

# Identification of API Management Patterns From an API Provider Perspective.

Andre Landgraf, 20.10.20, Master's thesis introduction

Chair of Software Engineering for Business Information Systems (sebis)  
Faculty of Informatics  
Technische Universität München  
[www.matthes.in.tum.de](http://www.matthes.in.tum.de)

# Outline

- Motivation
- Research questions
- Approach
- Interviews
- Pattern Candidates
- Timeline

# Motivation

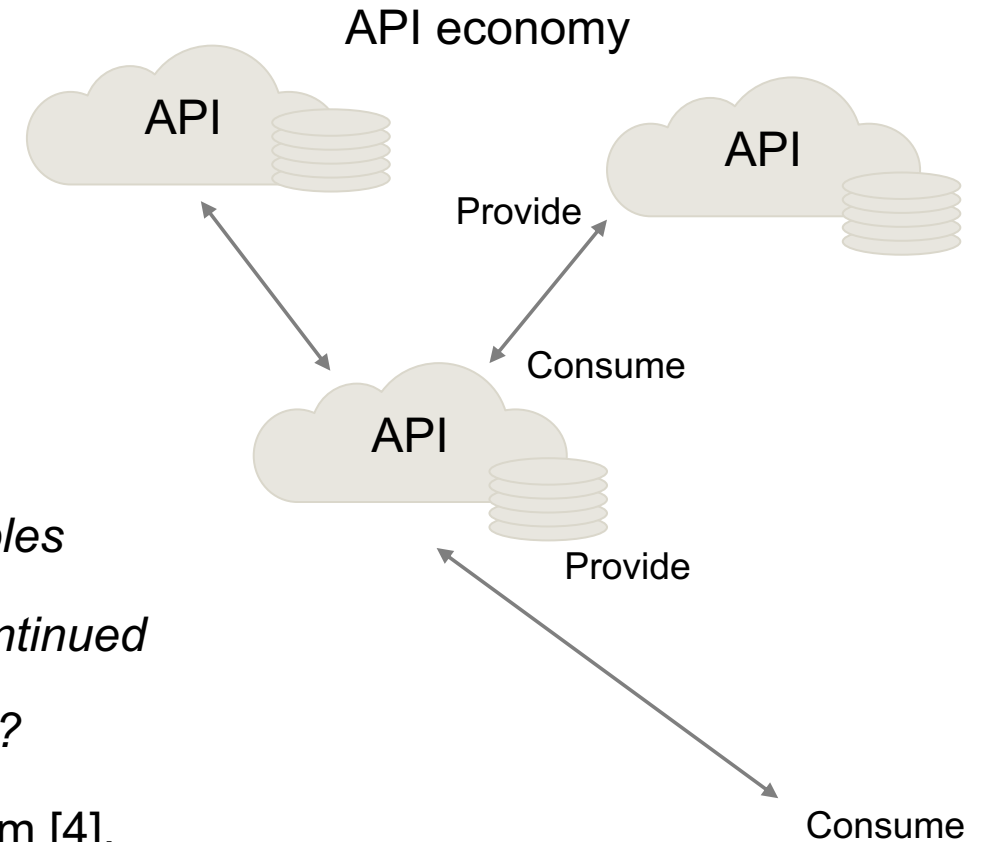
## API economy and the layered modular architecture

- Value-adding through composition [1, 2].
- Emergence of the API economy [1, 2].
- Instigation of the layered modular architecture [3].

- Yoo et al., 2010:

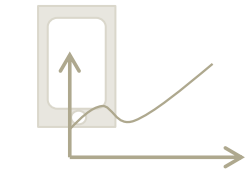
*What are the appropriate methodological and technological principles of the design of technical boundary resources that help sustain continued developments of novel components in doubly distributed networks?*

- In need for longitudinal studies on the evolution of digital platform [4].



# Motivation

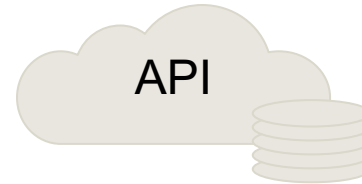
## API governance and management literature



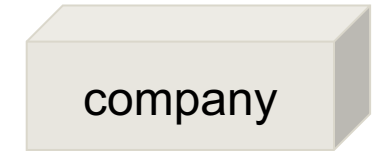
Use case



API consumer



API provider



Automation of API changes [5]

API governance strategies & pattern [10]

Consumer behavior on API evolution [6]

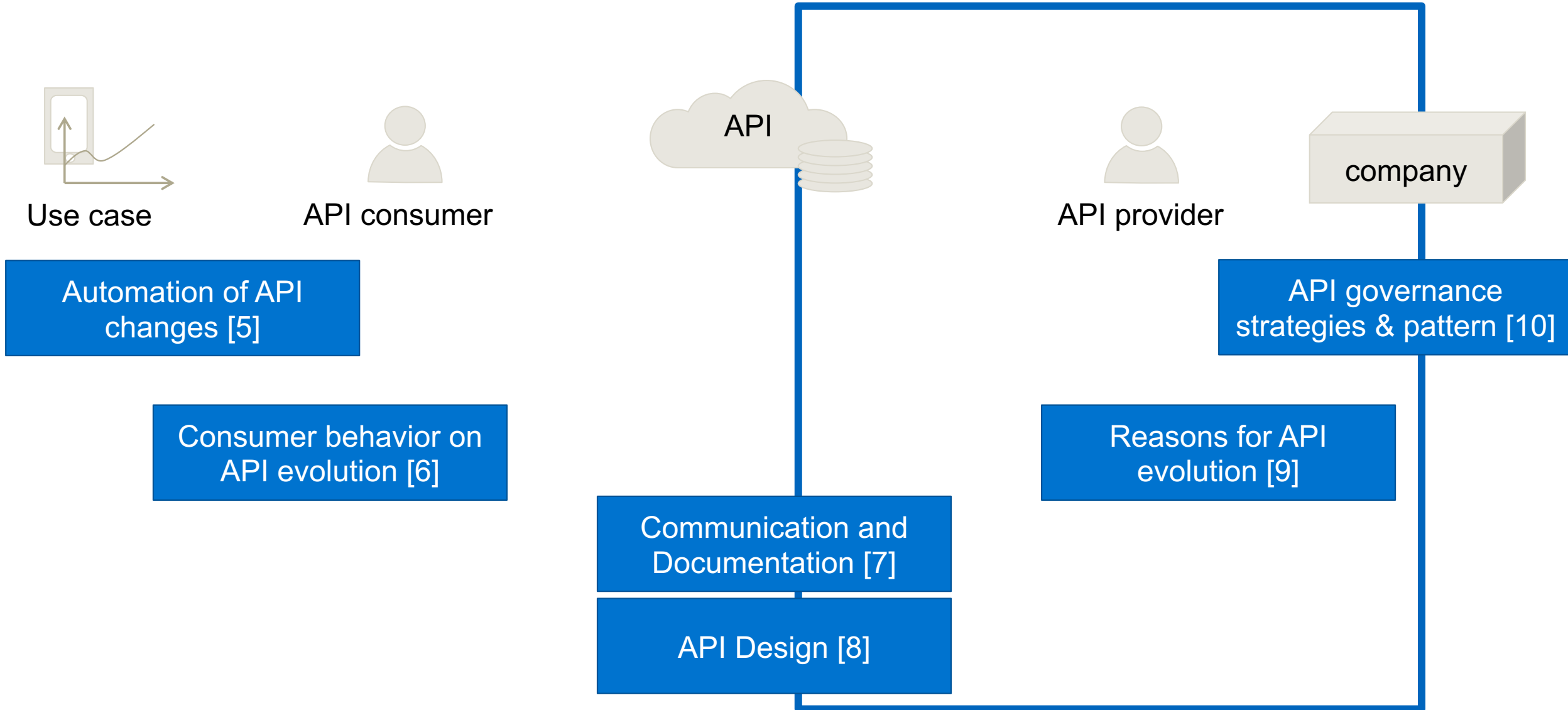
Reasons for API evolution [9]

Communication and Documentation [7]

API Design [8]

# Motivation

## API governance and management literature



[5] Perkins, 2005 | [6] Espinha et al., 2014 | [7] Sohan et al., 2015 | [8] Haupt et al., 2017 | [9] Hou et al., 2011 | [10] Lübke et al., 2019

**RQ1:** Based on longitudinal data, what issues do API providers face on their daily work? [what, who, when]

- *Result:* List of issues with information about: assignee, task, assigned points, category, reporter, start/end.

**RQ2:** How have the issues been resolved or what is blocking the issues? [how]

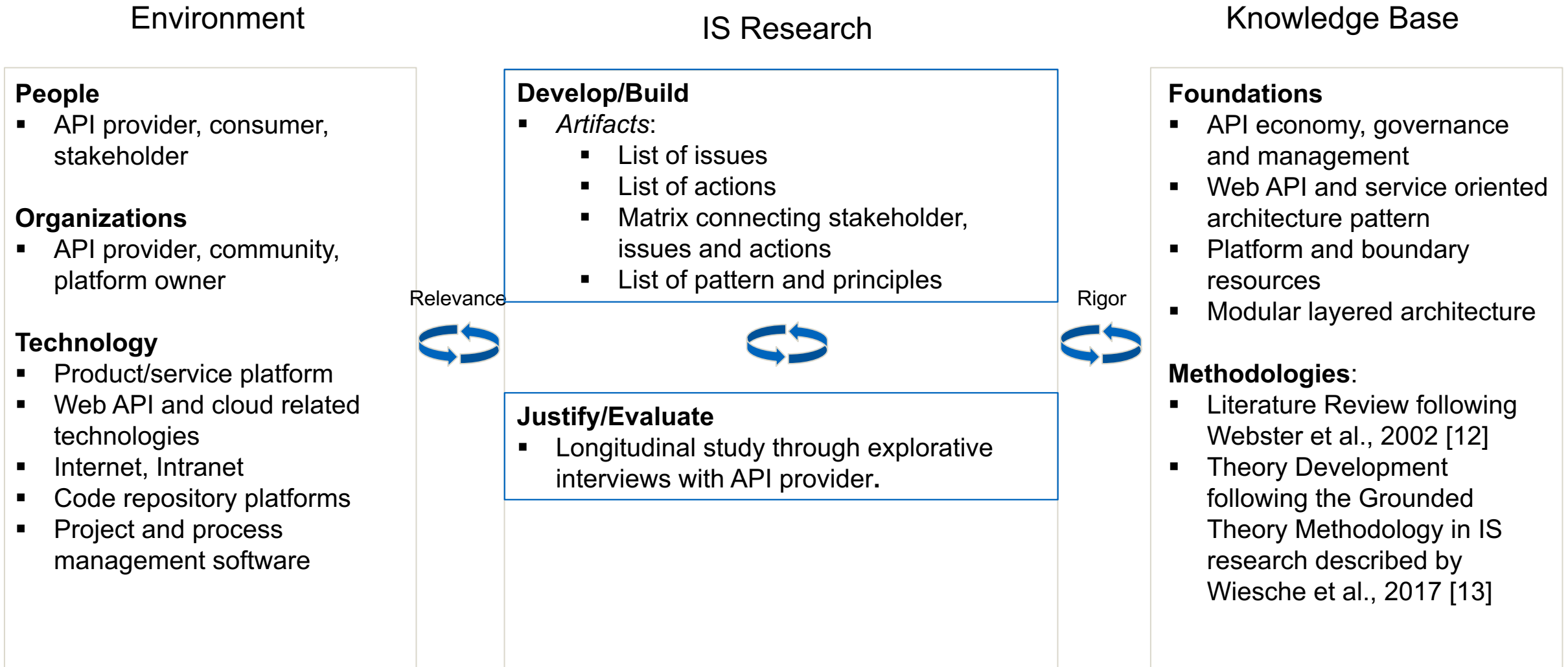
- *Result:* List of actions, action categories, involvement of third-parties (e.g. beta tests) and reference to issues.

**RQ3:** What is the rationale behind the actions and API changes? [why]

- *Result:* Matrix that connects actions (RQ2), stakeholder, reasoning/prioritization and the issue (RQ1).

*Final Result: List of detected pattern and principles based on the findings from RQ1-3.*

# Approach - Design Science Research [11]



[11] Hevner et al., 2004 | [12] Webster et al., 2002 | [13] Wiesche et al., 2017





# Semi-structured interviews with API providers

Number	Classification	Role	Employees	Duration	Participants	API Types	Maturity
1	Multi-banking startup	Backend Developer	11-50	00:22:52	IV1	Private	Production
2	Industrial manufacturing	Internal Consulting	>100.000	00:44:09	IV2	Public & Partner	Pilot
3	Automotive	Product Owner	>100.000	00:48:49	IV3, IV4	Public & Partner	Production
4	Software & IT service provider	Software Architect	1001-5000	00:42:25	IV5	Public & Partner	Production
5	IT service subsidiary	Portfolio Manager	1001-5000	00:51:12	IV6	Private	Production
6	Insurance subsidiary	Enterprise Architect	51 - 250	00:59:28	IV7	Private	Pilot
7	Industrial manufacturing	Service Owner	>100.000	00:46:34	IV8	Private	Development
8	Industrial manufacturing	Senior Software Architect	>100.000	00:47:03	IV9	Public & Partner	Production
9	Financial services	Senior Software Developer	10.001-50.000	00:35:25	IV10	Public & Partner	Production
10	Software & IT service provider	Internal Consulting	5001 - 10.000	00:50:49	IV11	Public & Partner	Production

\*Encoded using MAXQDA



 <b>Stakeholder</b>	 <b>Concern</b>	<b>API Management Pattern</b>
API Provider	How can I manage incoming feature requests?	Pattern IP1: Organization-wide Ticket System
API Provider	How can I provide support for API consumers?	Anti-Pattern CP1: API Provider Bottleneck

# Pattern IP1: Organization-wide Ticket System



**Category:** Issue Prioritization Pattern

**Connected concerns:**

- How can I provide support for API consumers? (C1)
- How can I manage incoming feature requests? (C2)
- How can I resolve bug reports effectively? (C3)

**Description:**

Since APIs are part of distributed systems, API providers have to collaborate with many different stakeholders and the contact might not always be direct. Stakeholders outside the company might interact with customer support, sales, and other teams. Feedback, feature requests, bug reports, and questions will arrive through different channels and have to find their way to the API provider. It is crucial that all incoming issues contain all relevant information and that no data is lost along the line. Incomplete or misleading issues can result in time consuming engagements or stop the issue from being worked on. It is important that all upcoming issues are tracked and communicated in an efficient and effective manner.

**Solution:**

Organization-wide ticket systems allow formal and explicit issue communication across team and business unit boundaries. Tickets track the history, progress, and relevant people. If a ticket reaches the API provider, the product owner can prioritize the ticket within the context of its backlog. This way, all requests can be handled in a unified manner. A number of questions--who reported the issue? who is interested in the issue? where can I ask follow-up questions?--can be answered easily without management overhead.

# Anti-Pattern CP1: API Provider Bottleneck

**Category:** Collaboration Pattern

**Connected concerns:**

- How can I provide support for API consumers? (C1)

**Description:**

API providers have to effectively and efficiently support API consumers in the usage of the API. Support requests might be presented to the API provider through channels such as customer support emails, Twitter messages, or GitHub issues. The cloud infrastructure of the API might be able to scale with a growing number of API consumers but the API provider team can only provide so much support. If the API provider tries to support a growing number of API consumers while maintaining its original team size and other activities, the API provider is in danger of creating a bottleneck.

**Revised solution:**

The API provider has several possible solutions to handle the growing number of support requests. A good starting point can be creating and growing an FAQ page (see Pattern AP1). If the API grows bigger, the API provider might need to collaborate with different functional teams (e.g. customer support or sales) to tackle the support of the API consumers and to separate first and third level support (see Pattern CP2).

# Timeline - Outlook

Activity/Month	August	September	October	November	December	January	February
Kickoff							
Literature review							
Interviews							
Evaluate interviews							
Writing thesis							
Submission							

## Timeline – Next steps

**Goal:** List of detected pattern and principles based on the findings from RQ1-3.

**Done:**

- First literature reviews and research of related literature.
- Kickoff interviews with ten API providers.
- Transcription and encoding of the ten interviews.
- First iteration of a pattern language and a list of possible pattern candidates

**Next steps:**

- Schedule second round of interviews for follow up questions and pattern evaluation
- Re-iterate and refine current the pattern catalogue

- [1] Bondel, Gloria & Nägele, Sascha & Koch, Fridolin & Matthes, Florian. (2020). Barriers for the Advancement of an API Economy in the German Automotive Industry and Potential Measures to Overcome these Barriers. 727-734. 10.5220/0009353407270734.
- [2] W. Tan, Y. Fan, A. Ghoneim, M. A. Hossain and S. Dustdar, "From the Service-Oriented Architecture to the Web API Economy," in IEEE Internet Computing, vol. 20, no. 4, pp. 64-68, July-Aug. 2016, doi: 10.1109/MIC.2016.74.
- [3] Yoo, Youngjin & Henfridsson, Ola & Lyytinen, Kalle. (2010). The New Organizing Logic of Digital Innovation: An Agenda for Information Systems Research. Information Systems Research. 21. 724-735.
- [4] Reuver, Mark & Sørensen, Carsten & Basole, Rahul. (2017). The digital platform: a research agenda. Journal of Information Technology. 10.1057/s41265-016-0033-3.
- [5] Jeff H. Perkins. 2005. Automatically generating refactorings to support API evolution. SIGSOFT Softw. Eng. Notes 31, 1 (January 2006), 111–114.
- [6] T. Espinha, A. Zaidman and H. Gross, "Web API growing pains: Stories from client developers and their code," 2014 Software Evolution Week - IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE), Antwerp, 2014, pp. 84-93.

- [7] S. M. Sohan, Craig Anslow, and Frank Maurer. 2015. A Case Study of Web API Evolution. In Proceedings of the 2015 IEEE World Congress on Services (SERVICES '15). IEEE Computer Society, USA, 245–252.
- [8] Florian Haupt, Frank Leymann, and Karolina Vukojevic-Haupt. 2018. API governance support through the structural analysis of REST APIs. *Comput. Sci.* 33, 3–4 (August 2018), 291–303.
- [9] D. Hou and X. Yao, "Exploring the Intent behind API Evolution: A Case Study," 2011 18th Working Conference on Reverse Engineering, Limerick, 2011, pp. 131-140.
- [10] Daniel Lübke, Olaf Zimmermann, Cesare Pautasso, Uwe Zdun, and Mirko Stocker. 2019. Interface evolution patterns: balancing compatibility and extensibility across service life cycles. In Proceedings of the 24th European Conference on Pattern Languages of Programs (EuroPLop '19). Association for Computing Machinery, New York, NY, USA, Article 15, 1–24.
- [11] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. 2004. Design science in information systems research. *MIS Q.* 28, 1 (March 2004), 75–105.
- [12] Webster, Jane & Watson, Richard. (2002). Analyzing the Past to Prepare for the Future: Writing a Literature Review. *MIS Quarterly*. 26. 10.2307/4132319.
- [13] Wiesche, M., Jurisch, M., Yetton, P., & Krcmar, H. (2017). Grounded Theory Methodology in Information Systems Research. *MIS Q.*, 41, 685-701.





B.Sc.

**Andre Landgraf**

Technische Universität München  
Faculty of Informatics  
Chair of Software Engineering for Business  
Information Systems

Boltzmannstraße 3  
85748 Garching bei München

Tel +49.89.289.17143

Fax +49.89.289.17136

andre.timo.landgraf@gmail.com  
[www.matthes.in.tum.de](http://www.matthes.in.tum.de)



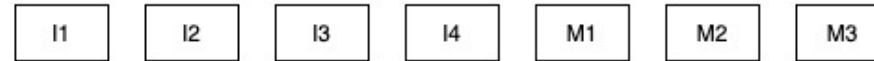
# Backup

# Do you have a list of all current pattern candidates?

Entity	Instance I	Instance II	Instance III	Instance IV
<b>Influence factors (I)</b> <b>Maturity levels (M)</b>	- Public API (I1) - In Production (M3)	- Partner API (I2) - In Pilot (M2)	- Private API (I3) - Service Agreement (I4)	- Private API (I3) - In Production (M3)
<b>Stakeholders (S)</b>	API Provider (S1)	API Provider (S1)	API Provider (S1)	API Governance (S2)
<b>Concerns (C)</b>	How can I provide support for API consumers? (C1)	How can I manage incoming feature requests? (C2)	How can I resolve bug reports effectively? (C3)	How can I take advantage of new technological improvements? (C4)
<b>Collaboration Pattern (CP)</b>	Anti-Pattern: API Provider Bottleneck (CP1)  Dedicated First-Level Support (CP2)	Pilot Projects (CP3)  Anti-Pattern: Blocking API Co-Creation (CP4)	Contractual Punishment (CP5)	API Provider Technological Lead (CP6)
<b>Issue Prioritization Pattern (IP)</b>	Organization-wide Ticket System (IP1)	Backlog Refinement Meetings (IP2)	Lean Kanban System (IP3)  Bugs first, Feature second (IP4)	-
<b>Action/Adaption Pattern (AP)</b>	Growing FAQ (AP1)  API Playbook (AP2)	-	Scrum Master Escalation (AP4)	Technology Push (AP5)  Platform Unification (AP6)
<b>Artifacts (A)</b>	FAQ (A1) Playbook (A2)	-	-	-

# What are the relationships between these pattern candidates?

Maturity levels  
Influence Factors



Stakeholder



Concerns



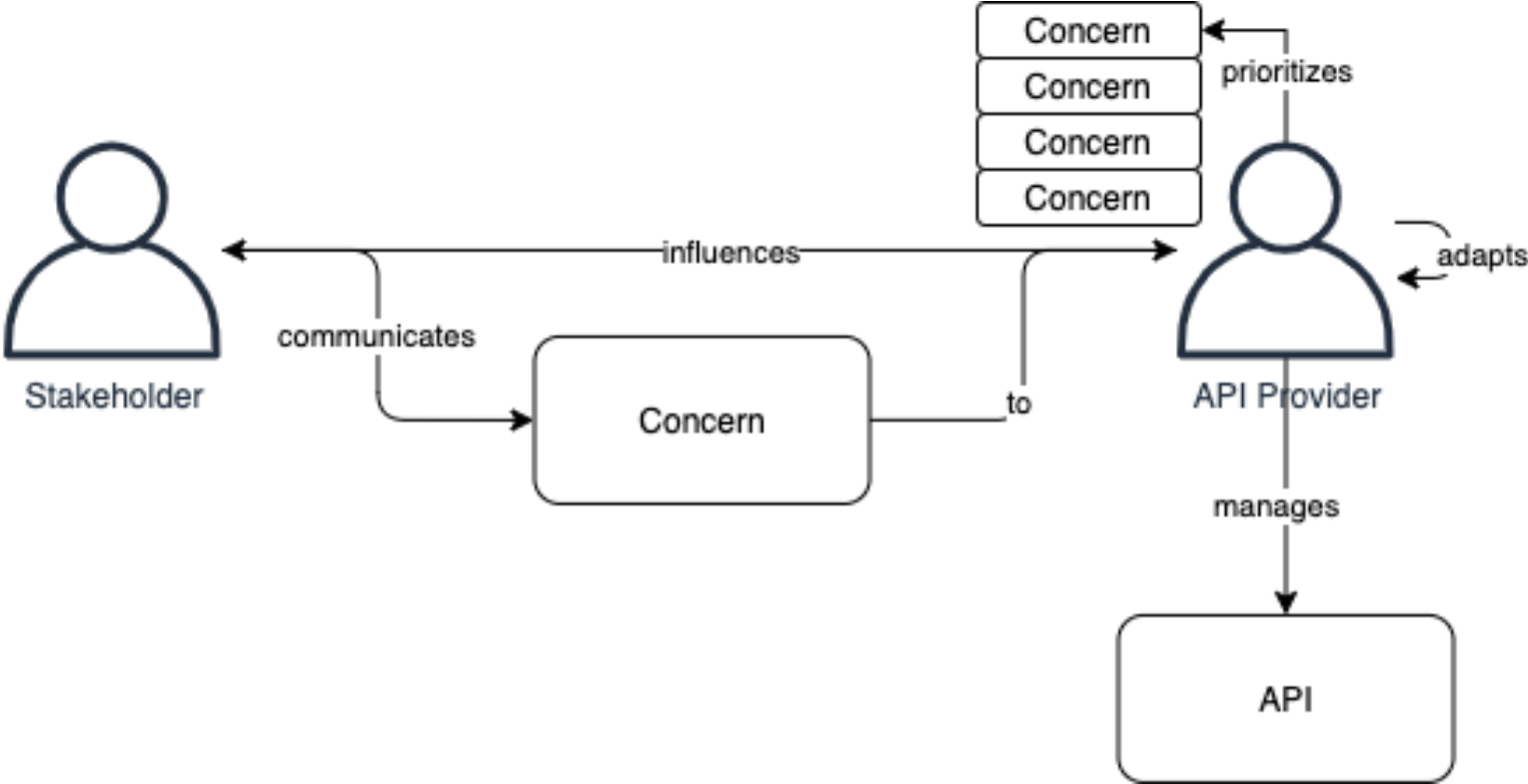
Collaboration Pattern  
Action/Adaption Pattern  
Issue Prioritization  
Pattern



Artifact

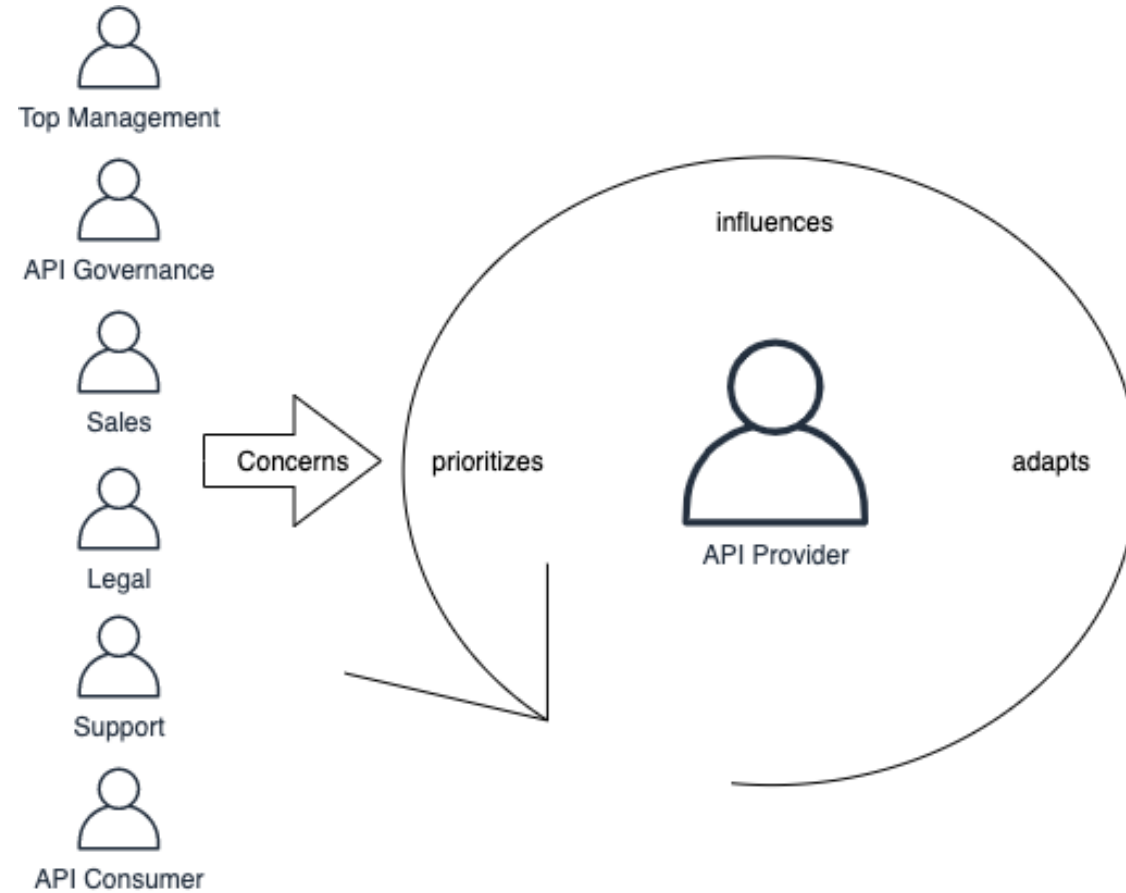


# Why do you categorize the pattern candidates?







# How does longitudinal data come into play?



# Can you show more pattern candidates?

 <b>Stakeholder</b>	 <b>Concern</b>	<b>API Management Pattern</b>
API Provider	How can I provide support for API consumers?	Pattern CP2: Dedicated First-Level Support
API Governance	How can I take advantage of new technological improvements?	Pattern CP6: API Provider Technological Lead



# Pattern CP2: Dedicated First-Level Support

**Category:** Collaboration Pattern

**Connected concerns:**

- How can I provide support for API consumers? (C1)
- How can I manage incoming feature requests? (C2)
- How can I resolve bug reports effectively? (C3)

**Description:**

API providers have to effectively and efficiently support API consumers in the usage of the API. The cloud infrastructure of the API might be able to scale with a growing number of API consumers but the API provider team can only provide so much support. If the API provider tries to support a growing number of API consumers while maintaining its original team size and other activities, the API provider is in danger of creating a bottleneck (see CP1). In the beginning, e.g. during pilot phases, being the first point of contact might be very valuable as it allows rapid prototyping and quick feedback cycles. When the API becomes more mature and gains more API consumers, this strategy might not be suitable anymore.

**Solution:**

The introduction of a dedicated first-level support into the process will change the initial point of contact for the API consumer. Simple support enquires can be handled by the customer support directly. Given the right resources like handbooks, documentations or FAQs, the customer support might be able to support common technical issues and significantly reduce the overhead for the API provider. A company-wide ticket system (see IP1) could be used to handle the communication between customer support and API provider in case more technical third-level support is required.

# Pattern CP6: API Provider Technological Lead



**Category:** Collaboration Pattern

**Connected concerns:**

- How can I take advantage of new technological improvements? (C4)

**Description:**

API Governance aims to govern the API landscape of an organization. Enforcing state-of-the-art approaches can stabilize and unify API offerings, reducing maintenance and management costs. New technologies challenge the state-of-the-art approach. API Governance needs to be open to technological improvements while enforcing best practices and standards. If API Governance acts too rigor it might prevent the adaption of new technologies that could mean technological improvements.

**Solution:**

API provider work with stakeholders in and outside the organization. Supporting and guiding API providers with company-wide standards and principles can help the API provider to get started quickly. If an API provider is presented a special set of problems that require technological changes or if a mature API provider teams does some technological reevaluations, the API governance team should take advantage of the pioneering work of the API provider. API governance should iteratively update its set of guidelines and principles based on the actual state-of-the-art within the company and industry. API provider work on real world problems and might face valuable obstacles that can create lessons learned and technological decisions that the API governance needs to adapt.

## Explorative interview questions (max. 20min)

- What are you currently working on? (This sprint / since the last meeting) (5min)
  - And who? Developer, PO, tester?
- For interesting issues: Who reported this (bug, feature request, requirement, change request... ) (3min)
- For interesting issues: Why did you prioritize those issues for now? (3min)
- Follow-up for the last interview:
  - Did you resolve the issue? (How, Why Not?) (2min)
  - Did it take longer or shorter than expected? (And why do you think that happened?) (2min)
  - [How] did you communicate the updates? (2min)
  - Any lesson learned from fixing those issues? (2min)

## Motivation for API provider

More and more organizations have started to grant third-parties access to data and functionality via APIs, giving raise to the so-called API Economy. Existing use cases show that participation in the API Economy creates value and offers strategic advantages to firms as API providers. However, little is known about the daily tasks of API developers and maintainers, their interaction with the API users or the rationales behind certain API management decisions. Therefore, our research aims at identifying common tasks and challenges of API management and corresponding solution approaches. The results of the study will benefit API managers and developers by providing solution patterns to common challenges in API management.

To achieve this goal, we are currently looking for study participants. The study will start with a kick-off interview to gain some contextual knowledge of the API under investigation. After the kick-off meeting, we will schedule a 15-30 minutes meeting every two weeks to discuss issues, solutions, and activities that emerged since the last meeting. Overall, the study will last for three months. The study data will be completely anonymized.

If you are an API manager or API developer, we would highly appreciate your participation in our study. As a participant, all results of the study will be made available to you free of cost as soon as the study is finalized.

## Literature review – API economy

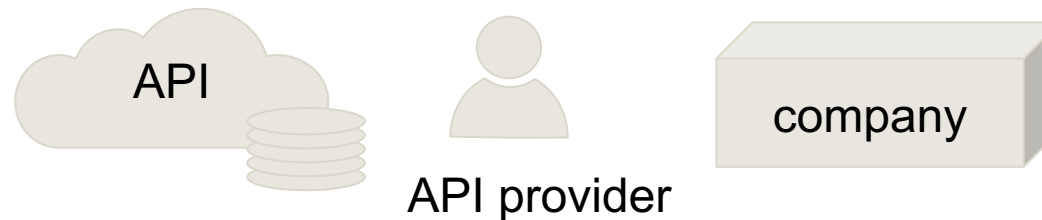
- [1] Bondel et al., 2020 research barriers preventing the advancement of the API economy.
- [2] Tan et al., 2016 describe how the service-oriented architecture emerged into the API economy.
- [3] Bonardi et al., 2016 describe a case study of an API ecosystem.

## Literature review – platform boundary resources

- [4] Ghazawneh et al., 2010 research third-party development governance through boundary resources.
- [5] Eaton et al., 2015 analyze blog posts to research distributed tuning of boundary resources.
- [6] Karhu et al., 2018 research how boundary resources sustain competitive advantage of the platform.
- **[7] Skog et al., 2018 use a longitudinal case study to research boundary resources on a digital service platform, namely Spotify.**
- [8] Bianco et al., 2014 categorize boundary resources and analyze how boundary resources should be built.

# Literature Review – API Provider

- [9] Sohan et al., 2015 research versioning of API changes
- [10] Haupt et al., 2015 research REST API designs
- [11] Hou et al., 2011 research the intent behind API change
- **[12] Lübke et al., 2019 research microservice API patterns ([MAP](#))**
- [13] Koci et al., 2019 research technical API changes





- [1] Bondel, Gloria & Nägele, Sascha & Koch, Fridolin & Matthes, Florian. (2020). Barriers for the Advancement of an API Economy in the German Automotive Industry and Potential Measures to Overcome these Barriers. 727-734. 10.5220/0009353407270734.
- [2] W. Tan, Y. Fan, A. Ghoneim, M. A. Hossain and S. Dustdar, "From the Service-Oriented Architecture to the Web API Economy," in IEEE Internet Computing, vol. 20, no. 4, pp. 64-68, July-Aug. 2016, doi: 10.1109/MIC.2016.74.
- [3] M. Bonardi, M. Brioschi, A. Fuggetta, E. S. Verga and M. Zuccalà, "Fostering Collaboration through API Economy: The E015 Digital Ecosystem," 2016 IEEE/ACM 3rd International Workshop on Software Engineering Research and Industrial Practice (SER&IP), Austin, TX, 2016, pp. 32-38.
- [4] Ghazawneh, Ahmad & Henfridsson, Ola. (2010). Governing Third-Party Development through Platform boundary Resources.. ICIS 2010 Proceedings - Thirty First International Conference on Information Systems. 48.
- [5] Eaton, Ben; Elaluf-Calderwood, Silvia; Sorensen, Carsten; and Yoo, Youngjin. 2015. "Distributed Tuning of Boundary Resources: The Case of Apple's iOS Service System," *MIS Quarterly*, (39: 1) pp.217-243.
- [6] Kimmo Karhu, Robin Gustafsson, and Kalle Lyytinen. 2018. Exploiting and Defending Open Digital Platforms with Boundary Resources: Android's Five Platform Forks. *Info. Sys. Research* 29, 2 (June 2018), 479–497.

- [7] Skog, Daniel & Wimelius, Henrik & Sandberg, Johan. (2018). Digital Service Platform Evolution: How Spotify Leveraged Boundary Resources to Become a Global Leader in Music Streaming. 10.24251/HICSS.2018.576.
- [8] V. D. Bianco, V. Myllärniemi, M. Komssi and M. Raatikainen, "The Role of Platform Boundary Resources in Software Ecosystems: A Case Study," 2014 IEEE/IFIP Conference on Software Architecture, Sydney, NSW, 2014, pp. 11-20, doi: 10.1109/WICSA.2014.41.
- [9] S. M. Sohan, Craig Anslow, and Frank Maurer. 2015. A Case Study of Web API Evolution. In Proceedings of the 2015 IEEE World Congress on Services (SERVICES '15). IEEE Computer Society, USA, 245–252.
- [10] Florian Haupt, Frank Leymann, and Karolina Vukojevic-Haupt. 2018. API governance support through the structural analysis of REST APIs. *Comput. Sci.* 33, 3–4 (August 2018), 291–303.
- [11] D. Hou and X. Yao, "Exploring the Intent behind API Evolution: A Case Study," 2011 18th Working Conference on Reverse Engineering, Limerick, 2011, pp. 131-140.
- [12] Daniel Lübke, Olaf Zimmermann, Cesare Pautasso, Uwe Zdun, and Mirko Stocker. 2019. Interface evolution patterns: balancing compatibility and extensibility across service life cycles. In Proceedings of the 24th European Conference on Pattern Languages of Programs (EuroPLop '19). Association for Computing Machinery, New York, NY, USA, Article 15, 1–24.
- [13] R. Koçi, X. Franch, P. Jovanovic and A. Abelló, "Classification of Changes in API Evolution," *2019 IEEE 23rd International Enterprise Distributed Object Computing Conference (EDOC)*, Paris, France, 2019, pp. 243-249.