eBook

APIs and IT Rationalization

Cost avoidance and cost savings for enterprise IT

By Brian Pagano

apigee

Table of Contents

apigee

Introduction

These days, technology leaders are asked to do more and move faster. But they usually aren't given any more budget to meet the demands of a fast evolving, agile, responsive enterprise. Winning in today's environment requires more intelligent ways to achieve our goals; this, in turn, requires finding ways to remove unnecessary spend from our budgets, and freeing more dollars for innovation.

We'll need to rationalize our spending in a rational way. Maybe, if we look beyond the obvious (redundancies and shelfware, for example), we'll find dollars to move back into play.

There are two main aspects to focus on in IT rationalization: cost avoidance and cost savings. There's a third category, ROI/monetization, which we'll omit from this discussion as it stems from monetizing transactions flowing through an IT investment and is a more straightforward business case. The less well-understood and sometimes harder to find opportunities for reducing spend include:

- **Cost avoidance:** not paying for something that you would have had to pay for otherwise
- **Cost savings:** reducing (or eliminating) a current payment

ra·tion·al·ize

/'raSHənlˌīz , 'raSHnəˌlīz/

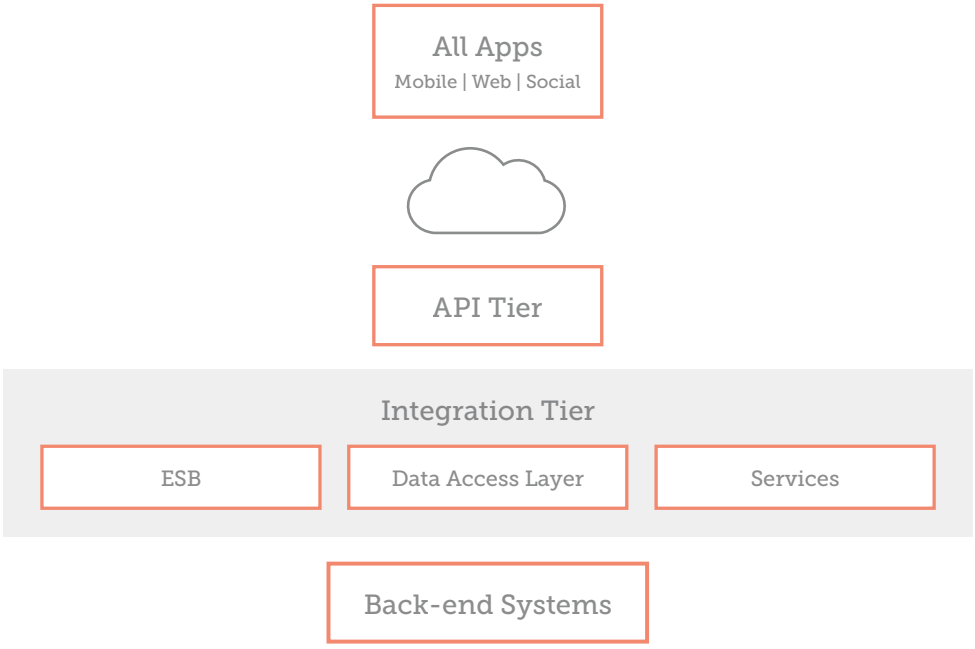
verb: to make (a company, process, or industry) more efficient by reorganizing it in such a way as to dispense with unnecessary personnel or equipment.
See also: doing more with less.

Cost avoidance is possible when one product can do the work of two. It can occur when additional headcount is avoided due to increased efficiency (for example if you add a product that can accomplish tasks via configuration instead of the usual full coding project).

The ability of an organization to accomplish a task faster leads to cost savings. This not only has time-to-market benefits, but it's also simple math: blended rate x number of hours to accomplish a task. The faster the task can be completed, the higher the savings.

Cost savings can also occur when an existing payment can be reduced because a less expensive (or more efficient) product can do some of the work of a more expensive system. This allows renegotiation and decrease of the more expensive contract at the end of the year.

Architecture

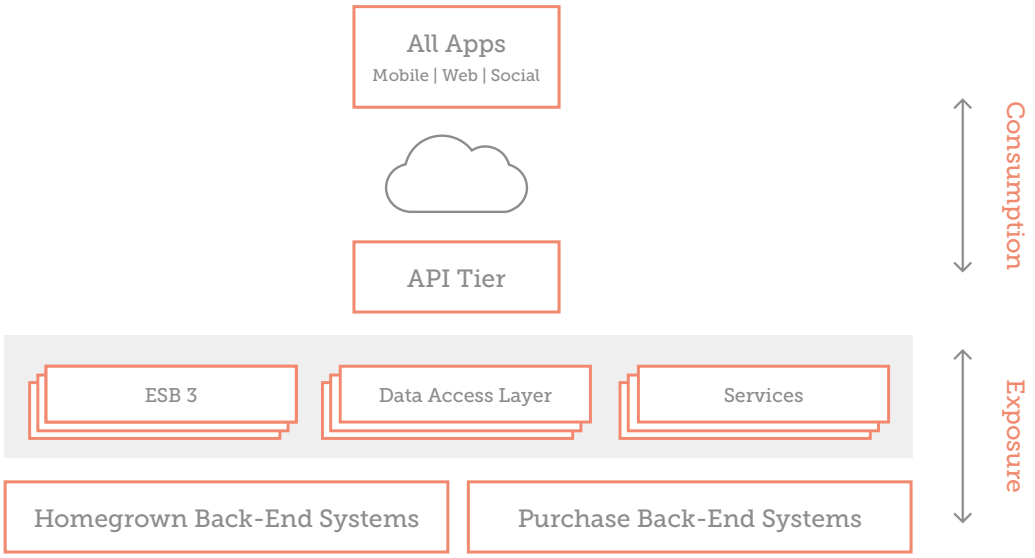


A high level, software layer view of an enterprise architecture

In a typical enterprise architecture, an API tier sits in front of the integration tier and enables the exposure of data from back-end systems and the consumption of data by apps. This is a privileged position in the architecture as it allows the APIs to perform mediation, orchestration, and traffic shaping that would be difficult or impossible in other tiers.

Several necessary and unnecessary costs are distributed throughout the architecture. We'll look at how an API can reduce or eliminate several of these costs, but first we need a quick reality check.

Modern IT's multiple personalities



A high level view of a typical architecture containing duplication at every layer

Unfortunately, before we can examine possible areas of unnecessary costs, we must pause to acknowledge a fact that complicates the life of every enterprise IT professional and multiplies the complexity and cost of completing projects.

Almost every modern organization will have multiple instances and possibly multiple flavors of each of the components of the enterprise architecture. This means that to have access to the right data and systems, a new application needs to coordinate with multiple teams across the company as well as negotiate multiple data repositories, multiple "sources of truth," different naming conventions, and different data formats. This is not the optimal route to efficient delivery.

apigee

Exposure and consumption - an architectural secret weapon

The best way to counteract the complexity of organic technology accretion (and its associated costs) is to introduce exposure and consumption layers to the architecture. In this way, access to back-end systems can be normalized via a uniform and highly consumable API (often viewed as a "facade").

This, in turn, cuts project costs by:

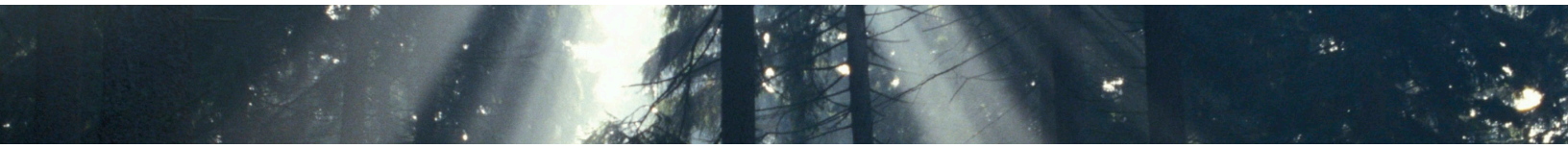
- **Reducing governance** access is only required to a facade layer (the API) and not to each individual underlying system.
- **Shortening development time** by simplifying the interface.
- **Eliminating** the need for back-end teams to make frequent changes for each new consumer all consumers now come through the API, thus cutting coding and maintenance costs.

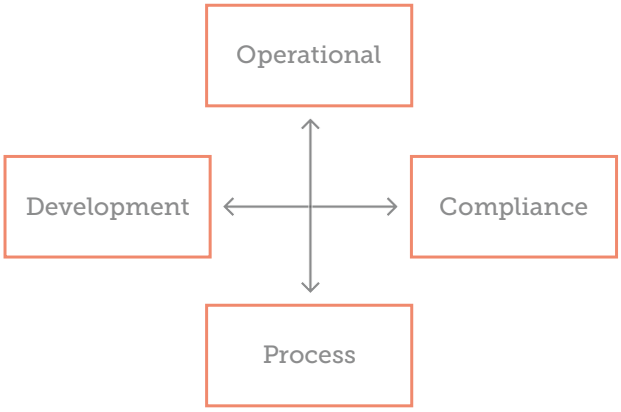
> The greatest bottom-line impact from digital transformation comes where most companies aren't looking—from cost savings and changes beyond the interface with customers. A year-long impact that can be realized from digital sales over the next five years is 20% but **the bottom line impact from cost reduction will average 36%.**
>
> *(McKinsey, November 2013, " Finding Your Digital Sweet Spot.")*

There are benefits over and above cutting costs at the project level. At the enterprise level, the technology ecosystem can now be modified without disrupting consumers of the facade. For example, one vendor's software can be swapped for a similar product made by a different vendor. As long as the API remains unchanged, consumers are unaffected, and indeed unaware, of the changes that have occurred.

apigee

A strategy for IT cost rationalization





Four categories of rationalization

An API tier enables both cost savings and cost avoidance in the following areas:

- **Operational**
- **Regulatory and compliance**
- **Development**
- **Process**

Operational savings

The path to rationalizing spending will often first lead you to your IT operations. Key areas to investigate include bottlenecks, capacity planning, and caching.

Blast through bottlenecks

Bottlenecks represent points in a process where optimal flow is impeded, often unnecessarily.

They come in three flavors: process bottlenecks, development bottlenecks, and request flow bottlenecks.

In the context of operational savings, request flow bottlenecks are a great concern. They can be caused by underperforming back-end systems, network issues, an underperforming data layer, or requests arriving in a deprecated format.

Bottlenecks due to requests in a deprecated format can be reduced through the use of an API management layer. Say your company moved to a new version of a service which accepts a different data format. For whatever reason, some of your partners are not able to send data in this new format. The options are to stop doing business with these partners (which seems shortsighted)

or to force your operations staff to support a deprecated version of the service.

A cleaner alternative is to have the API tier mediate the versions. In this way, you can accept the old format from partners and have the API management layer transform the request into the new format. This can be accomplished through simple configurations.

Process bottlenecks delay the completion of projects; let's examine them in the section on process savings. Similarly, development bottlenecks crop up when a development team is impeded from proceeding on its project. This is often due to a lack of availability of information, documentation, or system access. It can also occur when there's tight coupling of service producers and service consumers. We'll examine these development inefficiencies more closely in the section on development savings.

Calculate and plan capacity

There is an obvious advantage to having all internal requests, partner traffic, and external calls going through a single layer. This is advantageous for security and compliance, but, in the context of operational savings, this layer provides invaluable information for capacity planning.

With all of the traffic going through the API, the API management layer is a perfect place to spot trends and to calculate capacity. Given the lag times between budgeting cycles, procurement, and staging, it's crucial to have accurate information as early in the process as possible. This can prevent over- or under-ordering server capacity.

Reduce compute and network loads with caching

One way to reduce compute and network loads to back-end systems is through the use of a caching layer. As we saw earlier, the API tier in an enterprise architecture is optimally positioned for caching. Unlike more standard caching tools, an API management layer will have the intelligence to validate traffic, route requests, and perform both coarse and fine-grained caching.

This request filtering has the potential to greatly reduce invalid and unnecessary traffic to back-end systems (responses that could be served from cache, for example). Depending on the deployment topography, this will reduce latency, network saturation, and compute load on the back-end systems.

Save at runtime

Another unnecessary cost for IT is when bad requests reach back-end systems. A bad request is one that could have been served from cache or is an invalid request (a request lacking proper authorization). The unnecessary requests saturate the bandwidth and consume compute resources on servers. Filtering these requests at the API tier is an effective way to eliminate this cost.

Regulatory and compliance savings

Compliance and regulatory costs are huge burdens to doing business, whether these costs are industry-wide (the financial services industry is a prime example), or within the confines of a particular company's IT department. APIs offer many opportunities for savings the audit process and compliance.

> "APIs are meant to enable transactional business activity without prior restraint, and they are major tools for innovation and experimentation, saving organizations money through reuse and consistency. These characteristics can ultimately make governance easier."
>
> *Governance, Risk, Compliance and APIs*

Reduce audit costs

Audit committees require different information for different types of audit as well as for updating their requirements year after year. This burden can be cumbersome for an IT department. Having all access performed through a standard interface (an API) greatly reduces this burden by allowing a single system to produce a complete record of transactions. Further, a good API management platform will allow the logs to be output in the specific format requested by the audit committee, further reducing effort.

Keep up on the hamster wheel of compliance

Much like the changing requirements of the audit committee, the corporate security office will periodically introduce new or updated security policies that IT must implement. These changes require resources. People will need to be pulled off of other tasks. Tracking down each system that might be impacted by the policy change adds overhead.

However, with the API tier in place, most corporate security policies can be implemented in one place (versus chasing down all of the disparate systems that might be impacted by the policy). This can prevent the need for constantly pulling resources off of existing projects.

> "Reuse suppports a board objective to reduce IT service costs by at least 3% per year and allows [automobile electronics maker] Brose to respond to customer demands to reduce the price of products year by year. In 2009, Brose achieved a 10% reduction in IT service costs, contributing to price reductions of 5%."
>
> *(MIT, 2010, "Reuse is More Important and Harder than WE Thought")*

Development cost savings

Development bottlenecks can hamper a team by reducing information and documentation availability or system access. Making APIs configurable and consumable can alleviate many of the IT costs generated by these inefficiencies.

Avoid the headaches of homegrown systems

From a maintenance perspective, homegrown systems can be expensive, especially when they are utility systems and not directly related to the company's unique business domain. The team that owns this system must handle new feature requests, keep up with evolving technology standards (like security protocols), and support bug-fixing and trouble tickets. This is a lot to ask for a non-business, domain specific tool.

It's better to leverage a platform that provides a configurable (see below) mechanism to implement changes. An API tier is particularly well-suited to this task.

Many companies have some kind of homegrown platform, which started out as an integration platform, and has since morphed into a pseudo-exposure platform.

The pain and cost of maintaining such a system is alleviated by switching to a purpose-built API management tool.

Don't code it, configure it

Another source of cost leakage in a typical IT department is through the constant churn of unnecessary coding projects. There are good reasons to write code and bad reasons to write code. An example of a bad reason to write code is when a partner (or another department) asks for a slight change to the data format being consumed. This has a configuration opportunity written all over it. But, without a tool in place to handle these types of changes, one is left to spin up a programming project with all of the governance and overhead that entails.

A well-designed API platform will have the capability to allow quick configurations to modify behavior, shape traffic, and implement policies and logic. There is no need to do a full (and expensive) project.

Improve ramp-up time for development teams

Rather than being forced to learn new development and testing tools, a new programming language, and a new methodology, providing a consumable interface (an API) to all back-end systems allows new developers to ramp up quickly. This not only speeds time to market, but also has the benefit of producing higher quality code (because the developer is working in the context of his or her expertise) that is more maintainable (because the apps are based on standard technologies).

Reduce load on back-end development teams

Just as new development teams can come up to speed more quickly when using an API tier, it can also protect back-end development teams from much of the minor change request churn they're normally burdened with.

Many changes can be implemented using configuration (see above) instead of requiring a coding project from the back-end team.

> "The goal of an API Facade Pattern is to articulate internal systems and make them useful and consumable by app developers.
>
> *API Facade Pattern: A Simple Interface to a Complex System*

Process savings

The processes of governance, vendor upgrades, and system replacement can be burdensome to an IT organization. An API tier reduces headaches and costs on all these fronts.

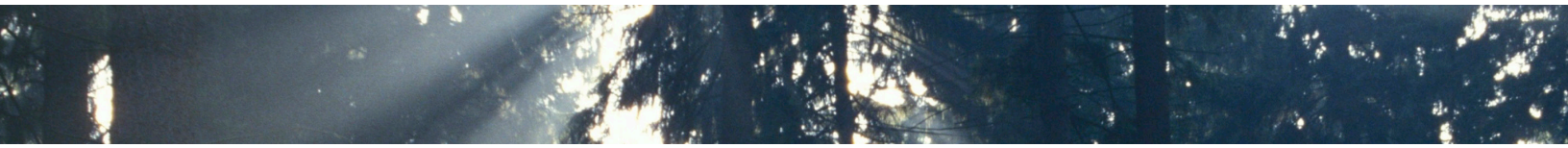
Streamline vendor upgrade and retirement

It's never easy to rip out an existing system and to replace it with a new one. But the process is much more painful and costly when there is tight coupling between that system and its consumers. Upgrades need to be coordinated across numerous teams, unless, that is, a facade has been placed in front of the back-end system. If all consumers are accessing the system via a facade, then this kind of system upgrade becomes much simpler (though never trivial).

Simplify governance

An API tier can reduce governance costs in a couple of ways. For one, RESTful APIs simply require less governance than the heavyweight contracts of something like SOAP. But another benefit is having a central point in which to implement governance policies, enforce compliance, and audit and log compliance.

Conclusion

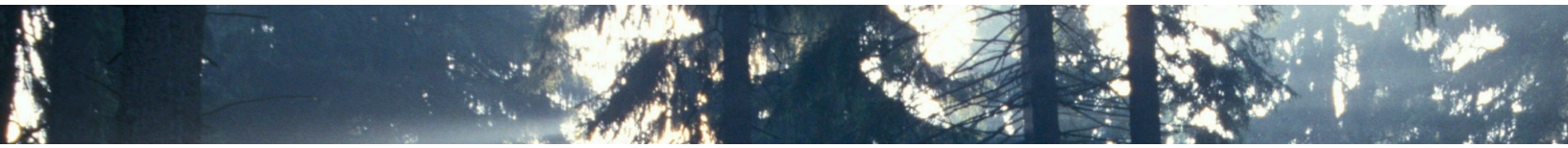


Having looked at the places where cost avoidance and cost reduction can hide, you should be ready to start searching through your own organization for opportunities to get some money back into the budget. Remember to look beyond the obvious and to examine each category: operational; regulatory and compliance, development; process.

Chances are that your organization is more efficient in some areas than in others. But it is a rare organization that cannot improve its efficiency and put its money to better use.

apigee

Appendix: Checklist for evaluating the IT rationalization opportunity



Cost Avoidance

1] Are you considering SOA tools like a sevice registry?

2] Will you need to write functionality that is common to all modern apps (location, device type, push notifications)?

3] How much effort will it take to expose your back-end systems safe and scalably?

4] Will you need to acquire or build a system to on-board and manage developers?

Cost Savings

1] What is the process for on-boarding a new internal development team?

2] What is the process for on-boarding a new partner?

3] How long does it take to get a new app to market?

4] How much of your traffic could be cached?

5] How easy is it to project server capacity requirements for the coming year?

6] Could your goverance process be simplified?

7] How much time and effort is spent on minor change requests?

About Apigee

Apigee is a leading platform for digital acceleration. Apigee empowers enterprises to gain the speed, scale, insight, and agility required to become a digital business. Through Apigee Edge API platform and Apigee Insights predictive big data analytics, Apigee helps businesses move at the new pace and scale of digital, while predicting and continuously adapting to change. Used together, APIs and predictive analytics create a powerful adaptive cycle of continuous improvement — and the faster an enterprise goes through this cycle, the faster it accelerates to become a digital business.

Many of the world's leading businesses, including 20 percent of the Fortune 100, use Apigee for digital acceleration. Apigee customers include global enterprises such as Walgreens, eBay, Shell, Live Nation, Kaiser Permanente, and Sears.

For more information, visit apigee.com.

About the Author

Brian Pagano is lead evangelist for digital success at Apigee. Before Apigee, Brian was Chief Architect of Guardian Life on Wall Street. He also served as the Chief Architect of a software company in Milan and as CTO of a software company in Turin, Italy. He did research and development at Saga Software, a company that made middleware solutions. Brian started his career as a programmer, specializing in object oriented languages.

Share this ebook