Next steps in open data reuse involving APIs

Submitted on 07 Oct 2013 by Jarkko Moilanen

We moved from being a manufacturing based economy, through services to a data or information based economy and API strategy is a key enabler of that economy. In this emerging economy information is currency. Cities across the world are opening data for others to consume. Often the starting point is to provide spreadsheets (files) as open data. Spreadsheets might contain economical information, statistics and other easy to list information about some geographical area.

In the same time governments are turning towards more open and engaging approaches, for example by opening easy to use channels for citizens to gain information about decision-making (Ahjo in Helsinki) and help in maintaining city infrastructure.

The above examples represent the next step in evolution of open data reuse – application programming interfaces or APIs. API stands for Application Programming Interface. In short, it is the equivalent of a Graphical User Interface (GUI) for programmers. Users interact with a program using the GUI, programs “interact” with other programs using the API.

API provides more flexible and application oriented tools to access data. APIs are often needed to access real-time data, for example bus or train locations, but can also offer views to spreadsheets as well. The amount of APIs raise in slower pace compared to data dumbs because API development requires more resources.

Despite the costs, it is expected that APIs will enable the next wave of open data. First wave is still on-going and refers to open data catalogs established around the world. APIs will take the data reuse to the next level. Of course rising amount of APIs will raise new issues which need to be tackled. API providers will need to find answers to questions like:

- how your organization manufactures,
- provides (distributes) and
- consumes APIs.

Often answering to those questions is referred as API strategy. As it was said, building APIs via conventional supplier chains is often expensive. Another option is to enable community driven API development. Opening development to community members:

- can save money and time.
- will boost the amount of open data driven apps in your area and
- offer public sector a chance to build services together with community and businesses.
- start distributed and open development cycle in your local open data community
- result to APIs, apps, ideas, extensions and general positive developer activity around open data.

Open Data APIs and files – different audience

Building APIs will not make data dumps obsolete. Files and APIs can supplement each other. Files are good for information which are updated relatively rarely, for example bus timetables in some area. Getting bus locations via spreadsheets would not make much sense, thus APIs are often used for that purpose. Another difference between files and APIs is the audience. Spreadsheets which contain information about economical figures in some area are something that we can read and understand directly from the files. APIs are not the same. They are intended to be used by applications, not humans. Of course spreadsheets and other documents can be and often are used by applications as well, but API outputs rarely make sense to others that developers and applications.

Reasons to pursue API development

There are many advantages to having an API. Here's a few of the reasons discussed briefly.
The API provides a way to add and enhance a program without having to get into the guts of the program itself. An API allows groups of developers (internal, partners or contract developers) anywhere in the world to enhance the product with minimal coordination. The advantages go beyond better architecture of the program's source code because

It is clear that partnerships can make companies and communities alike stronger by developing synergies between products and services to each other's existing clientele. The API can help turn your competitors into partners by allowing them to build solutions on top of your product's functionality. This concept lets partners leverage the existing capabilities of your program and enhance it by using their expertise and know-how to create new and innovative solutions. This approach sure beats duplicating the functionality of your base product first.

As you have probably figured out, the API lets one build on top of a product and expands its capabilities. An API opens up the opportunity for ways to tap into markets and geographies you would not otherwise have the resources or expertise to get into. The ability to have your product "countrified" with not just translation but also with features and content relevant to that market can be a very compelling offering in a new market. An API offers the possibility of creating or modifying features to suit specific markets and it can also help automate the creation of content relevant to your market.

Simply put, an API empowers end-users by making their product do things the original developers did not build into the product. It also lets users customize the product to better fit their needs and workflow. The real secret is to create features and APIs that are not dead ends, but rather extensible solutions used to solve more specific or specialized problems with just a little bit of effort.

Eat your own dog food

In application development adding one feature to system might solve developers' problems for today. But if you provide a man an API; you have solved his problems for a lifetime. Building API for developers to consume, most likely reduces your system development costs in the long run. Providing an API is not enough though. Developers need to know how it works, what the methods return, what is the error management and so on. In other words there has to be uptodate code examples great documentation, sufficient functionality, some developer support and documentation. The process takes time to take off but once it does, it is a gift that keeps on giving.

Keep in mind that developers need to trust your API to work now and also in the future. One method to raise the trust is to follow the motto "eat your own dog food". The saying refers to using same API in your own processes, instead of building something else for others to use. Utilize and continually test your own API. If it is not good enough for you to use, you cannot expect anyone else to use it either.

APIs come in different flavours

Of course, an API strategy is not a binary decision — to use them or not to use them. There are public (open to everyone) and private (open only to certified developers or partners) APIs. There are free APIs (Google’s, for the most part, although they charge under certain circumstances) and flat fee or revenue-sharing ones (Apple’s, for the most part). There are information and services that you want to give your ecosystem access to, and those that you probably should keep to yourself. In short, your organization needs to debate the various elements of an API strategy, and then you need a set of governance mechanisms to enforce it.

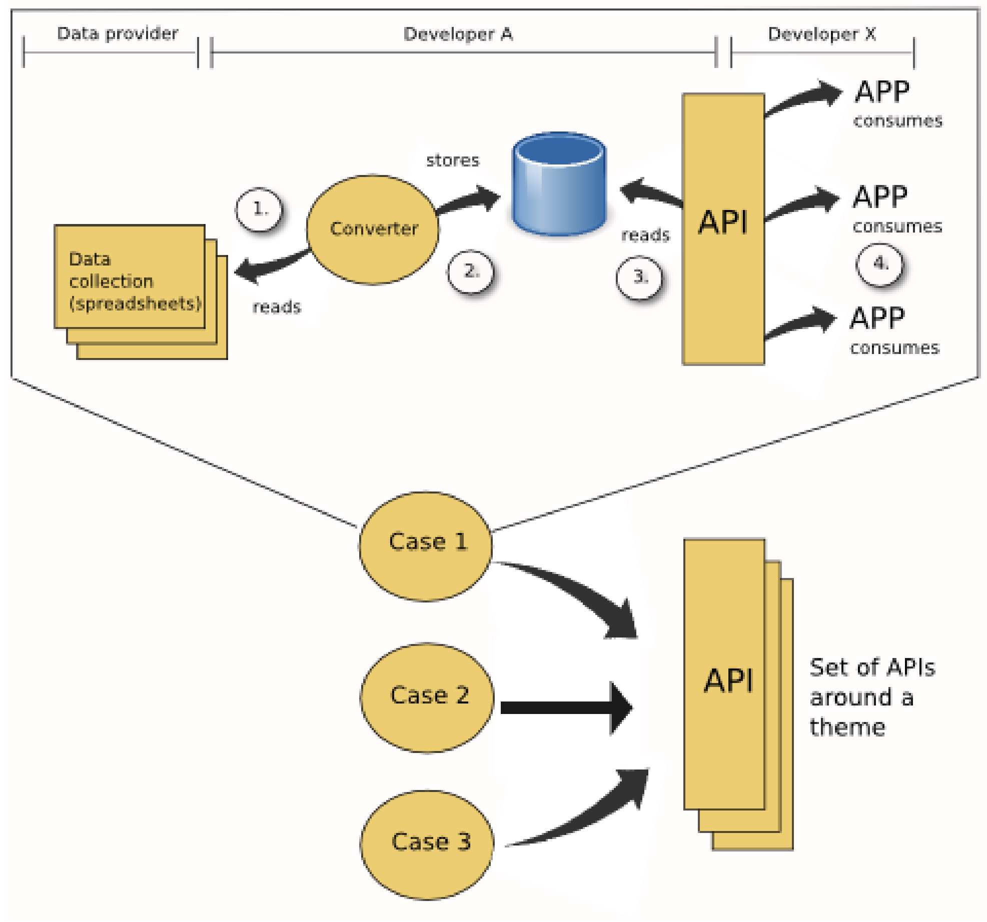
If we intend to build open data ecosystems and subsystems, APIs are the way to do it. Information shared is value squared! Often governmental organizations do not cheer loudly when they are asked to build APIs because they are often expensive although in the long run APIs 'pay back'. How to tackle expenses?

Another option is to enable and engage community driven development. Open Data communities around the world are filled with talented developers and they represent a pool of resources. You need to give something to the community and they will come. You need to give them freedom, access to data source (aka open the data) and if possible a “playground”. Playground refers to tools and servers in which community has test new ideas and build applications and APIs. Communities might even build the playgrounds themselves if you provide some resources. Let's take a concrete example.

Enable community driven development

In some cases systems which contain the information itself are proprietary and thus giving access to community developers is not possible. This is the situation due to agreements created in the past when open data was not known inside public sector or other organizations.

Therefore excel files are dumped (and converted to CSV) out of the system. Even enabling that capability costs money, since public sectors are in vendor lock-in situation. That refers to situation in which service provider is the only one allowed (or capable of) to create extensions to existing system in use. But if the data is available as data dumps, it's possible to engage community to define and build APIs to the data.



In the above sketch, some governmental organization has opened economical data as spreadsheets (case 1). Without APIs getting that information to each application developed by different developers would require that every data input method (converter) is coded separately to each application. Each developer has to dig deep into the spreadsheets and use innovation and programming skills to obtain needed parts of the information inside the document.

What if one developer creates one converter which is able to read the data and store it to community-wide shared database? And then the same or some other developer creates a light-weight API on top of that data. Better yet, the API is put into community server for all to use. Then all developers could access the spreadsheet information via the API. Of course there are ways to read data directly from the spreadsheets, but they are often not as fast and efficient as using databases. One of the advantages of API is that spreadsheet does not need to be downloaded to the app, application retrieves just the information it needs, nothing more nothing less. In the ideal world source code for the newly build API is shared via GitHub or alike thus enabling incremental development.

In case 2 some governmental organization has shared information about staff, wages and so on. If the same process is applied to case 2 as was in case 1, we could easily have a set of APIs which offer efficient methods to access information inside spreadsheets. And what would all this cost to organization that opened the spreadsheets? Most likely much less than ordering system integrated API from the same supplier that provided to background system to store spreadsheets. In addition, community driven approach would not affect the workload of the data opener organization.