Emerging Perspectives to API Strategy

Juho Lindman*, Imed Hammouda†, Jennifer Horkoff † and Eric Knauss†
*University of Gothenburg and Chalmers University of Technology
Gothenburg, Sweden, Email: juho.lindman at ait.gu.se
†Chalmers University of Technology, Gothenburg, Sweden

Abstract—Software specialists find themselves increasingly in situations where their decisions related to implementing effective and efficient API strategies have strong implications on business decisions. Through a long-lasting research collaboration with software specialists responsible for APIs in large industrial software-intense products from several companies we have established an analytical framework, reported here. We combine earlier work on layering of digital technologies and governance to provide a multidisciplinary framework that is of interest for professionals working with API design and development.

Index Terms—API; management; layering; governance; BAPO

1 Introduction

Business needs are increasingly driving force in the development of Application Programming Interfaces (APIs) that allow downstream developers to access (business) assets, for example data and services. Developers may use an API internally or APIs can be open to external third-parties. API providers expose assets to developers in order to produce value through different applications that rely on the API - monitoring and controlling access to some degree.

The benefits of APIs have been documented in numerous studies [1]. APIs provide a useful interface for service provision, customer access and third-party development. The use of APIs provides not only control, but also stability to encourage use and reduce abuse. In addition to solving technical challenges related to API management, research has discussed related organizational and business issues in different industries [2], [3].

APIs are more and more considered as an artifact that has to be continuously enhanced [4]. Further research has considered API “learnability” [5]. Despite these advances, more information is needed about challenges and best practices of API design and management in organizations.

Several reports offer useful practical design considerations for APIs, including advice on collecting usage data, monetization strategies, and at what point to open an API to external parties [6], [7], [8], [9]. However, such practical considerations do not make use of wider knowledge of established analysis frameworks. While companies currently have clear product strategies in place, we are just now starting to see systematic research-informed approaches towards understanding and managing API strategies in commercial organizations.

In this work, we report our work with companies to build a framework that synthesizes and summarizes API strategies. We incorporate frameworks beneficial for strategic API design, including resource governance [10] and BAPO (Business, Architecture, Process, and Organization) development perspectives [11]. Our research work was carried out within an industry-academia collaboration of the Software Center (See Table1). We show how conceptual frameworks can be used to drive API strategy development, drawing on our research work in organizations (Table2) with several companies in practice (Table2).

Table1. Software Center

Software Center (http://www.software-center.se) is an industry-academy collaboration aimed to increase the competitiveness of Nordic Software Engineering hosted by the Department of Computer Science and Engineering at the Chalmers University of Technology and University of Gothenburg in Sweden. Several other universities are also part of the center that primarily works with a selected set of partner companies. The work in Software Center is organized into dedicated projects where work is carried out with the industrial partners in half-year sprints. The results of the projects address the immediate concerns of these companies. Research has been carried out in the API Strategies Project (Project 26), which focuses on APIs as key enablers for innovation by providing mechanisms for different applications to hook up and integrate, generating new value for businesses and customers. Keeping up with ever-changing market requires a well-defined API strategy in terms of identified processes, methods, and instruments to manage the API value chain.

Table2. Methodology and case companies

The research employed multiple case study methodology. The case data was collected from workshops, interviews and thematic discussions with experts from four companies in the embedded systems industry. We worked with the companies developing frameworks for their immediate needs regarding management of APIs.
We first collected perceptions of the key stakeholders, then combined these insights with knowledge gained via literature reviews, providing feedback to organizations concerning their specific challenges in several consecutive workshops and iterations.

Company 1:
Company 1 is global firm operating in the area of network video cameras, currently providing network video products which are installed in public spaces (for example train stations and universities) and business areas (for example casinos and retail stores). The company adds value to their cameras by providing an API to enable their customers create their own applications where they use the generated camera data. Planned provision of a Cloud API will serve the needs of customers, increase customer satisfaction and customer retention.

Company 2:
Company 2 is a large telecommunications company with more than 110,000 employees. Company offer services, software and infrastructure in information and communications technology (ICT) for telecommunication and net working equipment. The study was conducted in cooperation with the Product Development Unit which was working on developing strategies for specific software frameworks related to state, fault and alarm handling.

Company 3:
Company 3 is an international company developing different mechatronic devices for its global customers around the world and is moving towards the implementation of a new suggested API ecosystem workflow, is expected to solve the bottlenecks in their current business process. The company wants to be able to analyze the effects these changes have on their API ecosystem design.

Company 4:
Company 4 provides primarily packaging for liquid and food products but also a range of processing and packaging technologies in a broader array of products. Company supplies databases that contain information their customers use to generate reports about manufacturing tasks, e.g., quality control reports. In order to make the generation of these reports easier, the organization aims provide an API to access the needed data easily and generate reports much faster.

2 API and Digital Innovation

We draw on earlier knowledge on digital innovation and approach APIs as *digital innovation objects* [12], specifically as one layer within a larger context. Figure 1 presents APIs in relation to other relevant layers. We also note the boundary objects between the layers which will impact an organization's strategy for their APIs.

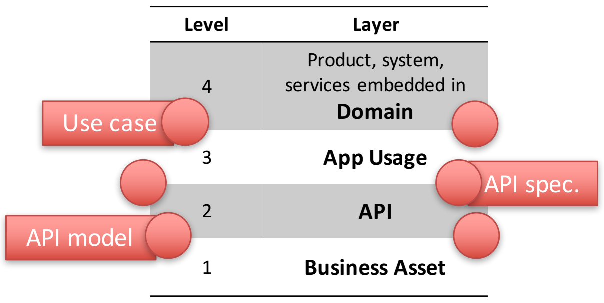


Fig. 1. APIs as Digital Innovation Objects with relevant Layers (rows) and Boundary Objects (dots between rows).

2.1 Background: APIs as Digital Innovation Objects

Digital objects (such as APIs) are interactive and editable due to their implicitly multi-layered architecture [13]. By this, they offer plenty of business opportunities and service design potential, similar to Yoo et al.'s illustration of a loosely coupled layered architecture for digital technology [12]. According to these authors, such multi-layered architectures can contain four different layers of devices, networks, services, and contents.

Loose coupling of the layers means that independence between the layers enables that the design decisions can be decoupled between the layers. More specifically, this also makes it possible to decouple the decisions related to the access and ownership between the different layers. Another benefit that comes from the separation of layers is the ability to innovate at the level of the individual layer instead of the whole architecture. This also means that companies can aim to attract different stakeholders to participate to the design and ultimately use of the company-controlled platform resources [12].

In order to benefit from this layered approach when reasoning about API strategies, four relevant layers have emerged from our empirical results. In addition, we have identified three kinds of boundary objects, i.e. artifacts that are of interest for two adjacent layers. In this discussion, we use an illustrative example of a surveillance camera system and present typical questions that we raised during our discussion with the case companies.

2.2 Relevant Layers for API Strategies

APIs offer decoupling between business assets and application software. However, when reasoning about API strategies, it is important to also investigate the domain if the application software.

2.2.1 Domain layer

The domain layer offers certain needs and events that are ideally supported by application software. In our example, such an event could be the decision of an organization to support their business goals through reorganizing, assumingly based on knowledge about the number of people using a building. Perceiving the domain as one relevant layer for APIs allows to address important *concerns for API strategies*. *Typical questions* that should be answered include: What is the domain? What business cases exist? What domain events can be anticipated?

2.2.2 App usage layer

Suited on top of an API, App usage sage realizes user visible features. An example is an App that counts individuals, which is installed on a network attached camera. Since the App SW interacts directly with one or several APIs, this layer raises concerns that need to be addressed in an API strategy. Typical questions with respect to this layer include: Is App SW developed by internal or external developers? Is App SW usually slim or thick (e.g. is the App is based on simple calls of a use case centric API, which it exposes to end-users, or does it have significant logic)? Among others, these aspects impact how users of the API react to updates.

2.2.3 API layer

An API allows access to one or more business assets. In the scope of the running example a network attached Camera could have an API to allow extensions on the camera itself. The main *concerns* on this layer are decisions with respect to the API design, which should be based on strategic considerations derived from knowledge about other layers as well as from boundary objects between the layers. *Typical questions* include: Should the APIs be designed as chatty or chunky? Should the API respond synchronously or asynchronously to requests? To what extent should the API provide protection from unauthorized access, use, disclosure, disruption, modification, or destruction?

2.2.4 Business asset layer

The business asset layer covers *concerns* related to business assets of a company, such as properties of a product, core algorithms, and data. In our example this is a network attached security camera and the associated video data it may produce, including the meta-data which can be extracted from the videos (e.g., traffic levels). With respect to API strategy *concerns*, the business asset layer raises the following *typical questions*: Are there new business assets emerging? Can existing ones be exposed better? Are assets nearing the end of their lifecycle?

2.3 Boundary Objects

If a Boundary object changes, both adjacent layers are affected. With respect to API strategies and challenges, we identified general characteristics of such boundary objects. API related boundary objects are visible in the organizations when taking decisions, because an API strategy can only be based on boundary objects that are known. Many problems in managing APIs can be avoided if one party does not unilaterally change a boundary object without prior coordination with other parties. Here, we give examples of three common boundary objects relevant for API strategies:

2.3.1 Use cases

A use case describes how actors could reach their (domain) goals based on available features. An example with some specific set of features in mind could be to identify individuals and count them across several cameras distributed over a building. API strategy *concerns* raise *questions* like: Which use cases exist? How could use cases change (triggered by Domain or App SW)? What is the impact of change?

2.3.2 API specification

A contract between app developer and API provider. In our example this might be a defined way to access raw data stream for data analysis. With respect to an API strategy, an important *question concerns* the expected impact of or resistance to change.

2.3.3 API model

Describes relevant aspects of business assets to be exposed. In our example these aspects include the ability to tilt, zoom, access raw data stream, but probably not the color of the camera's casing. *Questions* that relate to API strategy concerns include: Are all important aspects included? Is the model effective (does it allow to develop App SW that addresses important use cases)? Is the model efficient (does it allow addressing relevant use cases in a way that uses reasonable amounts of resources)?

3 APIs AND GOVERNANCE

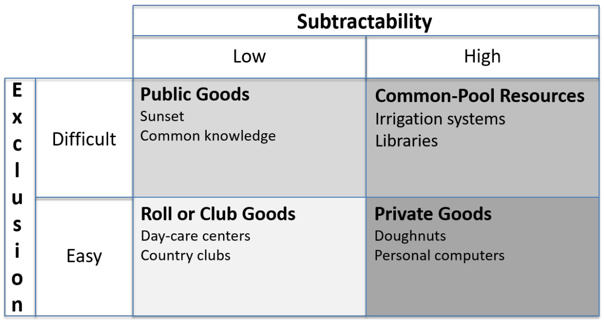
Governance issues, including access control, openness, and resource management are increasingly important issues in API strategy. Analyzing APIs in term of strategic layers raises several questions related to such governance issues. Different issues arise per layer (business assets, APIs, App usage and domain). Companies take decisions on which kinds of governance arrangements support their business goals at each layer.

3.1 Background: APIs as Resources

Our industrial partners have indicated that governance issues are of importance when considering API strategies. Ostrom's work on common pool resources [14] provides rich source of theories related to the issues of collective action framed in terms of governance, openness, ownership, and economic subtractability (rivalry) of the different social situations where governance is needed. Her theory of governance, however, posits that in terms of governance there are not only theoretically different types of goods, but behavior in situations related to provision of one or the other type of good is substantially different [14].

In general, there are two main criteria of collective action problems: one related to exclusion and one related to subtractability. Exclusion is determined by how easy it is to prevent access to the good (e.g. fishery, digital artifact) in question. Subtractability means how deleterious the uncooperative actions of the participants are to other participants.

Along those two dimensions, Ostrom identifies four types of goods, shown in Figure 2 [10]. Private goods such as doughnuts and personal computers are both excludable



		Subtractability	
		Low	High
Exclusion	Difficult	**Public Goods** Sunset Common knowledge	**Common-Pool Resources** Irrigation systems Libraries
	Easy	**Roll or Club Goods** Day-care centers Country clubs	**Private Goods** Doughnuts Personal computers

Fig. 2. Ostrom's theory of types of goods [10].

(other individuals are excluded from consuming) and rivalrous (whatever consumed, no one else can consume). Club goods such as day-care centers and country clubs are excludable (i.e. good accessed by club members only) but non-rivalrous (goods are available to all club members). Public goods such as sunset and common knowledge are characterized by the difficulty of exclusion (accessible to all) and low rivalry (consumption of a good does not limit the consumption by others). Finally, common-pool resources are those that are characterized by both difficulty of exclusion and difficulty of subtraction. They are also threatened by overuse and depletion.

Figure 2 was instrumental for discussing API governance. In fact, we have used this figure to outline API governance as collective action problem faced in each of the layers identified in Figure 1.

3.1.1 *Exclusivity (Access)*

Private goods are those that are limited to specific users. Club goods are those that are limited to a specific groups of users. Limiting deleterious access is no longer a main threat that should be stopped by exclusion. Instead governance is increasingly necessary to secure and manage third party contributions. For example Open Source Software, Open APIs, and Open Data assets are good examples of a situations where exclusion is not seen as the best strategy. APIs can of course technically be opened up for third-party or external contributions. In those situations managers controlling the APIs have chosen not to prevent access to the resource, but rather to attract innovation to the platform.

3.1.2 *Subtractability (Rivalry)*

Private goods (for example business assets) mean that their owners can exercise property rights and to withdraw the right to use the good. Also, in common-pool resource situations one actor's aggressive withdrawals can generate high costs for everyone else [10]. For club goods, the withdrawals do not prevent other actors from still using the good. In common-pool resource situations, members have incentives to limit the number of actors who can access the resource. In contrast, in public good situations the group tries to extend its membership base [10].

3.2 API Governance and Layers

The different situations requiring governance lead to different governance mechanisms. This means that when companies make decisions on what kind of governance is needed in their situation they have options. For example, a company may want to keep their business assets private and not disclosed to the general public. Access to selected parts of the assets is exclusively granted through API calls. On the other hand, a club goods type of scenario is adopted at the API layer by involving selected partners in API roadmap and evolution. Further, the company might release some part of the application stack as open source to attract third-party contributions. Finally, the company may set a QoS priority policy for bandwidth consumption. This is an example commons-pool scenario where bandwidth consumption by some API clients may degrade the QoS for other clients.

Independence of the layers also means that a company does not have to select one type for all layers. For example parts of the assets can remain private and some are made more open.

4 BAPO

All of our company respondents have emphasized that API development should not be an accidental or ad-hoc activity, but rather a systematic process supported by an integrated environment of policies, methods, tools, resources, etc. America et al. [11] argue that for a development method to achieve the best possible fit, four interdependent software development concerns shall be considered: Business, Architecture, Process, and Organization (often referred to as the BAPO model). Such a model helps to consider a wide range of strategic concerns in API strategy design.

4.1 Background: APIs from BAPO Perspective

From the lens of BAPO, one could identify the following kinds of challenges related to API design and development. We illustrate the discussion with example questions raised by our industrial partners.

4.1.1 *Business*

This perspective addresses *concerns* of making APIs a business capability and a business development model. *Typical questions* include: How API design drives business level decisions such as vendor selection and how to generate business value out of API usage?

4.1.2 *Architecture*

The main *concern* of this perspective is to investigate the technical issues associated with API design and development. *Typical Questions* include: How to manage API versioning (e.g. side-by-side deployment of different versions)? How to design APIs for extension? How to check backward compatibility of APIs between different versions?

4.1.3 *Process*

This perspective raises *concerns* related to identifying roles, responsibilities and relationships in order to respond and act more effectively within the API team. *Typical Questions* include: How to build a governance strategy for APIs that takes into consideration design phase issues (e.g. interface definition, quality trade-offs, reuse, documentation)?

4.1.4 Organization

This perspective covers *concerns* related the organizational aspects of API strategy, i.e. mapping the identified roles and responsibilities to existing organizational structures. *Typical questions* include: How to achieve better alignment between interacting organizational architecture units? How to build an effective API team? How to adapt existing organizational frameworks to the API context?

4.2 API Strategy: BAPO, Governance and Layers

We observe that BAPO questions can be discussed with regard to the different API layers identified in Section 2 and the social situations discussed in Section 3.

Bringing all three conceptual models together (i.e. layers, types of goods, BAPO), Figure 3 presents a unified framework for API strategy building. The framework is interpreted as follows. Any business (architectural, process, or organizational) matter could be considered and decided at each layer (i.e. Business Asset, API, etc.) separately and taking into consideration the desirable social situation (i.e. protection, collaboration, competition, etc) that the company wants to create. We argue that the framework could be used as an analytical tool to help companies analyze, develop, and evaluate their API strategies.

5 Conclusion and Outlook

In this paper, we present a multidimensional conceptual framework which has emerged from collaborative research on API strategies with several large companies. We found that this framework helps managers, designers, and developers to gain insights in emerging and novel trends related to API management. In practice, a company might need to apply the conceptual framework differently depending on forces such as whether the company has one or more APIs, or whether part of the API is strategic while other parts are more opportunistic and temporary.

We continue to apply and improve the framework with our partner companies in the software center, furthering our understanding of which existing and planned API elements fall within each layer, revealing gaps in practice. We invite others to contribute with their research or experience to our framework and the important field of strategic API design.

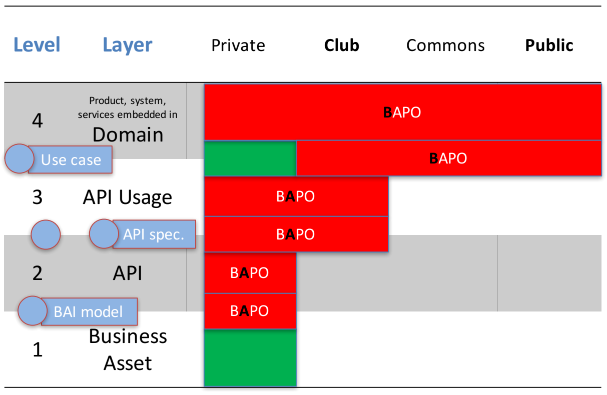


Fig. 3. Example: Layers, Governance and BAPO.

References

[1] C. R. B. de Souza and D. F. Redmiles, "On the roles of apis in the coordination of collaborative software development," *Computer Supported Cooperative Work (CSCW)*, vol. 18, no. 5, p. 445, 2009.

[2] T. Aitamurto and S. C. Lewis, "Open innovation in digital journalism: Examining the impact of open apis at four news organizations," *New Media & Society*, vol. 15, no. 2, pp. 314–331, 2013.

[3] F. Bolici, J. Howison, and K. Crowston, "Stigmergic coordination in FLOSS development teams: Integrating explicit and implicit mechanisms," *Cognitive Systems Research*, vol. 38, pp. 14 – 22, 2016, special Issue of Cognitive Systems Research Human-Human Stigmergy.

[4] I. Hammouda, E. Knauss, and L. Costantini, "Continuous api-design for software ecosystems," in *Proc. of 2nd Int. WS on Rapid and Continuous Software Engeering (RCoSE '15 @ ICSE)*, Florenz, Italy, 2015.

[5] M. P. Robillard, "What makes apis hard to learn? answers from developers," *IEEE software*, vol. 26, no. 6, 2009.

[6] Apigee, "Is your API naked? 10 roadmap considerations for API product and engineering managers," 2010. [Online]. Available: https://apigee.com

[7] Nordic API, "Developing the API mindset: A guide to using private, partner, & public APIs," 2015. [Online]. Available: https://nordicapis.com

[8] IBM Institute for Business Value, "Evolution of the API economy. adopting new business models to drive future innovation," 2016. [Online]. Available: https://www.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=GBE03759USEN

[9] Oracle Communications, "Making money through API exposure. enabling new business models," 2014. [Online]. Available: http://www.oracle.com/us/industries/communications/comm-making-money-wp-1696335.pdf

[10] E. Ostrom, "How types of goods and property rights jointly affect collective action," *Theoretical Politics*, vol. 15, no. 3, pp. 329–370, 2003.

[11] P. America, H. Obbink, R. van Ommering, and F. van der Linden, *CoPAM: A Component-Oriented Platform Architecting Method Family for Product Family Engineering*. Boston, MA: Springer US, 2000, pp. 167–180.

[12] Y. Yoo, O. Henfridson, and K. Lyytinen, "The new organizing logic of digital innovation: An agenda for information systems research," *Information Systems Research*, vol. 21, no. 4, pp. 724–735, 2010.

[13] J. Kallinikos, A. Aaltonen, and A. Marton, "The ambivalent ontology of digital artifacts," *MISQuarterly*, vol. 37, no. 2, pp. 357–370, 2013.

[14] E. Ostrom, R. Gardner, and J. Walker, "Rules, games and common-pool resources," University of Michigan Press, Ann Arbor:MI, Tech. Rep., 1994.