is that the time of the API economy is now, and an interesting time it is.

We begin with an article by Cutter Fellow Israel Gat and his coauthors, who take a critical look at some of the main selling points encountered amidst the hype surrounding the API economy. The authors first discuss whether the API economy is really a new type of economy and suggest that there are, in fact, a variety of different economic models in play — some old and some new. They then look at the current state of the API economy, taking note of its rapid growth. They elaborate on potential future directions, pointing out the risk of market saturation, and conclude that now is probably the best time for businesses to invest in the API economy, if they haven't done so already. Gat et al. continue with an examination of how the API economy affects human innovation and whether it really serves as a driving factor. The authors show that "human innovation" is a somewhat misleading term, arguing that it would be more accurate to talk about "human creativity" in this context. Finally, the authors examine the claim that the API economy acts as a leveling factor for companies of different sizes. While they agree that the API economy provides opportunities for smaller companies (particularly in the role of consumer), Gat et al. highlight the other side of the coin — the advantages that big players have when it comes to marketing their APIs and enforcing provider-favorable SLAs and API business models.

In our next article, Chandra Krintz and Rich Wolski present their strategy for implementing an API governance platform for managing, unifying, delivering, and composing APIs in a commercial setting. They advocate the use of cloud-based technologies and emphasize the need for consistent control over the APIs, uniformity of operations and management features, as well as standardized access control. The authors draw examples from their experience with such a platform and focus on the key capabilities and functionalities necessary for effective API governance.

Christian Schultz, our next author, takes an in-depth look at how traditional companies — so-called digital immigrants — can use an API-centric approach to achieve significantly faster digital growth and better cope with the challenges of the digital marketplace. He highlights the importance of embracing innovative business practices and suggests companies focus on creating new and convenient customer experiences. Schultz then discusses the typical advantages that an API program offers and follows up with API growth opportunities. He concludes with a look at how an API-centric approach can mitigate certain risks of digital business and ensure that the core team remains intact, thereby maintaining critical know-how within the organization.

In the issue's fourth article, the two of us present a methodology for API management based on the ITIL framework. We describe a 10-step approach that corresponds to the service strategy process in ITIL and is targeted at the creation of a new API program. Our focus is the business side of APIs, starting from high-level business goals and available business assets. We offer both top-down and bottom-up approaches to identifying which particular business assets should be exposed as APIs and in what way. In the article, we show how organizations can make use of business cases to form an initial API business strategy, identify target consumer (developer) groups, and come up with benefits for the consumers who adopt the exposed API. The described framework also includes risk, budget, and ROI assessment and ends with a construction of API marketing and consumer support strategies.

We wrap up the issue with an article by Chuck Hudson, who delves into common pitfalls that occur when Web APIs are used in a mobile environment. He identifies seven typical challenges faced by API providers and consumers, ranging from connectivity and optimization problems to authentication flaws and licensing limitations. Hudson accompanies each challenge with a short description of known solutions for mitigating the issue and gives examples of typical approaches currently used in the industry.

As you get into this issue, remember that the API economy is a new phenomenon that is evolving at a fast pace. Therefore, don't take anything presented in these articles as a strict rule or a certainty. Rather, use the information provided here as general advice gained from the experience of others and apply it in your own context as you see fit. Even though all signs point to the new API economy staying here for a long while, only time can tell where it will lead us. After all, unpredictability is an inherent feature of innovation.

To conclude, we would like to invite you to visit www.apiwisdom.com, our API economy research initiative, which includes an open API for analyzing RESTful APIs.

Giancarlo Succi is a Senior Consultant with Cutter Consortium's Agile Product & Project Management *practice. Dr. Succi is a tenured Professor at the Free University of Bolzano-Bozen (Italy), where he directs the Center for Applied Software Engineering. His research involves multiple areas of software engineering, including the API economy. In the area of Agile, he is particularly interested in empirically evaluating the relationships of methodologies and practices, assessing their impact on quality and productivity, and determining the scope of the application of different Agile methods. Dr. Succi has written more than 300 papers for international journals, books, and conferences, and is the editor of six books and the author of four. He has been the principal investigator for projects valued at more than €7 million and has received more than €10 million in research support from private and public granting bodies. Dr. Succi has been the chair or cochair of several international conferences and workshops, a member of the editorial boards of various international journals, and a leader of international research networks. He can be reached at gsucci@cutter.com.*

Tadas Remencius is a Researcher at the Free University of Bolzano-Bozen. His research interests include Web APIs, empirical software engineering, software and team metrics, teamwork in software development, data visualization and interpretation, and experience management. Mr. Remencius holds a master's degree in computer science from Vilnius University (Lithuania). He can be reached at Tadas.Remencius@unibz.it.

The API Economy: Playing the Devil's Advocate

by Israel Gat, Tadas Remencius, Alberto Sillitti, Giancarlo Succi, and Jelena Vlasenko

A lot of hype surrounds the API economy. Different observers have promised a variety of potential benefits, but which of these promises are true and which should be taken with a grain of salt?

In this article, we take a look at some of the major benefits allegedly produced by the API economy and its key facets. We examine the claims various authors have made and offer counterclaims and potentially opposing views. Our goal is not to criticize or disprove these claims, but to take a wider look at these issues and to provide additional food for thought.

A NEW TYPE OF ECONOMY?

The "API economy" has become a popular buzzword these days, but what does it actually mean? Are we really witnessing a new type of economy?

Many authors make this claim or assume it to be so. However, what are the *new* components that make it a different *type* of economy compared to what we have witnessed before?

Sure enough, we have new technological and social trends, including Web APIs, mobile apps, Internet-enabled devices, cloud computing, social networking, and social commerce. We have a bunch of very diverse and yet compatible API business models,[1] ranging from free to paid (either the developer paying or the developer getting paid) and indirect models (content acquisition, content syndication, etc.). The question is, though, is that really enough to call it a new type of economy? Perhaps we already have something similar now, or had it before, albeit in a different context or with different building blocks?

To examine this question, we need to understand the different actors in the API economy and the main interactions among them. Articles on the API economy mostly focus on two types of actors: API providers (businesses exposing APIs) and API consumers (businesses using or combining APIs exposed by others to provide new services or products). However, there is also a third player — the end users (direct customers of the API consumers). It is also worth remembering that these are not mutually exclusive; the same business can play all three roles when it comes to different APIs.

So in the API economy, we have a model of three main parties, but this in itself is nothing new. There are many economic models that have the same number of key players with the end user (customer) as one of the actors. For example:

- Supplier – manufacturer – customer
- Manufacturer – retail reseller – customer
- Subcontractor – service provider – customer
- Artist – agent – customer
- Developer – publisher – customer

How do these actors relate (or not) to API providers and API consumers? Are there some significant differences that would warrant the label *new type of economy*?

Upon closer examination of the main interactions among these actors (see Table 1), we can notice a couple of patterns when it comes to the three-actor relationship:

- The first actor (i.e., supplier, subcontractor) does not really see end users or care much about them. Its real customer is the second actor (i.e., manufacturer, service provider).
- The second actor (i.e., retail reseller, agent, publisher) acts as a proxy to the customers for the first actor and takes care of some specific area such as marketing, requirements elicitation, or actual sales. In some models, the second actor (e.g., publisher) also gains partial or full rights to the product made by the first actor.

Where does the "API provider – API consumer – customer" trio fit in? One could claim that it follows the first pattern, in which API providers deliver goods to API consumers and they, in turn, produce goods for the end users. However, it gets a bit more complicated than that.

On one side, consumers are indeed the direct customers of API providers, and there are APIs (e.g., those providing utility services) that deliver building blocks that are used by consumers but never reach the end users.

Table 1 — Comparing Actors of Similar Economic Models

Actor	Interactions and Relation to Customers
Supplier – Manufacturer – Customer	
Supplier	Provides basic materials, goods, components, or services to goods manufacturer(s). In fact, manufacturers are the customers. Customers of manufacturers are not of direct concern.
Manufacturer	A customer of supplier(s), dependent on their services to deliver its own goods. Sells goods to users.
Manufacturer – Retail Reseller – Customer	
Manufacturer	Produces goods aimed at users but does not sell those goods directly. Instead it sells them to retail resellers. Needs of users of great concern; marketing is not.
Retail reseller	Sells goods produced by manufacturers to users. Marketing is the main concern.
Subcontractor – Service Provider – Customer	
Subcontractor	Performs certain tasks or provides specific services that are needed by the service provider to deliver its service. The requirements of the service provider are all that matter.
Service Provider	Outsources some of the activities needed to deliver its services to subcontractors. Delivers services to users. Needs of users are key.
Artist – Agent – Customer	
Artist	Produces creative goods for users. Often focused on creativity or the creative process itself and not the target audience. Uses the services of agent(s) to sell and market his or her goods.
Agent	A bridge between artists and customers. Focuses on what is in demand by the audience and on marketing. Guides artist(s) toward the production of popular goods. Often sells the goods through additional actors (e.g., a record label).
Developer – Publisher – Customer	
Developer	Develops products aimed at end users. Gives partial or full control to publisher(s) to market and sell those products to the customers.
Publisher	Markets and sells products produced by developers to customers. Often gets partial or full control of the product rights and is frequently perceived by end users as the producer of the product.

On the other side, there are also APIs that fit neatly in the second model; for example, a provider selling its core products through APIs to markets not reachable directly. Here API consumers act as resellers and simply get a percentage of the sale.

Finally, there are plenty of APIs that are somewhere in the middle or span both categories. There consumers are still the direct customers and the end users are not of great concern to the provider, but unlike suppliers or subcontractors, the value of the exposed APIs is not directed at the consumers themselves but at the end users.

This is the scenario where human innovation kicks in. Providers have some products or services exposed through APIs but have no idea how or in what shape they will reach the end users. It is the consumers who add a creative twist, enable a new way of usage, or make a totally new product or service based on several different APIs and deliver it to the customers.

What makes the API model unique compared to the other models is that API providers are often operating in a sort of black box when it comes to the end users, and yet the latter determine the ultimate success or failure of the API. Furthermore, it is not about creating new APIs (think "new products") as such, but about exposing existing business assets (data, services, products, etc.). Therefore, the relationship with consumers is not driven by requirements or demand (as in the case of suppliers or subcontractors), at least not when it comes to the APIs themselves. Providers expose what they already have, and any tinkering or adaptation to consumers comes only in the form in which the API is exposed (technology, documentation, support, marketing strategy, business model, etc.).

Another unique aspect is the level of uncertainty. While most APIs hold value for the end users, providers have no real way of knowing how their APIs are going to be used in the end. On the one hand, the success of the API ultimately depends on its popularity with end users. On the other hand, that uncertainty makes it difficult for the providers to target the needs of those particular users. This becomes the job of consumers. However, API consumers generally have very little impact on the providers and the content/purpose of the exposed APIs. In the end, we have quite a peculiar situation where the value of the APIs is aimed at the end users, but it is delivered in creative ways by consumers without real control of the content of the API.

So what is the bottom line? Well, it all depends on how you decide to look at it. Our takeaway in this case is:

- Different APIs follow different economic models. Some fit nicely in already known patterns, where either the consumer is the main customer or the consumer simply takes responsibility of some specific activity, such as reselling of the provider's goods. Most APIs, however, are somewhere in between (or rather span both patterns). Whether we call it a totally new model or a new combination of existing models is not really that relevant.
- The API economy relies heavily on human innovation and, by definition, contains a great deal of uncertainty and unpredictability for the API providers. On the flip side, the cost of failure is generally so low and the chances of achieving at least some business benefits (even if not the ones initially anticipated) seems to be so high that, for most businesses, embracing the API economy should be a no-brainer.

HERE TO STAY?

How much of the API economy is hype and how much is real? What are the odds that the API economy is not a temporary phenomenon and will last a while? Can we talk about a stable trend or is it just a transition phase? To answer these questions, we need to look at the roots of the API economy and what is happening right now.

The API economy is said to be the result of several key components,[2-5] namely:

- Web APIs
- The appearance and growth of mobile apps
- Cloud computing
- Social networks and social commerce

Conceptually, Web APIs are like *classical* APIs — typically a library or a framework written in a native programming language (e.g., DirectX, jQuery). They serve as an easy-to-use (relative to solving the task at hand without the said API) interface to certain data, products, or services (or functions, in the case of classical APIs).

From a technical view, Web APIs are no longer tied to specific native programming languages and are typically implemented in a human-readable form using REST-like HTTP calls that result in data in XML or JSON format.[6] They are similar to Web services, except they don't have formal contracts (such as WSDL). In fact, Web APIs are often considered to be either a generalization of Web services or their special case.

Web APIs serve as building blocks that can be easily combined to create new products and services in a fast and inexpensive way. They are the key component that has made the API economy technically possible.

However, Web APIs are just a tool, a means to an end. The main reason why the API economy came to be was the sudden demand for new software applications and solutions caused by the rapid spread of mobile and Internet-enabled devices. The appearance of mobile apps coupled with the new mentality arising from social networks and social commerce was an especially significant trigger that opened up new markets and changed the expectations of end users and dictated their new needs.

Businesses that ventured into the newly born API economy soon discovered that Web APIs bring many more potential benefits than mere crowdsourcing. New customer markets, innovative solutions, and quick time to market are just some of the examples. What we have now is a totally new way for businesses to expand, gain additional revenue, and promote their brand names at relatively low cost — all by simply exposing some of their business assets as APIs. Businesses can also make use of the plentitude of APIs exposed by others to offer new and creative solutions with low risk due to low costs and fast time to market. This is what the API economy is all about — at least at the moment.

We are now at the stage where more and more new solutions aimed at helping companies expose and manage their APIs are appearing.[7] Big players, including governments, are realizing the value of the API economy and starting to make investments in this area.[8-12] The number of public APIs keeps growing rapidly (see Figure 1), even though there is some slowdown this year compared to the previous one (see Figure 2). Many companies are already involved or plan to be involved in the API economy in one way or another. For example, a recent survey by Layer7 Technologies found that 86.5% of respondents will have an API program in place within five years.[13]

Certain aspects of the API economy have seen some significant progress, such as the variety of different API business models. Other areas are still at a very immature state. Leaving technical issues (security, identity management, dealing with large data sources, etc.) aside, the API economy is yet lacking in standardization and predictability (reproducibility) of results.

A critical factor in the success of APIs is the maintenance of healthy and productive relationships with the API consumer base. Trust is the key here. Unfortunately, trust is hard to establish and very easy to lose. From a practical standpoint, trust is expressed through business agreements between providers and consumers, such as service-level agreements (SLAs), licenses, and the like. At the moment, there are no standards in this area. While the current philosophy behind the API economy is presumed to be one of openness, freedom, innovation, and equal rights, API providers can, in principle, significantly damage consumer businesses accidentally or on purpose by suddenly changing SLAs or enforcing monopoly-building licenses.

Right now, nothing really prevents an API provider from having different business models or licenses with different levels of restrictions and applying them on a subjective basis (i.e., giving only restrictive licenses to potential competitors or placing extremely high prices on unrestricted API usage that could potentially damage the provider's monopoly). There is, however, a possibility that the market will regulate itself, and API providers that blatantly abuse their power will be punished by the sudden loss of consumer bases and reputation.

Another interesting point that is not yet clear is what follows next. What is the future of the API economy? What happens when both the provider and consumer markets get saturated?

One likely result for providers will be a change in the motivation for exposing APIs — from potential economic benefits to the need to maintain competitiveness in the market. In other words, an approach that once showed great promise as a competitive differentiator might simply become a necessary way to survive.

From the perspective of API consumers, market saturation will mean that new ideas (innovations) will be much more difficult to come up with; the obvious ones will already have been implemented by someone else. There will always be room for something new, but the creative effort and amount of luck required will be much higher.

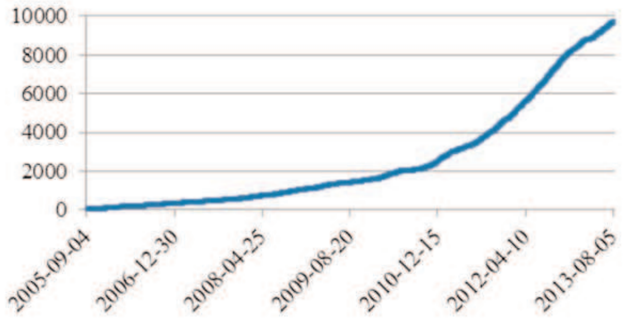


Figure 1 — API growth rate.
(Adapted from www.programmableweb.com.)

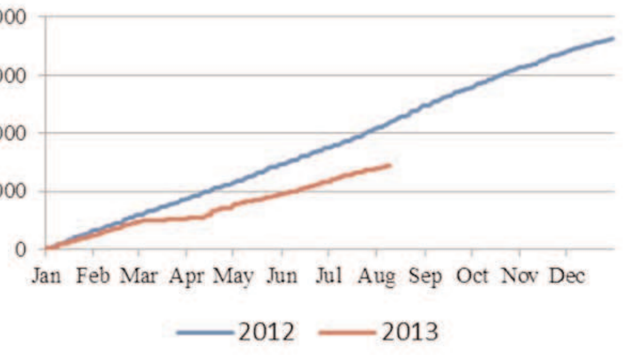


Figure 2 — API growth comparison in years 2012 and 2013.
(Adapted from www.programmableweb.com.)

Of course, this is just one possibility. That's the exciting part of a new economy — we never know where it will take us in the end. What is clear, however, is that now is the time to delve into the API economy if you haven't done so already.

DRIVING INNOVATION?

One of the major positive aspects of the new economy is said to be innovation.[14-17] Industry observers generally claim that innovation is enabled and encouraged by the very nature of the API economy. But is this really so? What kind of innovation are we actually talking about here?

The premise for this claim is that API providers expose APIs that are (relatively) free to be used in any way that API consumers deem to be meaningful. As such, there are a multitude of combinations of different APIs and even more usage scenarios. APIs become basic building blocks, and the end result is limited only by consumer creativity. So human innovation in this context consists of creatively combining APIs and applying them to various scenarios.

Of course, one can argue that this is “fake” innovation. Yes, there is a large sandbox that consumers can play in, but in the end it is still a sandbox. Consumers are constrained by the available APIs and cannot “invent” something that would go beyond those limits. While such sandbox creativity may give inspiration and birth to ideas that go beyond APIs and the API economy, being in a sandbox can also cloud our vision of what lies behind the borders of our playing field. We might struggle to anticipate something that is totally different from what we are used to.

Whichever outlook you prefer, it would probably be more correct to say that the API economy drives human creativity as opposed to real innovation (as in breakthroughs).

LEVELING THE PLAYING FIELD?

Another positive aspect commonly attributed to the API economy is that it acts as a leveling factor,[18, 19] giving similar business opportunities to companies of different sizes and types.

In some respects, that does appear to be true. In particular, one of the key benefits of the API economy is the reduction in costs and time, both in terms of production and time to market. Furthermore, this works both ways. API providers can expose their core business APIs, thereby allowing API consumers to supply different software solutions demanded by end users; API consumers, on the other hand, can make use of various exposed APIs to provide new products and services in an efficient and quick manner. In both cases, the costs and time required to produce these solutions are often reduced to extremely small amounts compared to development from scratch, so businesses are able to try different creative approaches with very low risk (i.e., low cost of failure). The power of the API economy is that the effort of bringing new ideas to fruition becomes so low that it is available to any player, including newly formed startups.

This does not imply, however, that every business starts on the same footing in the API economy or has the same chances of success. When it comes to API consumers, the situation is perhaps more even. The same cannot be said for API providers. The size and reputation of the business can be a huge factor in the success of its exposed APIs. Think of an API exposed by a giant like Google or Facebook versus one exposed by some unknown startup. Which API is going to have wider market appeal?

In the case of industry giants, the marketing of new APIs can almost be left to itself. Startups, in contrast, have to have a good, clear marketing strategy (and possibly a great deal of sheer luck). Furthermore, industry giants have much more room to enforce stricter licenses, SLAs, and less beneficial (to the consumer) business models.

To sum up, the API economy does level the playing field somewhat when it comes to business possibilities and opportunities, but big players still have significant advantages.

FINAL THOUGHTS

It is up to the reader, of course, to draw his or her own conclusions from a “devil’s advocate” article like this one. The whole point is to trigger new thoughts in the reader’s mind through an “on the one hand/ on the other hand” debate.

While we acknowledge the open-ended nature of this kind of article, we would like to conclude by highlighting two particularly intriguing aspects that we see in the API economy:

1. As one of the authors (Israel Gat) has pointed out previously,[20] by enabling programmatic access from the outside to your company’s APIs, you can share and leverage the information assets your company has accumulated over the years. Doing so enables your company to launch new lines of business that amplify, augment, and quite possibly outperform the business your company has traditionally pursued.
2. John Hagel III, John Seely Brown, and Lang Davison of Deloitte LLP’s Center for Edge Innovation[21] have highlighted the great importance of shaping platforms in an era characterized by constant change. In our humble opinion, following the principles of the API economy is a highly effective approach to creating, joining, and shaping platforms. Doing so harnesses today’s technologies to tomorrow’s business designs in the service of your company.

Any economy, of course, is subject to rise and fall over time. The API economy clearly has the potential to rise as a rocket. When, under what conditions, and how it might eventually fall is a subject we will explore in a forthcoming Cutter article.

ENDNOTES

[1]Musser, John. “API Business Models.” Presentation to the *API Strategy & Practice Conference*, New York, NY, USA, February 2013 (www.slideshare.net/jmusser/j-musser-apibizmodels2013).

[2]Medrano, Robert. “Welcome to the API economy.” *Forbes*, 29 August 2012 (www.forbes.com/sites/ciocentral/2012/08/ 29/welcome-to-the-api-economy).

[3]Lane, Kin. "The New API Economy." API Evangelist, 19 January 2011 (http://apievangelist.com/2011/01/19/the-new-api-economy).

[4]Scobie, Corey. "Keynote: The API Economy Is Here: Facebook, Twitter, Netflix and Your IT Enterprise." InfoQ, 8 November 2012 (www.infoq.com/presentations/API-Economy).

[5]Gat, Israel, and Giancarlo Succi. "A Survey of the API Economy." Cutter Consortium Agile Product & Project Management *Executive Update*, Vol. 14, No. 6, 2013.

[6]Gat and Succi (see 5).

[7]Lane, Kin. "API Deployment." API Evangelist (http://deployment.apievangelist.com).

[8]McKendrick, Joe. "Why No One Can Be a 'Passive Consumer' in Today's API Economy." SmartPlanet, 4 August 2013 (www.smartplanet.com/blog/bulletin/why-no-one-can-be-a-8216passive-consumer-in-todays-api-economy/25673).

[9]Burton, Craig. "More Consolidation for the API Economy." KuppingerCole Analysts, 24 April 2013 (http://blogs.kuppingercole.com/burton/2013/04/24/more-consolidation-for-the-api-economy).

[10]Willmott, Steven. "Acquisitions and Investments — Towards a Future of APIs Everywhere." 3Scale, 27 April 2013 (www.3scale.net/2013/04/acquisitions-and-investments-towards-a-future-of-apis).

[11]Williams, Alex. "Facebook and the Sudden Wake Up About the API Economy." TechCrunch, 28 April 2013 (http://techcrunch.com/2013/04/28/facebook-and-the-sudden-wake-up-about-the-api-economy).

[12]"US Government APIs." Data.gov (www.data.gov/developers/page/developer-resources).

[13]"API Security and Management Today." Layer7 Technologies (www.layer7tech.com/infographic).

[14]Medrano (see 2).

[15]Scobie (see 4).

[16]"Staying Ahead in an API Economy." StreamScape Technologies (www.streamscape.com/Technology/scale-and-cloud-api.html).

[17]Olson, Laura. "The Open API Economy: What Is It and How Do I Capitalize On It?" InfoQ, 14 December 2012 (www.infoq.com/presentations/Web-API-Economy?utm_source=infoq&utm_medium=related_content_link&utm_campaign=relatedContent_presentations_clk).

[18]Lane (see 7).

[19]Gat, Israel et al. "Not Technology — But Change Management." Cutter Consortium *Council Opinion*, Vol. 13, No. 2, 2012.

[20]Gat, Israel. "The API Economy: Why and How to Expose Your APIs." Cutter Consortium Workshop.

[21]Hagel, John, III, John Seely Brown, and Lang Davison. "Shaping Strategy in a World of Constant Disruption." *Harvard Business Review*, October 2008.

Israel Gat is a Cutter Consortium Fellow and Director of the Agile Product & Project Management *practice, a Fellow of the Lean Systems Society, and a member of the Trident Capital SaaS advisory board. He is recognized as the architect of the Agile transformation at BMC Software, where, under his leadership, Scrum users increased from zero to 1,000, resulting in nearly three times faster time to market than industry average and 20%-50% improvement in team productivity. Among other accolades for leading this transition, Dr. Gat was presented with an Innovator of the Year Award from* Application Development Trends *in 2006. His executive career spans top technology companies, including IBM, Microsoft, Digital, and EMC. Dr. Gat currently splits his time between consulting and writing, focusing on technical debt, large-scale implementations of Lean software methods, and Agile business service management ("DevOps"). His e-book,* The Concise Executive Guide to Agile*, explains how the three can be tied together to form an effective software governance framework. Dr. Gat holds a PhD in computer science and an MBA. In addition to publishing with Cutter and the IEEE, he posts frequently at* The Agile Executive *and tweets as agile_exec. He can be reached at igat@cutter.com.*

Tadas Remencius is a Researcher at the Free University of Bolzano-Bozen (Italy). His research interests include Web APIs, empirical software engineering, software and team metrics, teamwork in software development, data visualization and interpretation, and experience management. Mr. Remencius holds a master's degree in computer science from Vilnius University (Lithuania). He can be reached at Tadas.Remencius@unibz.it.

Alberto Sillitti is Associate Professor of Computer Science at the Free University of Bolzano-Bozen. He has been involved in several EU-funded projects related to open source software, services architectures, and Agile methods in which he applies noninvasive measurement approaches. Additional research areas include mobile and Web services. Dr. Sillitti has served as a program committee member of several international conferences and as program chair of OSS 2007, XP2010, *and* XP2011. *He is the author of more than 80 papers for international conferences and journals. Dr. Sillitti holds a PhD in electrical and computer engineering from the University of Genoa (Italy). He can be reached at asillitti@unibz.it.*

Giancarlo Succi is a Senior Consultant with Cutter Consortium's Agile Product & Project Management *practice. He is also Professor of Software Engineering and Director of the Center for Applied Software Engineering at the Free University of Bolzano-Bozen. Dr. Succi's research areas include Agile methodologies, open source development, empirical software engineering, software product lines, software reuse, and software engineering over the Internet. He is the author of four books and more than 300 papers published in international conferences proceedings and journals. He can be reached at gsucci@cutter.com.*

Jelena Vlasenko received a bachelor's degree in computer science in 2010 and a master's degree in software engineering in 2012 from the Free University of Bolzano-Bozen. Ms. Vlasenko is currently working toward a PhD. Her current research interests include Agile methodologies, noninvasive software measurement and analysis, software process improvement, and application of natural language processing techniques to documentation analysis. She can be reached at jelena.vlasenko@gmail.com.

Unified API Governance in the New API Economy

by Chandra Krintz and Rich Wolski

MANAGING DIGITAL ASSETS

Digital assets are becoming the value-carrying resources that underlie much of today's economic activity. Increasingly, businesses depend on the ability to produce, manage, trade, and, perhaps most problematically, destroy digital artifacts (software and data) as key components of commercial functionality and profitability. Because these assets exist entirely within computer systems that are interconnected via networks, new techniques for managing them, such as Hadoop, cloud computing,[1] DevOps, and NoSQL, continue to proliferate. At the same time, previously successful software and IT approaches (e.g., service-oriented architecture, Web services, and machine virtualization) are enjoying a renaissance of utility.

Providing software and data as a service — that is, enabling immediate, authenticated, and scalable networked access to digital assets — is critical to the success of any commercial enterprise that possesses them. To facilitate this access, asset owners export assets via an API that both defines and controls what operations can be performed on each asset, by whom, and under what conditions.

APIs also decouple the implementation of this access functionality from the technologies that are used to manage and store the assets. That is to say, while the assets may remain the same, the technologies used to serve and implement them can change, particularly as technological advances reduce implementation costs. APIs must preserve user access to the assets when this occurs. Thus, the lifecycle of the API follows the lifecycle of its assets and *not* the lifecycle of the surrounding technologies, which typically change at a more rapid pace.

Finally, APIs in the modern digital economy must provide standardized network-facing access so that the widest possible variety of applications and devices can access their digital assets. They must also support availability guarantees and fault management strategies associated with the assets and the implementing technologies. It is the combination of standardized, continuously available, networked access that enables a digitally based business to scale.

Thus, APIs provide three functions that are critical for the management of digital assets and artifacts. Namely, they:

1. **Implement control** over the assets, both in terms of operations and access control
2. **Protect the asset lifecycle** from technological changes driven by economics
3. **Enable scale** through standardized, networked connectivity and fault management

Because of these functions, the implementation and management of APIs can be more important than either the digital assets or the technologies that underlie them. For example, consider a company that specializes in website analytics. A change from a NoSQL database to an object store as the implementing technology should be possible without disrupting the business. Thus, the API for the analytics must remain stable while the technologies change. Similarly, the analytics data itself may be changing from day to day. The API for accessing the current data must remain constant, stable, and functional, though, or business will be interrupted.

Despite the primacy of APIs in the new digital economy, however, little technology has yet been developed to implement *API governance* — combined policy, implementation, and deployment control — in a commercial context. Good technologies exist for managing digital assets and for developing both hardware and software necessary to implement digital assets (including the necessary APIs). A few technologies[2, 3] are emerging for packaging and cataloging APIs. Yet technologies for providing stewardship of APIs through all phases of governance are rare.

INTRODUCING APPSCALE

In this article, we describe a strategy for implementing API governance using AppScale, a distributed software platform for managing, unifying, delivering, and composing APIs in a commercial setting. AppScale implements a set of core services that are specifically designed to implement high-level APIs in a consistent, unified way. Using such a platform to implement APIs for commercial digital assets offers several advantages with respect to API governance. In addition to the typical API management features (cataloging, search, deployment support, etc.), AppScale focuses on the following capabilities:

- **Change control.** When API changes are necessary, AppScale restricts how they are implemented so as to control the impact of change on API consumers. If changes need to be rolled back, AppScale returns to previous functionality consistently and completely. It enables this via API usage tracking, versioning, and compatibility checking and enforcement.
- **Consistent policy implementation.** Policies governing the use of digital assets and/or their APIs are implemented consistently across the platform regardless of the constituent technologies that are used to implement the assets themselves. Administrators specify asset properties via a single portal for access control, service levels, lifecycle, backup, and failover, which the platform applies consistently across all assets.
- **Implementation portability.** API implementation is decoupled from the implementation of the digital assets. As technologies evolve or, more problematically, devolve when they sunset, AppScale maintains API integrity by providing an intermediate abstraction layer that allows the implementations to change without impacting API consumers.
- **Monitoring and auditing.** As a platform, AppScale provides a unified fabric for monitoring and auditing API activity. By doing so, AppScale allows enterprises to gather and analyze data in the same way from digital assets that use different implementation strategies and technologies.

AppScale provides these capabilities as part of a freely available and extensible distributed open source platform. As such, AppScale can be used by enterprises for API governance and application deployment without vendor lock-in. We next describe API governance in greater detail and discuss how the AppScale design facilitates such use.

UNIFYING API GOVERNANCE

Increasingly, enterprise applications are taking the form of network-accessible services that export well-defined and access-controlled interfaces. As a result, the development process includes:

- **API development** — the process of designing and coding the software components responsible for implementing the interface
- **Service development** — the process of implementing the application logic
- **Deployment configuration** — the process (often coded as scripts) of coordinating the initiation of all application components when the application is run

Thus the term "application" in this context refers to three separate but interrelated sets of programs that implement the API, service, and deployment.

This decomposition allows the service implementation and deployment components to change w*hile the API remains the same*. In this way, application users maintain consistent, unchanging access to digital assets while the service implementations and underlying infrastructure evolve in response to advances in technology.

As a result of this modularity, the lifecycle for APIs is significantly longer than that of service or deployment implementations. Moreover, from a user perspective, APIs implement policy. Access controls, SLA specification and/or negotiation, fault and error response, and so forth are all presented to users through APIs. Changes to these policies are usually global and long-lived, making their correct implementation critical to the scalable usage of digital assets.

For these reasons, in addition to standard management functions such as installation support, software patching and upgrade, and software dependency resolution, APIs require the implementation of governance — the policies and auditing functions necessary to protect the integrity of the APIs in a unified way. A unified approach to API governance is key to managing applications at scale since the applications and the digital assets they manage are likely to be developed by different entities in a large organization. Indeed, DevOps (an organizational approach that combines development and IT operations) is designed specifically to promote scalable and Agile application development by independent suborganizations. Without unified API governance, however, the scale that this new methodology engenders can lead to a proliferation of incompatible interfaces and wasted or duplicated development effort.

Using a Platform to Ensure Consistency

To ensure consistent control over the APIs in an enterprise, our approach is to build the necessary control functionality into a complete platform that spans all resources and assets. The platform is unique in that it is designed end-to-end so that it monitors, manages, and protects all APIs under its purview in the same way, regardless of the infrastructure or digital assets involved.

Using such a platform, enterprise management is assured that policies governing APIs are implemented globally in a consistent way. This consistency of governance permits independent application development and operation by preventing the possibility that APIs will become suddenly incompatible due to changes or innovation.

To allow the technologies that implement the APIs to change as business or engineering needs warrant, AppScale plugs in multiple competitive alternatives for each service so that enterprises can compare/contrast them and choose the technologies that the local IT organization wishes to exploit.

A PLATFORM FOR UNIFIED GOVERNANCE, DEPLOYMENT, AND MANAGEMENT OF APIs

The AppScale platform[4] is a freely available, open source runtime system for Web, cloud, and mobile applications and the services they use for their implementation. AppScale implements a set of core functions that enable consistent management of the APIs that export access to these services, across the applications and digital assets it hosts. These functions include support for:

- **Plug-in integration** — a set of abstractions interposed between APIs and platform service implementations that facilitate independent and isolated service management
- **Configuration** — a service that all applications use to specify and access their respective configuration information in a consistent way
- **Deployment** — a service that invokes and decommissions APIs and service implementations under administrator control
- **Elasticity and autoscaling** — automatic resource allocation and application scaling according to an external policy, observed runtime load characteristics, and service failures
- **Auditing and monitoring** — consistent provenance for the APIs, service implementations, and digital assets managed by the platform

The AppScale platform combines these functions within a distributed system that is packaged as a virtual machine (VM) image. Platform administrators deploy AppScale via a toolset that constructs the platform as a collection of VM instances over any cluster system that supports virtualization, including public and private cloud infrastructures as well as on-premises and managed data centers. The combination of unified automated services for managing APIs separately from service implementations, the scale realized by AppScale's distributed architecture, and its portability across scalable data center technologies make it an ideal engine for implementing API governance.

Example: API Governance and Google App Engine

To illustrate how AppScale implements governance, we now describe its support for Google App Engine (GAE). In particular, AppScale exports (mirrors) the publically available APIs of GAE so that developers can deploy any GAE application either on the GAE platform over Google's resources or on the AppScale platform on-premises, without modifying their applications. To enable this, AppScale leverages plug-in integration to link each API to an open source implementation of each service. Between each API-service pair, AppScale implements a software abstraction that maps API calls to the interface of the service implementation.

To allow the technologies that implement the APIs to change as business or engineering needs warrant, AppScale plugs in multiple competitive alternatives for each service so that enterprises can compare/contrast them and choose the technologies that the local IT organization wishes to exploit, without impacting the digital assets they deliver. If, for example, an enterprise DevOps team uses the Apache Cassandra NoSQL data store, AppScale implements the GAE abstractions using Cassandra as a back end and the GAE API code as a front end. With AppScale, the applications no longer dictate the underlying technologies that must be used, allowing the IT organization to govern its infrastructure without concern for application modification. Further, if the team decides to adopt a different storage infrastructure, AppScale simply plugs in the new technology without changing the APIs the applications use to access it.

Because the API code and back-end software technologies are integrated by the distributed AppScale platform, they can be instrumented and monitored in a uniform way. If one or more of the software modules is/are modified, AppScale can track and report on these modifications. AppScale also supports automatic deployment of these technologies so that new code is introduced in a controlled manner and can be rolled out or rolled back in a way that is both auditable and scalable.

Since AppScale itself is portable to a variety of public cloud and on-premises software environments, it is possible to run AppScale in Google Compute Engine (GCE), Amazon's AWS, and Eucalyptus.[5] GAE applications then migrate between GAE, AppScale over GCE, AppScale over AWS, and AppScale on-premises over Eucalyptus. This deployment portability using a single, consistent platform allows IT to develop a wide variety of disaster recovery and cost management policies without the need to modify the applications.

Finally, APIs do need to change from time to time. However, it is often necessary to support applications that use the "old" API as a legacy. Because AppScale runs under the control of IT or DevOps, it will run whatever version of the API the local organization requires. Thus the organization controls the lifecycle of the APIs it uses through its business logic and not the lifecycle determined by a third-party service provider.

USE CASES

We next describe two common use cases that examine key aspects of platform-based API governance using AppScale: uniform policy implementation and implementation portability. Both cases rely heavily upon monitoring and decoupling of digital asset access via APIs from the software technologies that facilitate their delivery.

- **Uniform policy implementation.** Platform administrators can use AppScale to specify a set of policies to enforce across assets. Our most common use case employs this feature to provide uniform backup of data assets and automatic failover for the services that implement them. Administrators can specify a range of properties for data assets, including how many redundant copies to store, where to store them (locally, remotely, in any number of different public or private cloud systems, etc.), and the type of consistency that should be employed across copies. For executing services, administrators identify those that are fault tolerant and specify properties such as failover target(s) (i.e., what alternative implementations to use when a failure occurs).
- **Implementation portability.** AppScale can be used to enable businesses to avoid lock-in — the overhead associated with rewriting software in order to use alternatives to constituent software components. The implementation portability of the AppScale platform precludes lock-in in two ways. First, since the platform executes on a wide variety of deployment targets (public, private, and managed clouds, clusters, and data centers), AppScale brings cross-cloud portability to applications and services that execute over it. Second, because AppScale decouples APIs and assets from the technology that facilitates their export, administrators can easily employ different alternatives — without changing the API, the application code that uses the API, or the underlying digital assets — by selecting between them during platform deployment.

AppScale significantly simplifies API governance for these two use cases by managing the complex distributed technologies that underlie important enterprise digital asset functions and features (fault tolerance and disaster recovery) and by allowing implementation technologies to change without impacting asset access (precluding lock-in).[6-8] Moreover, AppScale provides users with a uniform way of specifying, monitoring, and customizing this functionality across assets so that developers can focus on innovation and digitally based businesses can scale their digital asset offerings.

CONCLUSION

APIs have emerged as a key component of the modern digital economy. The reason for this is that they provide access to software and data in a standardized way that is easily consumed by humans and software over a network. The aspects of APIs that are critical for their successful use by enterprises include standardized access control, protection of asset lifecycles against technological changes in their implementation ecosystem, and uniform operations and management for platform-wide features such as elasticity, availability, and fault tolerance.

Advanced cloud platforms can facilitate API governance by decoupling digital assets and their APIs from the technologies used to deliver them. The abstraction layer that enables this decoupling allows the creation of software systems that implement a set of core services that can be reused across a wide range of digital assets. AppScale is one such distributed software platform for managing, unifying, delivering, and composing APIs in a commercial setting. Additional information on the open source AppScale cloud platform can be found at www.appscale.com.

ENDNOTES

[1]Armbrust, Michael, Armando Fox, Rean Griffith, Anthony Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ian Stoica, and Matei Zaharia. "A View of Cloud Computing." *Communications of the ACM*, Vol. 53, No. 4, April 2010.

[2]Layer7 Technologies (www.layer7tech.com).

[3]Mashery (www.mashery.com).

[4]Krintz, Chandra. "The AppScale Cloud Platform: Enabling Portable, Scalable Web Application Deployment." *Internet Computing*, Vol. 17, No. 2, March-April 2013.

[5]Nurmi, Daniel, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, and Dmitrii Zagorodnov. "The Eucalyptus Open-Source Cloud-Computing System." Paper presented to the *International Symposium on Cluster Computing and the Grid (CCGRID '09)*, Shanghai, China, May 2009.

[6]Chris Bunch, Vaibhav Arora, Navraj Chohan, Chandra Krintz, Shashank Hedge, and Ankit Srivastava. "A Pluggable Autoscaling Service for Open Cloud PAAS Systems." Paper presented to the *IEEE Fifth International Conference on Utility and Cloud Computing*, Chicago, Illinois, USA, November 2012.

[7]Chohan, Navraj, Anand Gupta, Chris Bunch, Kowshik Prakasam, and Chandra Krintz. "Hybrid Cloud Support for Large Scale Analytics and Web Processing." Paper presented to the *3rd USENIX Conference on Web Application Development (WebApps '12)*, Boston, Massachusetts, USA, June 2012.

[8]Chohan, Navraj, Anand Gupta, Chris Bunch, Sujay Sundaram, and Chandra Krintz. "North by Northwest: Infrastructure Agnostic and Datastore Agnostic Live Migration of Private Cloud Platforms." Paper presented to the *4th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud '12)*. Boston, Massachusetts, USA, June 2012.

Chandra Krintz is a Professor of Computer Science at the University of California, Santa Barbara (UCSB) and cofounder of AppScale Systems, Inc. She joined the UCSB faculty in 2001 after receiving her MS and PhD degrees in computer science from the University of California, San Diego (UCSD). Dr. Krintz has mentored over 60 undergraduate and graduate students, has published numerous research articles, participates in efforts to broaden participation in computing, and is the progenitor of the AppScale project. She can be reached at ckrintz@appscale.com.

Rich Wolski is a Professor of Computer Science at UCSB and cofounder of Eucalyptus Systems, Inc. Having received his MS and PhD degrees from the University of California at Davis (while a research scientist at Lawrence Livermore National Laboratory), he has also held positions at UCSD, the University of Tennessee, the San Diego Supercomputer Center, and Lawrence Berkeley National Laboratory. Dr. Wolski has led several national-scale research efforts in the area of distributed systems and is the progenitor of the Eucalyptus open source cloud project. He can be reached at rich@cs.ucsb.edu.

How APIs Can Reboot Commerce Companies

by Christian Schultz

Data feeds and APIs have been key elements of the commerce and retail business chain almost since the start of the commercial Internet. In most cases, they have been used to run affiliate ads or to generate a secondary revenue stream through the provision of white-label services for partner companies (i.e., services provided by one company but given the look and feel of another company's website). But what is the opportunity of an API-centric approach for traditional companies — the so-called "digital immigrants"? Does it help them cope with the challenges of the digital marketplace, and does it make it possible for them to gain a bigger share of the digital growth? Why is this a better way to manage the risk of digital investment?

DIGITAL IMMIGRANTS HAVE PROBLEMS PARTICIPATING IN FAST GROWTH

Most digital immigrants coming into the online business world use a modification of their business model from the physical world. That way the new venture is just an extension of their existing business and can be financed from operating profit. But this approach may lead to poor profit performance and a lot of inefficiency; furthermore, it may not allow companies to take full advantage of innovative market opportunities.[1] Typical advantages of the Internet, such as network effects and global presence, are rarely exploited.

In the physical world, the organization and presentation of the store make up a major part of a retailer's success. Maybe this explains why, when going digital, so many companies focus on the implementation of their Web storefront.[2] But the merchant's website is the *last* step of a long customer journey, and it contributes little to a sales conversion. The factors that contribute most to sales are marketing campaigns and traffic acquisition from Web content publishers. So it's all about figuring out complex marketing ecosystems such as search/affiliation/performance-based advertising and finding a way to the desired ROI.

In this field, aggregators and pure players like eBay or Amazon dominate the market. Therefore many companies believe they need to implement an integrative digital approach with extensive optimizations of all relevant marketing channels and aspects of the customer relationship. According to board-level observers, omnichannel retail represents "a revolutionary change which will profoundly alter the future of retail, and require significant reengineering of the retail business operating model."[3] Yet this "mantra" of optimization — as Alexander Graf from German e-commerce consultancy Etribes explains — is expensive and takes a long time to implement. If such an approach does not reach the highest possible level in terms of conversion rates and other important commerce KPIs for some companies, it may become a dead end. And even those who *are* able to make it work are in danger of neglecting alternative business models with higher growth potential.[4]

Internet startups, in contrast, form their business around the idea of providing new customer experiences — an approach that enables them to discover and make use of emerging trends (something traditional companies often do quite poorly). Following the principle of serving customer desires, the commerce experience increasingly takes place by embedding product data into non-commerce websites and apps. Examples described by e-commerce strategy consultant Jasper Bell include referrers such as Polyvore, Svpply, and Pinterest, which "have affected [sic] a massive swing toward product-pinning and scrapbooking."[5] In this way, merchants can influence the customer's journey early on, when he is still looking for orientation rather than actively investigating products. This might even create impulse buying opportunities for direct transactions.

Advertising campaigns, in contrast, are more successful with the "last click" than in influencing customers in the early stages of the decision-making process.[6] This is because the publishers involved early on, such as blogs and magazines, don't earn enough money from transactional advertising. Although they do an important job by providing content that users need to make their decision about a product, the last click before they conduct the transaction rarely happens on the publishers' sites. Instead it takes place on a limited number of specialized sites with price comparisons or discount offers. And even if those publishers that do the early work were to get an equal portion of the reward that merchants pay for a successful sale, the user's journey would be basically the same; she would execute the transaction for the lowest price or for the highest discount. The opportunity to let the customer spend her money more conveniently has been missed. Retailers and brands should try to reach the customer in a different context. For instance, someone might feel ready to spend more money in a social media environment or when they are able to pick up the product locally.

Relying on advertising campaigns alone can create a price-dumping environment that most commerce companies should avoid. Creating convenient customer experiences leads to a higher value perception of products and, according to SV Angel founder David Lee, it should be regarded as the key for operating leverage in commerce.[7] Right now we are seeing a great variety of new commercial models. Some of these models — and the assets they make available through APIs — are as follows:

- **Subscription commerce.** Services and products can be provided on a subscription basis, such as monthly shipments of artisan foods or baby products. *Assets exposed:* all functionalities needed to enable customers to perform a transaction.
- **Social commerce.** Social commerce involves the use of social network(s) to facilitate the online purchase or sale of products and services. *Assets exposed:* mechanisms for connecting products and services to social content (e.g., recommendations and referrals).
- **Location-based commerce.** When a consumer enters a certain location, he receives information on his mobile phone about products and services available in that area. *Assets exposed:* local product catalog, geographical information about the nearest store.
- **In-app purchases.** Widely known from game apps that allow users to pay for extra features during play, this mechanism can be applied to other sorts of apps. *Assets exposed:* product recommendations, connection to the payment system of the app marketplace.
- **Shoppertainment.** Merchants try to engage customers through entertaining retail experiences, such as the digital showrooms of car manufacturers. *Assets exposed:* video and rich media content.
- **Collaborative consumption.** Wikipedia describes this as a C2C-based model "in which participants share access to products or services." One example is the peer-to-peer accommodations service, Airbnb. *Assets exposed:* user profiles, payment function.

Statistics from Germany show that the online market share of pure digital merchants is growing, while that of traditional players is declining despite continual investments.[8] So those who want to be in the online commerce fast lane should go the way of innovation and better adaptation to customer desires.

ADVANTAGES OF AN API-CENTRIC ARCHITECTURE

An API platform connects organizations, developers, and users. Partners use API-provided content and functionalities to develop their own websites, apps, and/or tools. Based on that, users identify and share popular services. This kind of connected architecture enables companies to touch base with users on a variety of websites, apps, and devices (e.g., smartphones, tablets, Xbox). It is more than just submitting a product catalog to another system in order to acquire traffic from it. The API principle is user-centric and works by letting customers discover commerce content within the apps and things they want to use. An API architecture is a forward-thinking design suited for offering great customer experiences.

Content and services are the digital assets and the core of any business. Relationships from "node to node" mean that partners co-create a service, and each partner does what it does best while thinking of and planning for the resulting value chain. So a retailer or manufacturer takes care of its data and tries to enable different use cases that ultimately serve customer desires.

An API-centric business can be a lever for innovation. A broad portfolio of partner services and the company's own existing services opens many fields of experimentation. Such an environment causes a significant increase in the R&D speed, which in turn drives the costs of experimentation down. An API provider can achieve a high degree of innovation while being freed from managing highly fluid app development lifecycles.[9]

Other resulting effects of an API-centric architecture are:

- **New fields of growth.** Certain partners might arise as sources for large numbers of transactions. This effect is known, for example, in the micro cosmos of price comparison engines when they started to provide some websites with data that had previously been arbitraging only eBay product content. With additional product results, these arbitragers have been enabled to place bids on Google Adwords that are more profitable than before, and so they generate high traffic volumes.[10]

- **Reaching new international or global customers.** Unlike physical products, content and functionality are not limited to geography. A retailer, for example, might develop a catalog framework with smart features such as filtering options that exactly fit the product segment. Now this company might provide local retailers with the catalog system. These partners take over the physical distribution of products in each market, and they pay a small revenue share in return for the service. In this way, a business can expand to new markets rapidly without shipping the products itself (although it may take over the physical distribution of products later).
- **Getting into local commerce.** Enabling customers to go from digital to physical retail or vice versa makes existing physical infrastructure a competitive advantage (bricks and clicks business models).[11]
- **Independence from big channels.** More innovative positioning and new fields of growth make the conventional channels in which big aggregators set the rules less important. This allows companies to skip some of the poor-performing campaigns for certain products that they previously paid for just to achieve critical volumes.

An API-centric approach allows a company to separate its back end from front-end operations. Also, self-operated media properties and partner media properties can be managed in distinct ways. This separate viewing brings increased internal flexibility. A company might decide, for instance, to develop its own apps for some devices and allow third parties to develop apps for other devices as they want to. With more partners, it becomes easy to shift the focus to successful use cases, which makes back-end development more agile.

API GROWTH OPPORTUNITIES

According to e-commerce software provider Elasticpath, more than 250 shopping or payment APIs are currently available to developers. They are exposing "core content and commerce functions — such as customer information, merchandising systems, order history, search tools, and product catalogs."[12]

Companies can take on the role of API provider, consumer, or both. It depends what use cases can be associated with the existing digital or physical assets and what kind of customer experiences companies want to create.

APIs for Direct Transactions

For many years, merchants have provided their product data to aggregators and affiliate networks. While price comparison engines just link to the merchant website, on eBay and Amazon a merchant doesn't even need a store front end. Their APIs connect directly to the merchant's commerce system and update related information automatically. In this case, a merchant is just trading with raw data. The problem is that aggregators and networks address the last stage of the customer journey, so merchants may end up paying advertising fees for very low margins.

However, brands and manufacturers could act as merchants directly on eBay and Amazon. In this way, they can establish direct customer relationships (e.g., in new countries without investing in their own websites and related marketing efforts).

Providing content and functionality to developers leads to greater exposure in content sites and apps. This exposure will especially increase value to the merchant if it acts as an influencing mechanism early in the customer journey. The case of the German website PC-Welt shows how implementing price comparison results in context with professional product reviews supports the decision of the customer in her subsequent journey.[13] This approach greatly influences what product the customer will buy and which source she will buy it from. At present, the conventional ad business model does not reward content publishers appropriately. An API-centric approach offers brands and merchants the ability to provide content and rich functionality to those websites and more appropriate methods with which to reward them.

As an example of rich functionality, the Spreadshirt API provides developers with T-shirt design features that allow users to customize T-shirts on a blog or inside an app and then purchase it from Spreadshirt.[14] Developers can pre-design T-shirts in the context of their website or app, which adds important value to the user experience.

Distribute Core Functionalities Internally

A multidevice approach can lead to a lot of complexity if a service must be developed for each device separately. The iPad app might not provide the same user experience as the native app on a PC and so on. As users increasingly expect the same quality standards on all devices, it becomes necessary to harmonize the development of all projects. Otherwise frustrated users generate higher customer service costs and churn. A central API makes sure that core functionalities are provided to all devices, a key to achieving customer satisfaction.

Integrate External Data and Functionalities for Enhancement

Some commerce players might stand on the threshold of becoming a feed aggregator by providing a marketplace API that allows other merchants to integrate their product catalog. In this way, Zalando — a German merchant that started as a copy of Zappos — quickly increased the number of products offered on its website, especially in new categories.[15] Now the company can better exploit its marketing power and existing traffic on its site.

Using the Pinterest API, a website can enable users to pin pictures from it, which then appear in the social network. Early experiences show that this way of spreading data into social media might arise as a new source for transactions for fashion commerce.[16]

The rule is that changing the customer experience requires more or less adaptation of the business model. A company might add new data to its API and enhance the existing range of its product catalog or just pass that data to partners. An interesting field is the implementation of payment options and virtual currencies. Functionalities of reward programs can be matched to product inventory so that users can choose to pay with earned bonus points — as, for example, the partnership of Best Buy and Citibank shows. Customers can use a mobile app to earn Citibank rewards and spend them on Best Buy products.[17]

Connect to Real-World Services

Walgreens has launched a project that is typical of how access to existing physical infrastructure can be leveraged by third parties. The "Quick Prints" API can provide functionality to a whole range of use cases such as apps for cameras, photo editing, funny greetings, and many more. Customers can order pictures directly from the app and pick them up at the nearest Walgreens retail location. The drugstore profits from revenue shares and cross-selling within the physical store.[18]

Adapting to a new business model may result in role changes, such as moving from trade to commerce, from retail to rich functionality, or from manufacturing to retail. Using the advantages of physical infrastructure might be one of the biggest opportunities to consider. The focus of using APIs should largely be on creating customer experiences and satisfaction rather than on marketing campaigns.

MANAGING THE RISK OF DIGITAL BUSINESS

As the need for investments in digital units and legacy system upgrades rises, spending on marketing and digital hires increases — after all, a failure of the digital activities might cause severe damage to the business as a whole. Distribution from an open platform can reduce risk through diversification. If a company's own media properties (e.g., apps, websites) fail, other revenue streams can still flow and later compensate for the lost properties.

However, the company that is exposing its APIs must try to ensure that it reaches a high number of third parties and distributes traffic equally. A portfolio that consists of few dominant key partners is in danger of yielding declining revenue shares because the partners will use their negotiation power to lower their payments to the API provider unless the provider offers other benefits. A portfolio that consists of a few dominant key partners is in danger of declining revenue shares because the partners will use their negotiation power unless the API provider offers other benefits. In the opposite case of many long tail partners, the cost for the relationship in terms of acquisition, administration, and billing might become too high. If a company goes global, support requirements for languages and payment types can cause a lot of problems. Two of the most serious problems of working with many third parties is the difficulty in predicting business volumes and the provider's lack of influence on the decisions of the partner. It can take a long time before partners connect and go live with a service, and traffic may not really ramp up sufficiently.

From an API-centric perspective, product management and development take place on a API platform level. A company's own websites and apps can be seen as separate projects with dedicated designers and marketing specialists. In the case of failure, there is often a danger of team disintegration. Important talent moves on, so it might become hard to recover from a crisis. With an API-centric approach, the platform team remains intact, and the know-how stays in the organization. Also, the brand can continue to exist for the digital customer.

CONCLUSION

Having already dived deep into digital marketing, it may now be the right time for retailers and manufacturers to reassess their business for further growth capabilities and investment risk. The most successful Internet companies are based on innovative models, so it just makes sense to find ways of participating as much as possible in new fields of innovation. By constantly discovering new sites, apps, and ways of communicating, users decide which models are successful. The API-centric approach allows companies to distribute their digital assets in a widespread way and encourage developers and partners to create a diversity of models, some of which will doubtless become popular. In this way, the business grows with high speed. This approach allows digital immigrants to keep pace with rapid innovations and change their role to enablers of great customer experiences. The risk of innovation remains, but diversification allows companies to manage it. A well-managed value network might even offset the suboptimal performance of a company's own websites and apps and thus offer an escape from the conventional mantra of optimization.

ENDNOTES

[1]Randall, Greg. "12 Common Mistakes to Avoid for Offline Retailers Moving into E-Commerce." Econsultancy, 30 May 2013 (http://econsultancy.com/de/blog/62810-12-common-mistakes-to-avoid-for-offline-retailers-moving-into-ecommerce).

[2]The German retail industry institute EHI has done research on 33 major German retail companies showing that 49.8% of their digital marketing budget goes to the companies' own websites. See: "Verteilung des Online-Marketing-Budgets im Handel im Jahr 2011 nach Einsatzbereichen." EHI Marketingmonitor, 2011 (www.handelsdaten.de/statistik/daten/studie/208013/umfrage/verteilung-des-online-marketing-budgets-im-handel-im-jahr-2011-nach-einsatzbereichen).

[3]"Retailers Investing £5bn in Move to Omni-Channel." SupplyChainStandard.com, 28 May 2013 (www.supplychainstandard.com/Articles/4499/Retailers+investing+5bn+in+move+to+omni-channel.html).

[4]Graf, Alexander. "Der Heinemann Kegel." Etribes Framework, 13 March 2013 (http://de.slideshare.net/herrgraf/der-heinemann-kegel).

[5]Bell, Jasper. "E-commerce Is Dead, Long Live Distributed Commerce." Econsultancy, 31 July 2012 (http://econsultancy.com/de/blog/10451-e-commerce-is-dead-long-live-distributed-commerce).

[6]Butcher, Mike. "Study: Apps and Content Startups Miss Out Because Affiliate Model Is Broken." Techcrunch, 25 July 2013 (http://techcrunch.com/2013/07/25/study-apps-and-content-startups-miss-out-because-affiliate-model-is-broken).

[7]Lee, David. "Reading Jeff Bezos." David Lee (blog), 26 October 2012 (http://daslee.me/reading-jeff-bezos).

[8]Ottersbach, Thomas. "Verdrängungswettbewerb: Internet Pure Player Weiterhin Vorn." ecommerce-vision.de, 30 July 2013 (www.ecommerce-vision.de/07-2013/verdraengungswettbewerb-internet-pure-player-weiterhin-vorn).

[9]Liu, Jay. "The Lifecycle Cost of Developing and Managing Mobile Apps." *BrightIdeas*, 20 January 2012 (http://blog.brightcove.com/en/2012/01/lifecycle-cost-developing-and-managing-mobile-apps).

[10]From my own experience in arbitraging websites at Pangora from 2008 to 2010.

[11]Bell, Jasper. "Now Is the Time to Embrace Showrooming." Econsultancy, 13 May 2013 (http://econsultancy.com/de/blog/62712-now-is-the-time-to-embrace-showrooming).

[12]Bustos, Linda. "The Rise of Shopping APIs" (infographic). GetElastic, 18 July 2012 (www.getelastic.com/the-rise-of-shopping-apis-infographic).

[13]From my own experience and measured conversion rates with Partner IDG at Pangora 2010.

[14]Breest, Martin. "Introduction to Spreadshirt's Platform Architecture for Providing Customized Apparel as a Service." Spreadshirt, 19 May 2011 (http://de.slideshare.net/Spreadshirt/it2-halle-introductiontoplatformarchitectureforprovidingcustomizedapparelasaservice).

[15]In an interview, David Schneider, founder of the German online store Zalando, explains how his company integrates partner feeds. See: "Zalando Verrät Details zum Marktplatzkonzept Wir Integrieren Wöchentlich Neue Partner." Internetworld Business, 29 February 2012 (www.internetworld.de/Nachrichten/E-Commerce/Handel/Zalando-verraet-Details-zum-Marktplatzkonzept-Wir-integrieren-woechentlich-neue-Partner).

[16]Kuang, Cliff. "Infographic: The Astounding Power of Pinterest." *Fast Company*, 10 September 2012 (www.fastcodesign.com/1670750/infographic-the-astounding-power-of-pinterest#1).

[17]Perez, Sarah. "Citi and Best Buy Launch Mobile Rewards App." Techcrunch, 18 August 2011 (http://techcrunch.com/2011/08/18/citi-and-best-buy-launch-mobile-rewards-app).

[18]Koetsier, John. "The Apocalypse Is Upon Us: Walgreens Has an API and SDK." Venturebeat, 10 July 2012 (http://venturebeat.com/2012/07/10/walgreens-api-sdk).

Christian Schultz is founder and Managing Director of desiremetrics consulting (desiremetrics.com), a consulting firm that provides business concepts and ideas from an API-centric perspective. desiremetrics helps its client companies build platforms that address their partners, enabling them to use the companies' data, mesh that data, and possibly figure out the desires of the end user better than the clients themselves could do. Mr. Schultz has over 12 years of business development experience in the field of digital traffic generation. He has worked in major telecommunication and Internet companies such as Nokia, Excite, and AOL. In his previous role at a major European shopping engine — formerly Pangora — he developed an API business program that provides data to a broad portfolio of publishers of websites and apps. Successful growth made the API program the company's core business. He can be reached at chris@desiremetrics.com.

Tailoring ITIL for the Management of APIs

by Tadas Remencius and Giancarlo Succi

We are currently witnessing the rise of a new economy — the API economy — in which businesses are able to gain significant business value by exposing (parts of) their business assets as APIs. It is a new phenomenon that enables innovation by third parties and levels the playing field for businesses of different sizes and degrees of influence. It allows new products and services to be delivered in extremely short time frames and companies to reach markets and users that would normally be too costly or impossible to tap into otherwise.

API economy. An economy in which companies expose their (internal) business assets or services in the form of (Web) APIs to third parties with the goal of unlocking additional business value.

As much as the API economy is exciting and promising, it is also full of unknown and uncertain factors. We do not yet have a clear methodology for reliably achieving its promises and potential benefits. Simply exposing an API does not guarantee positive results. We need an effective way of getting the best out of the new economy.

There are already a number of API management platforms out there,[1] but it will likely take some time for a mature and experience-proven methodology or framework to appear. However, we do not need to start from scratch. We can take as a basis an existing industry best practices framework such as ITIL (Information Technology Infrastructure Library),[2] which is a widely used framework for IT service management and covers the whole lifecycle of a service. It has a strong focus on business value and on alignment with business strategy and business objectives. In fact, if you replace the notion of service in ITIL with the notion of API in the API economy, you will find that the framework fits remarkably well with the management of APIs.

API. A software interface that exposes certain business assets to other parties (systems and/or people).

This application is not totally straightforward, though. The API economy has its own particularities and specifics that we will want to take into account.

In this article, we propose an adaptation of the ITIL framework to suit APIs. Whenever possible, we make use of information from and experiences of businesses that are already taking part in the API economy. The methodology we describe here corresponds to the service strategy process of ITIL and should not be taken as a set of mandatory rules. Instead, we look at it as a practical starting point for businesses that want to expose their business assets as APIs.

ADAPTING ITIL TO API MANAGEMENT: THE SERVICE STRATEGY PROCESS

The service strategy process in ITIL includes five activities:

1. Strategy Management
2. Service Portfolio Management
3. Financial Management
4. Demand Management
5. Business Relationship Management

Overall, all of them fit nicely within the context of API management. The key difference, however, is that ITIL takes the business side as a sort of "black box" input and makes no judgment on how good or fitting the supplied business goals and strategy are. Its purpose is to deliver IT services that are optimized to the given business needs, whatever they may be. But in the case of API management, business aspects are an integral and critical part, and we have to consider them in our methodology. In this regard, we start from a higher level (i.e., business layer) than ITIL does.

API STRATEGY MANAGEMENT ACTIVITY

Step 1 — Define API Business Goals

As in ITIL, we want to use existing business strategy as a general guideline. We also need to come up with a business strategy specific to our API(s). The first step in doing that is to identify API business goals. (See sidebar "Sample Scenario: Business Goals → Business Cases → APIs.")

SAMPLE SCENARIO: BUSINESS GOALS → BUSINESS CASES → APIS

"Business case" is quite an abstract term and in practice might refer to very different things in terms of level of detail and formal structure. In this context, the important part is the *purpose* of business cases: to come up with business scenarios that are likely to result in the accomplishment of the desired business goals through the exposure of selected APIs.

Let's say the business in question is a professional sports club that is neither among the top clubs nor very popular in its geographical area. Its main source of revenue comes from sponsors/advertising. A smaller part comes from ticket and merchandising sales.

BUSINESS GOALS

The club decides to see if the API economy can help it grow. The management takes a top-down approach and starts with the following three business goals:

- (**G1**) Grow fan base
- (**G2**) Make existing fan base more active and involved in terms of attending games and purchasing merchandise
- (**G3**) Expand brand name visibility outside the fan base

ASSETS

After exploring the local market and business environment, the club's management looks at what could be exposed as APIs and how those APIs could lead to the accomplishment of business goals. The identified assets include:

- (**A1**) Games (main product of the business)
- (**A2**) Merchandising (secondary product)
- (**A3**) Game results and various statistics, such as play-by-play and individual player metrics (available data)
- (**A4**) Expert knowledge and valued opinions (those of the coaching staff and star players)

INTERNAL BUSINESS CASES

The management comes up with several *internal* business cases that identify three different APIs:

IBC-1

- **Information and ticket API:** Allows users to retrieve up-to-date information about upcoming games and other public events organized by the club and to reserve and buy game tickets
- **Business objectives:** (**BO1 ← G1, G3**) Promote club-related activities, (**BO2 ← G2**) increase game attendance
- **Key metrics:** Ticket sales, number of tickets sold through the API, number of Web referrals to the club website through the API
- **Involved business assets: A1**
- **Target user groups and/or ecosystems:** Users of social networks and various sports-related blogs and forums, businesses that provide guides to upcoming city events
- **Expected outcomes:** Increase of ticket sales and game attendance, increased brand name recognition

IBC-2

- **Merchandising API:** Allows users to see and buy available merchandise
- **Business objectives:** (**BO3 ← G2**) Increase merchandise sales
- **Key metrics:** Merchandise sales, merchandise sales through the API
- **Involved business assets: A2**
- **Target user groups and/or ecosystems:** Users of social networks and various sports-related blogs and forums
- **Expected outcomes:** Increased merchandise sales, increased brand name recognition

IBC-3

- **Feedback API:** Allows users to query existing fan feedback and question database and to submit new entries
- **Business objectives:** (**BO4 ← G2**) Increase fan involvement, (**BO5 ← G1, G2, G3**) information acquisition
- **Key metrics:** API spread (number of API consumers), API traffic (amount of feedback submitted by fans)
- **Involved business assets: A4**
- **Target user groups and/or ecosystems:** Users of social networks and various sports-related blogs and forums
- **Expected outcomes: I**ncreased fan involvement and fan base growth, acquisition of useful information from fans (e.g., about desired complementary entertainment during games, preferred type of merchandise)

The key outcomes of these cases are the actual APIs that are going to be exposed and target API consumers.

SAMPLE SCENARIO: BUSINESS GOALS → BUSINESS CASES → APIS (cont'd)

EXTERNAL BUSINESS CASES

The management of the club further refines the API business strategy by constructing *external* business cases that show why target API consumers would use these APIs:

EBC-1

- **API:** Information and ticket API
- **Scenarios:**

  (a) A user of a forum/social networking site/blog posts a message about an upcoming game using the API. The message includes the possibility of reserving or buying tickets for that game.

  (b) An online city event guide business includes upcoming club games using the API.
- **Benefits (API consumer):** Information about upcoming club events, percentage of ticket sales

EBC-2

- **API:** Merchandising API
- **Scenarios:** Website/blog owner uses the API to show and sell club merchandise. The owner can specify his or her own text and/or small logo that will be included on predefined spots on the selected merchandise (e.g., additional logo on the back of the jersey) sold through his or her site.
- **Benefits (API consumer):** Percentage of merchandise sales, possibility to promote own brand/website

EBC-3

- **API:** Feedback API
- **Scenarios:**

  (a) Website/blog owner or forum/social networking site user submits a desired question or request to the coaching staff, team management, or specific athlete and uses the API to display that submission and allow other users to vote it up. The submissions that get the most votes are answered every week and posted on the official website of the club, citing the author of the question/comment and its source (website hosting the API).

  (b) Website/blog owner uses the API to show the most recent fan questions and corresponding answers.
- **Benefits (API consumer):** Promotion of own name/website, new interesting information about the club (e.g., responses of star players to fan questions)

At this stage, new requirements for the APIs are identified: the ability for API consumers to customize certain merchandise and the ability to vote on submitted user feedback and questions.

These cases serve as guidelines for the initial API marketing strategy. Once API consumers actually start using these APIs and the human innovation factor kicks in, these scenarios might get updated or even totally replaced by new ones (i.e., if consumers discover different ways of using the APIs and getting different/additional benefits).

We have two possibilities for accomplishing this. We can follow the top-down approach and begin with a goal or a set of goals that we want to achieve, as shown in Figure 1.

A good starting point is to look at a list of potential benefits offered by the API economy:[3]

- Decreasing application development costs and time
- Keeping up with application demands
- Achieving wider and quicker coverage of different platforms and devices
- Focusing on your core values by leaving application production to API consumers
- Finding and capitalizing on new partnerships
- Expanding into new customer bases
- Expanding brand name and loyalty
- Influencing industry standards and user expectations
- Keeping up with the competition

We can take selected benefits as our business goals or, if needed, we can make them more specific based on our context and needs.

The advantage of this approach is that we get specific goals that we can use to guide the whole process, monitor its progress, and measure its success or failure. On the other hand, we are not necessarily "squeezing" the most value out of our existing business assets. In fact, it might well be that the goals we aim for do not match what we have to offer API consumers.

An alternative option is to go bottom-up, starting from the available business assets, as shown in Figure 2.

Business assets. In the context of the API economy, business assets are assets that can be used via exposed APIs. Typically these include things like data, products, and services.

Here the general goal could be described as "exposing API(s) to achieve maximum business gains from the existing business assets."

In this approach, we put the focus on one of the key features of the API economy — the unlocking of human innovation. It is impossible to predict all of the potential usages of the exposed API(s) that API consumers will come up with. Therefore, this approach targets human innovation so as to leverage the most out of existing assets. On the flip side, we do not know which exact types of benefits such a move will yield. It is also more difficult to come up with good measures of success.

Regardless of the chosen approach, the result of the first step is to identify one or more API business goals. Everything that comes next is targeted to accomplishing these goals.

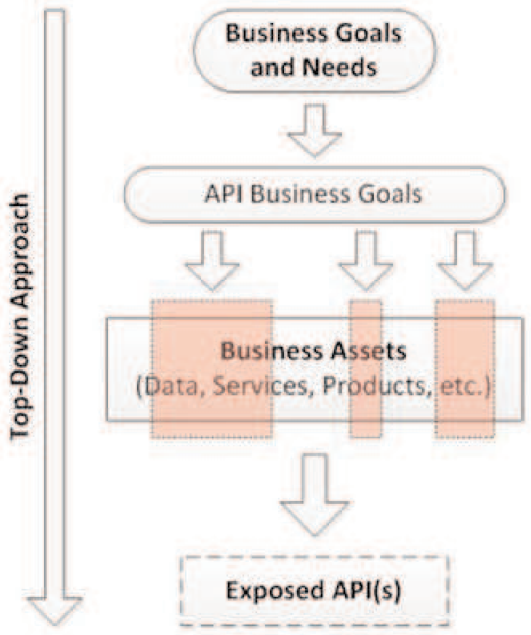


Figure 1 — Top-down approach to API design.

Step 2 — Examine the Market and Business Environment (Internal and External)

The second step is to understand the market situation and the internal and external environment related to our business. We want to consider potential competition, partners, and consumers for our API(s). At this stage, general business methods of domain analysis are applicable. Another option is to use an approach tailored specifically to software production, such as the one Paolo Predonzani and his coauthors discuss in *Strategic Software Production With Domain-Oriented Reuse*.[4]

API ecosystem. An API ecosystem is a set of interconnected API networks that might include different APIs, API providers, API consumers, and end users.

The API economy is an economy of networks and network connections. Therefore, an important aspect of environment analysis is identification of API ecosystems that might be relevant to our API(s). We might want to join or tap into specific ecosystems, or we might decide to form a new one based around our own API(s).

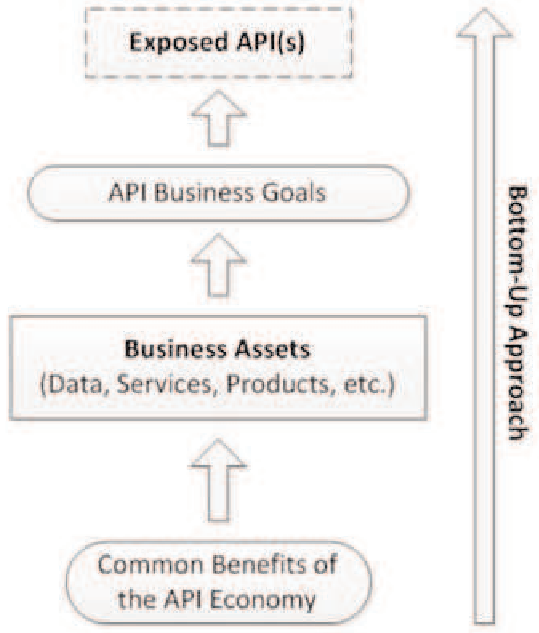

Exposed API(s)
API Business Goals
Business Assets
(Data, Services, Products, etc.)
Common Benefits of the API Economy
Bottom-Up Approach

Figure 2 — Bottom-up approach to API design.

Step 3 — Identify the API Consumers

While we are unlikely to be able to identify every possible type of API consumer for our API(s), we can at least predict the likely initial consumer groups. These groups will determine our initial API marketing strategy.

We also advise taking into account the different API ecosystems that these API consumers belong to. This can give us a clear understanding of which API networks are the primary targets for our API(s). It can also help refine the list of ecosystems identified in the previous step.

API PORTFOLIO MANAGEMENT AND API FINANCIAL MANAGEMENT ACTIVITIES

Step 4 — Consider the Risks

Now that we have precise business goals and some understanding of the environment and potential consumers, we can analyze the risks related to these goals. Here again we can start with a list of some of the common challenges faced by companies in the API economy (see Table 1).[5]

Once we have a list of potential risks, we can follow the same basic steps as in ITIL: assess impact and estimate probability of each individual risk. Based on that, we can decide which risks we want to address or safeguard against and to what extent.

Step 5 — Make Business Cases

In ITIL, we use business cases to understand the likely outcomes of specific business decisions. In particular, we focus on business objectives addressed by a specific service and its anticipated business impacts (quantifiable and unquantifiable). This is not so straightforward when we deal with APIs. The difficulty here comes from the fact that, as API providers, we have limited control of the road to achieving our API business goals.

Table 1 — Some of the Common Challenges Companies Face in the API Economy

General Challenge	Examples of Potential Risks
Safeguarding security and privacy	Increased security risks and privacy concerns because of data openness and sharing
Maintaining trust between providers and consumers	Losing API consumer trust because of failure to maintain API compliance and backward compatibility Alienating API consumers because of changes in the exposed API, accompanying business model(s), or related SLAs
Attracting API consumers	Failure to attract enough API consumers to get the API going Losing API consumers due to competing APIs
Maintaining effective and efficient APIs	Failure to cope with increased and/or unanticipated usage of the exposed API API requiring critical changes that break compatibility or have other heavy negative impact(s) on existing API consumers

A big part of it depends on the API consumers and possibly even end users.

End users. In the context of the API economy, the end users of an API exposed by an API provider are users of products or services delivered by API consumers of that API.

As API providers, we cannot control consumers directly, but we can influence and guide their actions. However, before thinking of how to influence consumers, we need to understand first what that influence should lead to. This is where business cases come into play.

We use business cases to construct scenarios of how we could achieve our API business goals. Here we make use of the information about target API consumers and API ecosystems to build cases that address possible roads to achieving our targets.

To be more specific, business cases tell us which types of consumers we need to target to reach specific business objectives (see Figure 3). Business objectives are breakdowns of API business goals into smaller, more specific targets. Sometimes, however, objectives can be the same as business goals (e.g., when the anticipated outcomes of one business case completely cover the business goal).

Notice that the target of these business cases is our own business and API business goals. As such, we refer to these cases as *internal business cases*. Having internal business cases helps us understand what we want from API consumers.

The next question is how to make consumers deliver what we want. To find an answer to that, we employ business cases again (see Figure 4), but this time the target is the benefit from the exposed API(s) to the API consumers — a direct analogy to building business cases for services. (See sidebar "Sample Scenario: Business Goals → Business Cases → APIs.")

To differentiate from previous usage, we call this type of business case an *external business case*. We are only interested in those external business cases that involve the types of API consumers we have identified previously.

Basically we need to come up with scenarios that will motivate consumers to use our API(s) in a specific way. This might seem counterintuitive, as the whole point of exposing APIs is to unlock the human innovation of third parties. In practice, however, innovation does not happen by itself. Just because we expose an API does not guarantee that consumers will come. We want to give a starting impulse (direction) to get the innovation going. Think of it as a snowball that we make and roll downhill — we give it initial mass and momentum, but then it takes on a life of its own. In other words, all we need are some initial business cases that will attract the right kind of consumers. (See sidebar "Sample Scenario: Business Goals → Business Cases → APIs.")

A common piece of advice given to API providers for unlocking human innovation is to do it in iterative stages: first target human innovation internally (involve your own employees), then move to your business partners, and finally go to the general public. This principle can be effectively applied to the construction of external business cases as well. We start by constructing such cases for internal use (within our own organization). As we expose our API(s) to partners and/or the general public, we add new business cases or improve the existing ones.

Step 6 — Choose the Right Business Model(s)

At this point we should already know what types of consumers we are targeting and how we are going to attract them. Based on the internal and external business cases, we can choose a business model or models that are best suited to our API business goal(s).

A variety of API business models are being used in the industry, including: free, developer-pays, developer-gets-paid, and indirect models. In fact, ProgrammableWeb founder John Musser identifies around 20 different API business models and their variations.[6] It is helpful to keep in mind that most models are not exclusive and can be used in combination.

Step 7 — Estimate and Manage API Costs and Budget

Business scenarios together with selected business model(s) allow us to estimate the amount of financial investment we need to devote to the API(s) and their expected ROI. Here it is important to remember to include the ongoing costs related to the marketing of the API(s) and support of the developer (API consumer) community. Generally, a raw budget would be estimated at this stage and then later adjusted, after we finalize the marketing strategy and determine the exact type and amount of developer support we are going to provide.

API DEMAND MANAGEMENT ACTIVITY

Step 8 — Construct PBAs

In ITIL, we use patterns of business activity (PBAs) to identify any patterns in business activity that the target service is meant to support. When we talk about APIs, however, we do not have a corresponding business activity. In fact, the exposure of the API itself *is* that business activity. Consequently, we cannot examine patterns of the activity that are not yet in place. Nevertheless, we can analyze similar business activities — other APIs we have previously exposed (if any) or similar types of APIs exposed by other businesses. PBAs here are the patterns in API usage by different types of API consumers. The purpose is similar to that of PBAs in ITIL — to get a better understanding of the activities so that the API(s) in question can be better optimized to usage requirements.

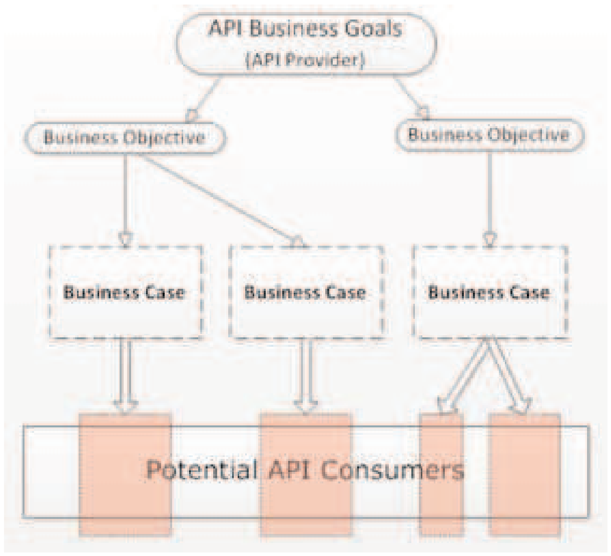


Figure 3 — Overview of internal business cases.

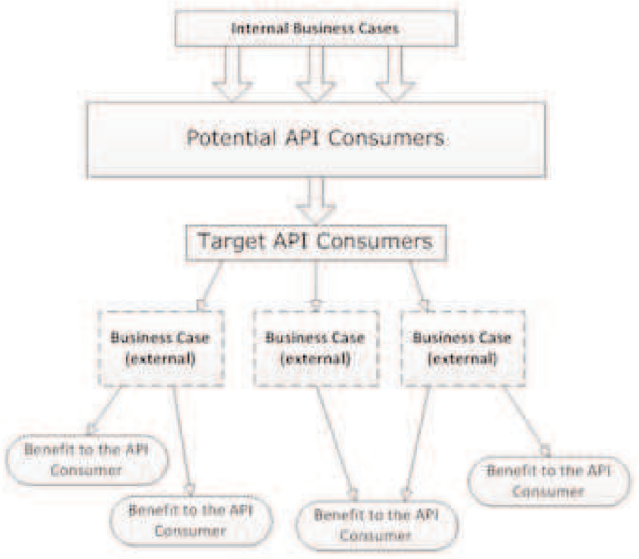


Figure 4 — Relationship between external and internal business cases.

There might be situations when we are not able to find similar APIs. In such scenarios, we skip this step. However, we can still construct PBAs at a later stage (after the deployment of the API) to use them as part of the continual improvement process and when exposing new APIs.

Trust is something that gets established over time and is very easy to lose.

API BUSINESS RELATIONSHIP MANAGEMENT ACTIVITY

Step 9 — Determine the Marketing Strategy

We need to be able to attract target API consumers to achieve our API business goals. For this reason, it is critical to decide on a clear marketing strategy. It is especially important at the initial stage, when the API is first exposed to the public. Later, if the API is successful, no active marketing strategy may be needed — the API consumer community will do our marketing for us.

Step 10 — Determine the API Consumer Support Strategy

Unlike the marketing strategy, support of API consumers is an ongoing process that lasts the whole lifecycle of the API, and possibly even longer. Therefore, it is necessary to anticipate support costs and adjust the API budget accordingly.

An API consumer support strategy has two main purposes: attracting new consumers and building and maintaining trust with the existing ones.

To attract new consumers, we need to consider different ways and tools to help developers get started with our API(s). This is where our external business cases become invaluable. Other than that, there are a number of things we need to take into account, including:

- Providing easy-to-use and helpful API documentation
- Providing practical API usage examples and tutorials
- Maintaining the developer portal
- Having a convenient feedback and support system

Trust is something that gets established over time and is very easy to lose. Therefore, it is especially important to have a clear strategy for building and maintaining trust. The keys to that are clear and honest communication with consumers and a guarantee of safety in terms of consumer effort and dependencies on our API(s). This means that we avoid making breaking changes to the API(s), drastically changing the licensing or the terms and conditions of API usage without consumer knowledge, "borrowing" consumer ideas and outcompeting their products, and so on.

WHAT TO DO NEXT? OTHER STAGES OF THE API LIFECYCLE

In the sections above, we covered the API strategy process, which is arguably the most critical one in the whole API lifecycle. After all, it governs how we expose our API(s) and explains why we do it in a specific way. What follows next is the actual design and implementation of the API(s), followed by their transition into production and actual operation. Of course, there is also the continuous improvement cycle that allows us to optimize our API(s) and related processes and to adapt to the ongoing changes in the market.

In principle, at this stage we could form what we have as a new business request to the IT organization (i.e., to implement the exposure of the selected API based on the established strategy) and proceed with the standard ITIL framework. Alternatively, we can continue using the adapted version of the API lifecycle stages. The advantage of the latter approach is that we can have a methodology that is it tailored specifically to the APIs. Debating all the pros and cons of each approach, however, is beyond the scope of this paper. Finally, we are not forced to use the IT service–based approach at this stage at all and can proceed with the actual implementation as a regular software development project.

The process of exposing a new API is not a trivial one. There is always uncertainty stemming from the fact that ultimately it's the API consumers who determine the success or failure of the exposed API. We believe that the methodology described here can help alleviate this uncertainty to an extent and make the creation of a new API program more likely to succeed. Furthermore, the focus on business aspects should help an organization keep track of the API program and ensure it is going in the right direction — namely, toward delivering business value. That said, however, we encourage readers to look at this methodology as a general framework and to improve or expand its steps as necessary.

ENDNOTES

[1]Lane, Kin. "API Deployment." API Evangelist (http://deployment.apievangelist.com).

[2]Official ITIL Website (www.itil-officialsite.com).

[3]Gat, Israel, and Giancarlo Succi. "A Survey of the API Economy." Cutter Consortium Agile Product & Project Management *Executive Update*, Vol. 14, No. 6, 2013.

[4]Predonzani, Paolo, Giancarlo Succi, and Tullio Vernazza. *Strategic Software Production with Domain-Oriented Reuse*. Artech House Publishers, 2000.

[5]Gat and Succi. See 3.

[6]Musser, John. "API Business Models." Presentation to the *API Strategy & Practice Conference*, New York, NY, USA, February 2013 (www.slideshare.net/jmusser/j-musser-apibizmodels2013).

Tadas Remencius is a Researcher at the Free University of Bolzano-Bozen (Italy). His research interests include Web APIs, empirical software engineering, software and team metrics, teamwork in software development, data visualization and interpretation, and experience management. Mr. Remencius holds a master's degree in computer science from Vilnius University (Lithuania). He can be reached at Tadas.Remencius@unibz.it.

Giancarlo Succi is a Senior Consultant with Cutter Consortium's Agile Product & Project Management *practice. He is also Professor of Software Engineering and Director of the Center for Applied Software Engineering at the Free University of Bolzano-Bozen. Dr. Succi's research areas include Agile methodologies, open source development, empirical software engineering, software product lines, software reuse, and software engineering over the Internet. He is the author of four books and more than 300 papers published in international conferences proceedings and journals. He can be reached at gsucci@cutter.com.*

7 API Challenges in a Mobile World

by Chuck Hudson

Recently, Apple released its next-generation iPhone models, including a reduced-price and multicolor iPhone 5C. Android devices are being registered worldwide on a daily basis, and a new update to the Android OS has been named. Today it is typical to see business executives launching presentations from their tablets or other mobile devices. Through sheer numbers, the mobile smart device has rapidly become a primary user interface.

In turn, the applications that users launch on their devices have become an essential part of the mobile ecosystem, and an explosion of mobile development has resulted in hundreds of thousands of apps available for download. When the new range of smart and powerful mobile devices is combined with an ever-growing set of applications leveraging the Internet and various data repositories around the world, there are bound to be challenges involving the systems that support these applications, including Web services and APIs. Whether APIs are purely for internal developers and branded applications or are supporting a public developer community, a new set of challenges is created by the widespread use of mobile devices and native applications employing remote API calls. In many cases, these challenges could not have been foreseen, as the architecture of the APIs being employed dates prior to the explosion of smart mobile devices.

The good news is that if you are an API provider or API consumer, the difficulties of migrating an API to support this large wave of new usage are not insurmountable, and there are many best practices that you can leverage. The following is a list of seven critical challenges that the incorporation of APIs in mobile applications has exposed, along with some possible solutions for both providers and consumers.

CHALLENGE #1: CONNECTIVITY

The most basic challenge of consuming APIs in the mobile space is device connectivity via Wi-Fi or cell coverage. Just think, when was the last time your smartphone dropped a call? Probably yesterday — if not an hour ago. Continuous connectivity is not a realistic expectation in the mobile space, unlike desktop browsers, as users move from one location to another with varying degrees of coverage, and signal blockages occur. (Elevators, for example, are not usually the best location for maintaining a connection.) Thus, the connection to remote APIs cannot be guaranteed while in a mobile environment since the level of service underlying the connection is variable. It would be rather frustrating for users (albeit slightly amusing) if an application, when making an API request, displayed the following message: “While your information is retrieved, please do not move.”

Since the mobile application interface is ultimately responsible for display of the retrieved data or call status to the user, users will naturally blame the application if timeouts or failures occur. It therefore becomes the responsibility of the API consumer to handle connectivity issues and timeouts gracefully. API consumers should expect that, at some point when a user is navigating a mobile application that engages remote requests, those remote requests will fail for a reason outside their control. Responding to this failure either through the display of an appropriate “lack of connection” message or retrying the call automatically is essential for an improved user experience. Whether fortunate or unfortunate for application developers, users have become accustomed to connectivity issues when using mobile devices, so the display of this message would not be out of the norm. Simply leaving a user waiting for a response that never comes, however, would be both frustrating and result in poor mobile application reviews.

In addition to basic error checking and messaging, a mobile application developer can add programming logic that can decide when and what calls to make based on connectivity. This enhanced programming logic would leverage the device system properties to determine the current connection performance. In cases where an important API request may be called or a request is known to be “heavy” in terms of the amount of data transported, the mobile developer can check both the connection type (Wi-Fi or cell) and the strength of the connection prior to making the call. If a minimum threshold is not met, then the call can be delayed until a more stable connection is attained. In the Android platform, checking the connection properties is a relatively straightforward task using the *android.telephony* library and *SignalStrength* class. The iOS environment makes it a bit more difficult, as the signal strength of the device connection is not exposed. To work around this iOS SDK limitation, a developer can instead monitor connection performance by testing data speed when desired and setting a minimum threshold for executing critical or large calls.

CHALLENGE #2: OPERATION OVERHEAD

In addition to the connection through which API calls are made, there is the total operation overhead, which contributes to the performance of API calls from mobile applications. The total call overhead results from the combination of the size of the payload sent and returned and the number of requests being made in an operation. In many cases, several remote calls will need to be performed in a particular flow with specific data pieces retrieved from each response to be used in subsequent calls or for a final set of data to display.

A common example of this flow is the retrieval of weather information for the current location of a mobile device. First a geolocation call is made to retrieve a location data element such as a zip or postal code that can be used in a subsequent weather request from a different service. This is a fairly simple example but one that shows how the overhead can become significant as the mobile application acts as a call coordinator for these multiple calls. Caching the location information in the application is an option for improving subsequent performance, but if the location is used again, there is a more reliable solution, as explained below.

The bulk of the call overhead is the number of request and result calls that must be strung together. If instead the mobile application is able to make a single request, this will reduce the call overhead and processing significantly. What is needed is a new endpoint located in the cloud that can act as the call coordinator performing the call requests, response handling, basic logic, and even some intelligent caching. In fact, the caching and logic can be even more powerful when put into the cloud, since the service can take advantage of knowledge gained from calls from multiple devices. In our weather example, this could mean that if device M is in close enough proximity to where device C previously requested weather information, then a cached data set could be provided without any other requests to external APIs being made.

To make creating these types of call-brokering endpoints in the cloud easier, a new type of service offering has been developed. The service is called "BaaS" or "backend as a service," by which developers can build new endpoints in the cloud in a language of their choosing (see Figure 1). Several small startups are providing these services,[1, 2] and even Google is previewing a release of what is called, appropriately enough, Google Cloud Endpoints.[3] Providing a custom endpoint in the cloud makes it easy for the API provider to maintain the original API set with minimal changes, while the application developer (the API consumer) has an endpoint layer with which to request exactly what is needed in a single request.

CHALLENGE #3: LOOSE CALLS

Many API calls were originally designed for a specific task: saving a profile, returning a list of items in a cart, and so on. For website implementations, these general calls were sufficient. But for mobile integration, where less is more due to connectivity and processing overhead, providing a level of granularity is essential. As an example, when developing a mobile application that retrieves a large list of data from an API, a list view control available from the platform's SDK can be used for displaying the list. However, the default behavior when displaying a large list of items is for the list to be rendered after the list entries are created. Some mobile platforms such as Android have native adapter classes that can manage the display of large data sets from memory or a database set; however, without being able to define the granularity of the return list in the original API request, there is still the overhead of returning the

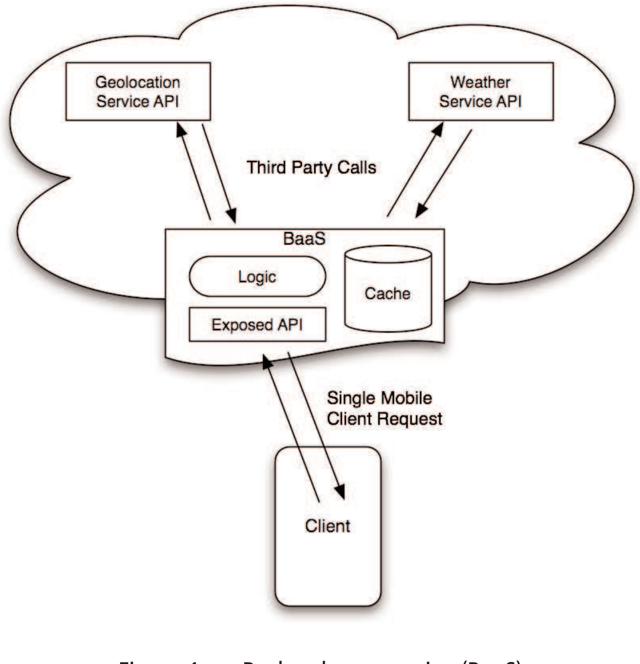


Figure 1 — Backend as a service (BaaS).

entire set of data to the device and processing the data set.

To facilitate this deeper level of granularity, filtering should be available so consumers can set restrictions on the type and quantity of data to be returned. A typical filter for a return list of data would include a start index and a count of items to return. In this way, the call can leverage "paging" of the data set for only the pages that the user desires. To determine the quantity sought in each block or page, a balance must be struck between the overhead of the data to be returned and how many pages the user would wish to view on average. Depending on the format of the API, filtering could be as simple as query string options, such as "?start=20&count=10".

CHALLENGE #4: AUTHENTICATION

One of the first things you will notice when examining how to use authentication with an API set is that there are several options available, including open source libraries, commercially available libraries, OAuth and OAuth 2.0, and specific OS libraries such as the Android Account Manager. In general, there is no standard for authentication across the board. Many APIs have incorporated OAuth or OAuth 2.0 into their framework for authentication, but in the mobile space these authentication practices have their own issues.

The OAuth process requires that an access token be retrieved by having the user authenticate through a browser set to the endpoint with the authenticating site. After the user authenticates his or her profile, an access token is returned to the mobile application through a registered URL scheme. The URL scheme is a map for browser windows or other applications on a mobile device to call a native application on the device and pass data. As part of the startup logic of a mobile application, the developer can register a specific URL scheme. However, a phishing application installed on the same device could also register for the URL scheme, thus gaining access to the access token when returned. If instead a Web view is embedded in the native application for the authentication endpoint to be loaded, then unfortunately the native application can access the data entered in the contained Web view. OAuth 2.0 attempted to address these issues by opening up new methods, however it breaks a fundamental rule of authentication by collecting credentials from users directly.

At the moment, there is no real solution to this problem. One potential avenue for API providers is to create their own authentication with the level of security that is required for their offering. In turn for good practice and easy integration by the API consumer, they can provide a standard library for each of the common mobile platforms that an integrator can easily leverage. PayPal, for example, has created a set of mobile SDKs for both Android and iOS that encapsulates the payment processing secure calls.[4]

CHALLENGE #5: GENERIC APIS

When developing an API, architects will focus on deciding which call model to employ, such as REST, and determine the data transport format to leverage, such as JSON. Typically this call style and data format conclusion is based on the systems already employed and technical knowledge within the organization. The problem with this approach is that it assumes all API consumers and consumer platforms can use the chosen data format with equal ease. Given the ever-expanding array of mobile device platforms available, including iOS, Android, and others, a single data format is not necessarily the best option, even though it may be easier for the provider to implement and maintain.

Instead, while continuing to separate the data gathering from the data formatting, the concept of client adapters can be employed to provide a layer between the client side and server side through which consumers can choose a return format based on their environment, such as JSON for Web pages, plists for iOS, and so forth. In this way, the core API calls can remain universal, but specific client adapters can be used to provide easier consumption across a wide range of devices and developer preferences. Netflix, as an example, must support a multitude of different devices. With a client adapter architecture, Netflix is able to keep a single base API for easier maintenance while providing platform-specific adapters for easier integration by developers.[5]

With the solution of a client adapter architecture, a single API can be exposed across multiple platforms through different data formats. To provide even more flexibility with the client adapter solution, the specific adapter to be used can either be specified by the consumer in the call request or automatically determined by the API host system based on the consumer's platform.

CHALLENGE #6: LICENSE RESTRICTIONS

Mashup developers are accustomed to API providers placing restrictions on usage, including time and quantity call quotas and data caching limitations. However, when incorporating third-party APIs for the mobile space, developers may be surprised to find a clause similar to the following in several providers' terms of use: "You may not implement the API or distribute the data on mobile apps." In most cases, this clause is inserted into the terms of use API license agreements in the hope of protecting the provider's own mobile application offering or potential entry into the mobile space. Unfortunately, given the broad positioning of the legal statement, this prevents the use of such an API in your mobile application. Hopefully over time (just as when Web APIs were first implemented years ago), companies will remove this clause as they learn that the incorporation of the calls and data in other mobile applications with proper attribution is a more powerful business strategy.

PayPal recognized this strategic reality a while back when other mobile applications had to launch the PayPal application installed on the mobile device to effect a payment transaction. This, of course, required users to have already downloaded and installed the PayPal mobile application on their devices. PayPal recognized that this limited the reach of the service and now provides a developer library on key mobile platforms for direct incorporation of its payment transaction service into mobile applications.

Some API providers are legitimately concerned about the potential system impact from a significant increase in API traffic. However, this can be addressed through similar practices as used with APIs for website incorporation, such as usage limitations and endpoints allocated just for mobile integration.

CHALLENGE #7: INCREASED AND REPETITIVE CALLS

One of the challenges for API providers entering the mobile space is the potential for a significant increase in call volume. In addition, through automated functionality of mobile applications, calls may get repeated frequently. Take, for instance, new offerings that provide users with discounts and recommendations via mobile devices as they walk through a retail store. Upon entering the store, users are registered via their mobile device through technologies such as sonic signal detection, geofencing, and Wi-Fi fingerprinting. If API calls are executed each time the device is detected in the store — or even worse, on a regular basis for the duration of the visit — then the increase in call volume could be significant. To mitigate this automatically created traffic, several tactics could be employed, including caching data and mirroring frequently accessed data in a secure manner at the location.

CONCLUSION

This has been a brief look at some of the various challenges of leveraging APIs designed for well-known Web application usages to a mobile space. API consumers need flexibility based on device OS platforms, data formats, and connectivity dependencies from API providers. In turn, mobile application developers need to be good citizens when using APIs by minimizing the number of calls performed. By mutually addressing these challenges, API providers and consumers can improve the mobile application user experience by providing the right data when it is needed in a timely manner. On both sides, solving these challenges requires a careful balance between the breadth of API coverage and the depth of control over return data.

ENDNOTES

[1]Parse (www.parse.com/products).

[2]StackMob (www.stackmob.com).

[3]"Overview of Google Cloud Endpoints." Google Developers (https://developers.google.com/appengine/docs/java/endpoints).

[4]"PayPal Mobile SDKs." PayPal Developer (https://developer.paypal.com/webapps/developer/docs/integration/mobile/mobile-sdk-overview).

[5]Jacobson, Daniel. "Embracing the Differences: Inside the Netflix API Redesign." *The Netflix Tech Blog*, 9 July 2012 (http://techblog.netflix.com/2012/07/embracing-differences-inside-netflix.html).

Chuck Hudson has been at the intersection of Web business and technology since the inception of online commerce in the mid-1990s. Having programmed in numerous Web and mobile languages, he combines a passion for the commerce lifecycle and a wealth of experience to create innovative solutions. He is currently a Director of Application and Mobile Development at Control4, focusing on HTML5, iOS, and Android products. He has authored the HTML5 Developers Cookbook *and the* eBay Commerce Cookbook.

With an MBA concentrating on entrepreneurship and managing technologically innovative enterprises, Mr. Hudson has been a successful entrepreneur and consulted with companies for Web and mobile product and service strategies. He shares his knowledge of Web and mobile product execution through business advisory roles and as a visiting faculty member in the Masters of Internet Technology program at the Terry College of Business, University of Georgia.

Mr. Hudson has spoken and led lab sessions on Web and mobile best practices at development conferences nationally and internationally. In 2008, he received the eBay Star Developer award for the first iOS-based Web and native applications for users of eBay. Mr. Hudson is also a certified PayPal developer and certified PHP programmer, and he sits on the PayPal Developers Council. He can be reached at chuckahudson+api@gmail.com.

Cutter Consortium

Executive Education+

SUMMIT 2013

4–6 November 2013
Cambridge, MA, USA

Intense, Interactive Instruction. Effective Learning.

Register today and save!

Single Seat: $1995 for a limited time (save $500!)

Team Builder: Buy 1 seat for $2495 and bring a colleague at a deep discount. (Save 30%)

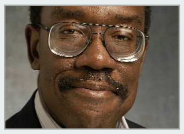
Executive education on IT leadership and emerging trends. A year's worth of professional development and personal enrichment in 3 invigorating days.

From an in-depth case study taught in the popular business school style, to interactive small group exercises and keynotes that help you identify business opportunities made possible by emerging technologies, you'll enjoy truly unbiased discussion and meaningful debate on today's IT opportunities and challenges at the Cutter *Summit*. Discover and learn about new strategies, technologies, and leadership skills — from Cutter's exceptional lineup of experts —that will help you embrace the ever-unfolding opportunities and challenges of the SMAC business environment.

In addition, you'll benefit from hands-on seminars and roundtables led by Cutter's Practice Directors and Senior Consultants on topics such as software engineering and agility, business and enterprise architecture, CIO/CTO issues, and data insight and social BI, to name a few.

You'll enjoy (and join in on!) raucous panel debates; networking at lunches, breaks, and entertaining evening events; and get one-on-one guidance and input from expert presenters and participants.

Monday, 4 November 2013



The Evolving Role of 21st-Century Technology Leaders

Keynote by Robert D. Scott

Cutter Fellow; Director of the Information Systems Executive Forum, Ross School of Business, University of Michigan

Panelists: Robert Austin, Sheila Cox, Art Hopkins

Big — and Fast — Data Analytics

Case Study with Vince Kellen

Cutter Fellow; Senior Vice Provost for Academic Planning, Analytics & Technologies, University of Kentucky

Lightning Talks

Hosted by Tim Lister

Short and to-the-point presentations around a single strategy, technique, or success story.

Evening Cocktail Party

Unwind and socialize with the speakers and your fellow attendees while enjoying some of Boston's tastiest regional specialties.

Learn more at www.cutter.com/summit.html

Executive Education +
SUMMIT 2013 4–6 November 2013
Cambridge, MA, USA

Cutter Consortium

Tuesday, 5 November 2013

Putting Your Leadership Skills to the Test

Active Learning Exercises with Michael Roberto

Cutter Fellow; Trustee Professor of Management at Bryant University

Agile in the API Economy

Keynote by Israel Gat

Fellow and Director, Cutter Agile Product & Project Management Practice

Panelists: Tom Grant, Giancarlo Succi

SMAC: Could It (or Does It) Alter the Way Your Company Does Business?

Roundtable with Curt Hall

Senior Consultant, Cutter Consortium

Digging for Gold in the Emerging Technology Pile of Hype

Keynote by Lou Mazzucchelli

Cutter Fellow

Panelists: Ron Blitstein, Madge M. Meyer, Ty Vaughan

Sustainable Growth: Achieve It with Highly Motivated Teams

Roundtable with Lynne Ellyn

Cutter Fellow

Serious Games

Roundtable with Tom Grant

Senior Consultant, Cutter Consortium

Wednesday, 6 November 2013

The Chief Data Officer

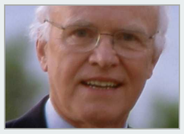
Roundtable with Larissa Moss

Senior Consultant, Cutter Consortium

Designing Effective Dashboards

Roundtable with Giancarlo Succi

Senior Consultant, Cutter Consortium



A Theory of Practice: Soft Decision-Making in the Context of a High- Pressure IT Organization

Keynote by Tom DeMarco

Cutter Fellow

Choose a Track ▸

BUSINESS TECHNOLOGY STRATEGIES TRACK

CIO/CTO Roundtable

Forum with Ron Blitstein

Fellow and Director, Cutter Business Technology Strategies Practice

BUSINESS & ENTERPRISE ARCHITECTURE TRACK

Business and Enterprise Architecture

Workshop with Dan Dixon

Senior Vice President, Wells Fargo

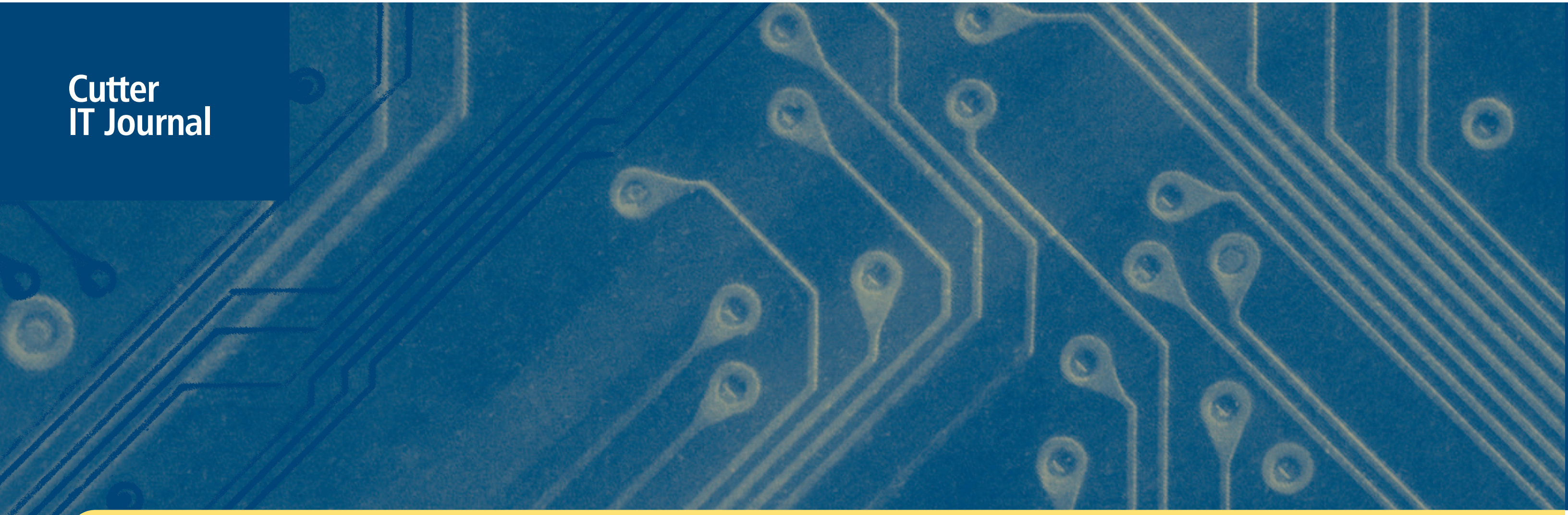
AGILE TRACK

Agile Masterclass: Beyond the Basics, Beyond the Hype

Workshop with Israel Gat* and Hubert Smits**

** Director, Cutter Agile Product & Project Management Practice; ** Senior Consultant, Cutter Consortium*

See the full program at www.cutter.com/summit.html



About Cutter Consortium

Cutter Consortium is a truly unique IT advisory firm, comprising a group of more than 100 internationally recognized experts who have come together to offer content, consulting, and training to our clients. These experts are committed to delivering top-level, critical, and objective advice. They have done, and are doing, groundbreaking work in organizations worldwide, helping companies deal with issues in the core areas of software development and Agile project management, enterprise architecture, business technology trends and strategies, enterprise risk management, metrics, and sourcing.

Cutter offers a different value proposition than other IT research firms: We give you Access to the Experts. You get practitioners' points of view, derived from hands-on experience with the same critical issues you are facing, not the perspective of a desk-bound analyst who can only make predictions and observations on what's happening in the marketplace. With Cutter Consortium, you get the best practices and lessons learned from the world's leading experts, experts who are implementing these techniques at companies like yours right now.

Cutter's clients are able to tap into its expertise in a variety of formats, including content via online advisory services and journals, mentoring, workshops, training, and consulting. And by customizing our information products and training/consulting services, you get the solutions you need, while staying within your budget.

Cutter Consortium's philosophy is that there is no single right solution for all enterprises, or all departments within one enterprise, or even all projects within a department. Cutter believes that the complexity of the business technology issues confronting corporations today demands multiple detailed perspectives from which a company can view its opportunities and risks in order to make the right strategic and tactical decisions. The simplistic pronouncements other analyst firms make do not take into account the unique situation of each organization. This is another reason to present the several sides to each issue: to enable clients to determine the course of action that best fits their unique situation.

For more information, contact Cutter Consortium at +1 781 648 8700 or sales@cutter.com.

The Cutter Business Technology Council

The Cutter Business Technology Council was established by Cutter Consortium to help spot emerging trends in IT, digital technology, and the marketplace. Its members are IT specialists whose ideas have become important building blocks of today's wide-band, digitally connected, global economy. This brain trust includes:

- Rob Austin
- Ron Blitstein
- Tom DeMarco
- Lynne Ellyn
- Israel Gat
- Vince Kellen
- Tim Lister
- Lou Mazzucchelli
- Ken Orr
- Robert D. Scott