Modeling Support for Strategic API Planning and Analysis

Jennifer Horkoff[1,2(✉)], Juho Lindman[1], Imed Hammouda[2,3], Eric Knauss[1,2], Jamel Debbiche[1], Martina Freiholtz[1], Patrik Liao[1], Stephen Mensah[1], and Aksel Strömberg[1]

[1] University of Gothenburg, Gothenburg, Sweden
{jennifer.horkoff,eric.knauss}@gu.se, juho.lindman@ait.gu.se
[2] Chalmers Institute of Technology, Gothenburg, Sweden
[3] Mediterranean Institute of Technology, South Mediterranean University, Tunis, Tunisia

Abstract. APIs provide value beyond technical functionality. They enable and manage access to strategic business assets and play a key role in enabling software ecosystems. Existing work has begun to consider the strategic business value of software APIs, but such work has limited analysis capabilities and has not made use of established, structured modeling techniques from software and requirements engineering. Such modeling languages have been used for strategic analysis of ecosystems and value exchange. We believe these techniques expand analysis possibilities for APIs, and we apply them as part of a cross-company case study focused on strategic API planning and analysis. Results show that goal, value, and workflow modeling provide new, API-specific benefits that include mapping the API ecosystem, facilitating incremental API planning, understanding dynamic API-specific roles, identifying bottlenecks in API change workflows, and identifying API value.

Keywords: APIs · Strategic analysis · Conceptual modeling

1 Introduction

Traditionally, software APIs (application programming interfaces) have been viewed from a technical perspective, as a means to separate implementation from functional calls – a way to define a contract of software functionality. More recently, it has become apparent that APIs are able to play a key role as part of a strategic business plan for software-intensive companies, noted by both academia [1,2], and industry [3–5].

In this work, we introduce and evaluate the use of established conceptual modeling approaches from requirements and software engineering in order to understand and analyze APIs from a strategic business perspective. Existing work, has facilitated various forms of API analysis, e.g., when to open an API [3], how to use APIs as part of a business model [4–6], and assessing API readiness

K. Wnuk and S. Brinkkemper (Eds.): ICSOB 2018, LNBIP 336, pp. 10–26, 2018.
https://doi.org/10.1007/978-3-030-04840-2_2

as part of software ecosystems [1]). However, such work has not made use of structured conceptual models to capture, understand and analyze APIs. Such models open new possibilities for API analysis. There is a rich body of literature in the use of conceptual models for strategic software analysis, particularly capturing the interplay between technologies and the organizational or business domain, e.g., [7–9] Such work allows one to map goals and dependencies in a software ecosystem [10], evaluate reciprocal value flows between actors in a value network [8], and capture organizational processes [9]. These approaches show promise for strategic analysis of APIs, but must be evaluated for this context.

Our main driving research question is: *what are the benefits and drawbacks of applying established modeling notations to strategic API analysis?* In describing our methods and experiences, we make it possible for others to replicate our modeling process, to whatever degree it is possible in a different context, performing strategic API analysis using structured models, allowing for novel types of API analysis.

More specifically, we have conducted a cross-company case study with four software-intensive companies working in the embedded systems domain. For each company, we have focused on a specific API, either established or a in planning. Via several cross-company and in-company workshops, we have worked through understanding and analyzing the strategic plans and challenges of each API using goal modeling [11], e^3 value modeling [8], and workflow modeling using UML activity diagrams [9]. We have selected these particular notations due, in part, to their appropriateness for strategic software analysis in a business context and, in part, due to the interests and requests of our particular companies, who were especially interested in ecosystem mapping, value and workflow analysis.

In this paper we present the results of our API analysis using (anonymized) examples, highlighting several benefits of conceptual modeling for strategic API analysis. Namely, modeling the ecosystem of an API in conjunction with our layered API architecture from [12] allowed the companies to find gaps in their plans, and to see the changing roles of various strategic actors depending on the API of focus. Modeling allowed us to conduct incremental planning for API deployment, and to evaluate bottlenecks in API workflows. Finally, we conducted an analysis of API value, understanding why an existing API was or was not used in a particular company. We also consider the drawbacks of the modeling approaches applied in this context, and discuss which aspects of our findings are particular to APIs, or more general for any software modeling. The primary contribution is to illustrate how widely known modeling approaches can be used in a not-yet-explored way: strategic API analysis.

Our overall goal in this continuous project is to build a framework to provide structured guidance for strategic API analysis, including the use of conceptual models. In [12] we give an early and broad overview of our framework, focusing on a 4-layer strategic API architecture. In [13], we work with some of the same results as described in this paper, but purely from the perspective of comparing the effectiveness of various modeling approaches in practice, ignoring the issues and findings related to APIs. The current submission describes further aspects

of the planned framework, including ecosystem mapping, incremental API planning, workflow and value analysis. Other components, to be elaborated in future work, include API governance, metrics, and life cycle.

This paper is organized as follows: Sect. 2 describes background and related work. In Sect. 3 we describe how our industrial modeling sessions were conducted and how our results were validated. Section 4 describes the results of our modeling efforts, organized into key findings. Section 5 discusses our results, while Sect. 6 concludes the paper and discusses future work.

2 Background and Related Work

API Analysis. APIs have been studied from an academic perspective, although the body of work in this area is not extensive, and does not make use of conceptual modeling. De Souza and Redmiles looked at how APIs help to facilitate software coordination by providing contracts and boundary objects, facilitating communication [2]. Particular attention has been paid to the use of Open APIs, particularly as a way to stimulate R&D and generate new revenue streams [6]. In this light, our study is unique in that we focus APIs in very large organizations (our partner companies range from thousands to tens of thousands of employees). As such, there is focus on internal APIs, which, due to the size of the organizations, reside in complex ecosystems.

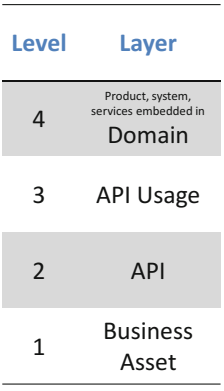

Fig. 1. Strategic API layered architecture [12]

Initial work has considered APIs from the point of view of software ecosystems, pointing out that the fast-pace changes within an ecosystem require guidance to continually assess and modify APIs [1]. In our past work we have begun to develop a framework for strategic API analysis, including a layered architecture, to understand API business value and usage, shown in Fig. 1. Here, an API protects and strategically exposes business assets. The API is used by software application(s), either internal or external, and the API usage is embedded in a domain, occupied by strategic actors and motivated by businesses cases. In previous work we have applied this general framework to several company cases, identifying elements in each layer. This analysis was helpful to map API ecosystems at a high level, but in this work we find greater insight in combination with structured modeling notations.

Several reports from industry 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 [3–5]. The existing body of API work provides useful input to our overall framework for strategic API analysis (e.g., when to change, improving usability, when/if to open). However, these approaches do not make use of established, structured, modeling frameworks to facilitate API analysis. Given strategic API concerns, existing modeling languages with a focus on strategic analysis, such as goal or e^3 value modeling, may

be able to provide additional analysis power. In this work we evaluate the utility of such tools via application to industrial cases.

Ecosystem Mapping. Over the last decade, inspired by open source and cooperative business communities, software analysis has taken an ecosystem perspective (e.g., [14–16]). Further work has focused on capturing and evaluating ecosystems with structured models. For example, Boucharas et al. provide a formal modeling language for software supply network modeling, including products, platforms, mediums, customers, and suppliers [14]. Handoyo et al. focus on capturing software ecosystems via value chains, using software supply network diagrams from [14] as a foundation [17]. Other work has used goal models to capture and understand software ecosystems [10] and tradeoffs in the degree of openness in software platform data [18].

Although work has focused on modeling software ecosystems, we are not aware of work focusing on API ecosystems in particular. As the supply network diagrams from [14,17] focus on trade relationships, not easily applicable to API analysis, we opt to use a goal modeling ecosystem approach in this work [11]. In this way, we capture a more general concept of dependencies between actors, as well as internal actor motivations for participating in the ecosystem, including problems and challenges.

Incremental Modeling. The practice of capturing as-is vs. to-be is wide-spread in conceptual modeling. Although API analysis has considered various stages of API design or release (e.g., private to public [3]), we have not seen examples of incremental planning using conceptual models specifically for APIs.

Workflow Analysis. Modeling and analysis of workflows (business processes, activities) is widespread in both software and business (e.g. [9]). To our knowledge, we have not seen specific consideration of API-related workflows, e.g., the process of updating or changing an API. In this work we apply UML activity diagrams for this purpose.

Value Analysis. The emphasis on value as part of agile methods, as well as the focus on value in, for example, value-based software engineering [19], has provoked a recent academic focus on value analysis and modeling. Several value-oriented modeling approaches have been introduced, including [17]. In this work we use e^3 value modeling as per Gordijn et al. [8]. We select this language due to its simple visual syntax, continued application, development in research (e.g., [20]), and availability of tool support.

3 Methodology

In this section we describe the research context, including a brief description of our anonymized companies and their APIs of focus, and a description of our modeling methodology.

3.1 Research Context and Case Companies

This research is carried out as part of the Chalmers Software Center (SWC)[1]. Work in the center is organized into half-year sprints, renewable as part of continuous projects. Projects involved interested software center companies, including many of the leading software companies in Northern Europe.

The high-level goal of the research sprint (January to May 2017) was continue to develop aspects of the strategic API framework while providing analytic value in API management and strategies to our four partner companies. All project companies (C1–C4) are SWC partner companies working in the embedded systems domain. Each company selected a particular API for more in-depth work as part of the project. We describe the APIs of focus for each company in Table 1.

Table 1. Case company API description

Company	Description
Company 1 (C1)	C1 offers many APIs to its physical devices as well as a through cloud services. The API of focus, a cloud API, was in the planning stages during this study
Company 2 (C2)	C2 supplies databases to its customers that are used in the generation of reports that support tasks such as quality control. The API of focus, a reporting API, was in the planning stages
Company 3 (C3)	The investigated API is mainly internal and related to the reuse of common function signatures across products. The API of focus, a profile API was in partial operation
Company 4 (C4)	The studied API was internal and encompassed global software design rules handling faults and alarms. The API of focus was in use

3.2 Model Creation and Validation

Modeling and analysis was conducted as part of three SWC thesis projects [21–23]. The thesis groups (G1–G3) each worked with 1–2 companies, continually sharing results with the research team in weekly meetings. Based on the interests of each company, different modeling methods were applied. As all companies were interested in mapping their API ecosystem, inspired by work using goal modeling for ecosystem mapping [10], goal modeling was used in all cases. C3 was particularly interested in API workflow analysis, thus activity diagrams were applied to this case. C4 was interested in understanding the value of their API to potential users, thus we applied e^3 value modeling in this case.

The project started with a cross-company coordination workshop. Each of the three groups conducted a series of group and individual workshops and interviews in order to collect qualitative data to facilitate modeling. G1 and G2 conducted

[1] https://www.software-center.se/.

workshops on location with the company, as well as follow-up online interviews. G3 was situated within C4 for a period of roughly three months.

Information gathered from workshops and interviews, including a selection of technical documentation, was used to create models. It was agreed with the companies that the first round of modeling should focus on a particular scenario or user story, to keep the scope of the models in check. Each group attempted to classify their resulting models in terms of the layered API architecture reported in [12]. The modeling process was iterative, with the students receiving iterative feedback from someone knowledgeable in the modeling approaches (the first author), researchers knowledgeable about the cases, and the company contacts. The general elicitation process can be summarized as follows:

1. **Introductory Group Interview:** Necessary to understand the API ecosystem of the companies.
2. **Off-site Modeling:** Using available context knowledge and the API usage scenario of focus (user story) to create initial model versions.
3. **Interactive Workshop:** Starting with initial models, expand and correct the models interactively using group input.
4. **Follow-up Online Interview(s):** Finalize data collection and fill gaps discovered while modeling. Discuss experiences with modeling approaches.
5. **Dissemination Workshop:** Summarize modeling and analysis results in a workshop with all company representatives present.

Company workshops had 3–6 company participants including roles such as developer engineers, development managers, software engineers, product managers, and expert engineers, i.e., those involved with and familiar with the development or operation of the API. Participants had a technical background, and were generally familiar with software modeling, although not specifically with goal or value modeling. Online interviews were conducted with individual representatives from each company, who had been present in the workshops. Workshops lasted three hours, while individual interviews were typically one hour.

G3 also conducted an introductory group interview with C4, but gathered further data with individual semi-structured interviews and a survey, selecting participants involved in the API Framework (FW). Ten interviews and two surveys were conducted with the same questions, with interviews lasting 45 min. G3 also had access to archival data concerning their API of focus.

For G1 and G2 model creation was iterative and continuous, with workshops and interviews presenting and receiving feedback on the models. G3 explicitly used member checking to improve the accuracy, credibility and validity of the collected data [24]. Four new interviews were scheduled with previous participants, each lasting around an hour, during which G3 described and went through the models step by step, receiving feedback. Further member checking was conducted when the authors elicited feedback and general impressions on modeling results from each company in the final shared workshop. More information on the modeling method and company participants can be found in the full theses [21–23].

4 Results

We describe our results, grouping them into categories.

API Ecosystem Mapping. For each case, the teams used goal modeling to map the ecosystem of the API, including the API itself, the company, and the various internal and external actors in the ecosystem. For each actor, the actor's motivations and dependencies on other actors were considered. A high-level view of the resulting ecosystem map for C1 can be seen at the top of Fig. 3 and for C2 (with layers) in Fig. 5. A more detailed view of part of a resulting model for C3, analyzing the situation before API workflow redesign, can be seen in Fig. 2 (see [11] and red annotations for language constructs). This figure focuses on the process of approving changes to the profile, the API equivalent construct in this case, enabling a common interface for specifying device functionality. The model uses qualitative goal model analysis (as described in [25], legend on top right of figure) to determine the satisfaction level of quality goals based on the contributions of tasks/goals.

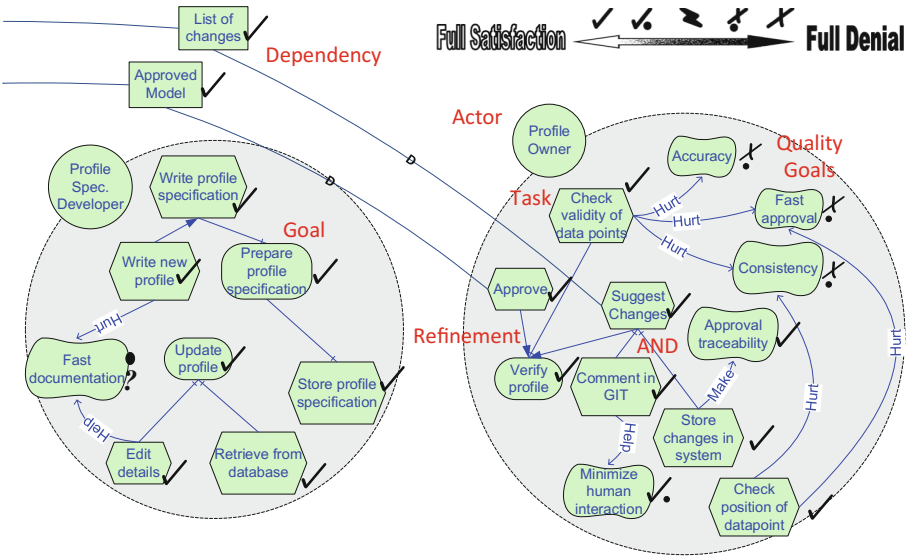

Fig. 2. Selected details of the goal model for C3, capturing the as-is situation before workflow redesign (Color figure online)

Although the models were complex, much of the modeling was done as part of company workshops, in a participatory manner with direct input from participants. Thus, company partners were generally engaged, and found this type of modeling useful to have a high-level view of the ecosystem. For example, C1 stated that it helped them understand the needs and wants of the various types of customers in their API ecosystem (top of Fig. 3).

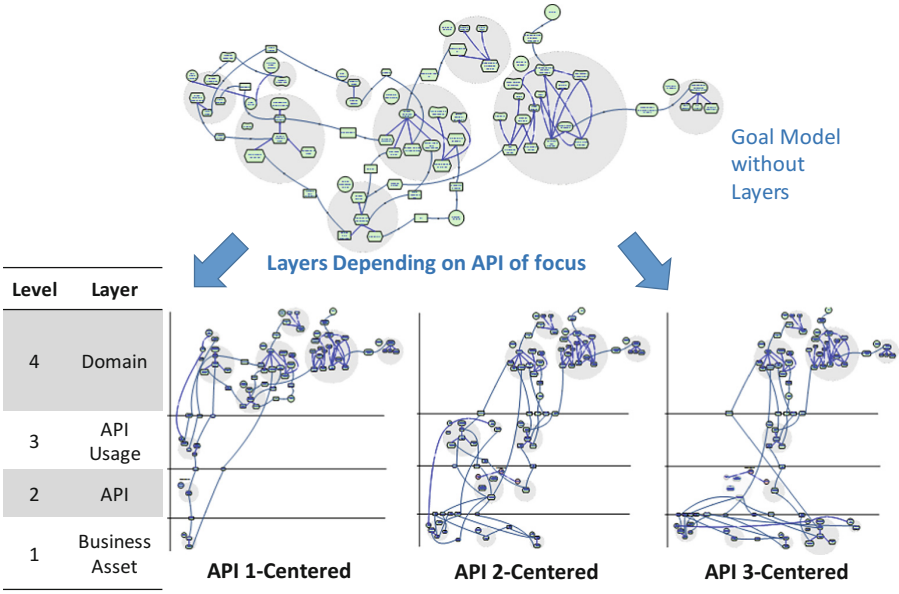

Fig. 3. Layering of goal model using API layers from [12] showing different API-centric views for C1 (high-level view, details obscured)

The process of iterative ecosystem modeling helped to show gaps in the modeler's knowledge. For example, for C2, several questions arose: what is the role particular module in API design? The relevance of the cloud? Who accesses the API? When? etc. Most of these questions could be answered easily by the customer partners, but as this case was an API in development, forcing the C2 participants to hammer out details was often seen as helpful.

Summary: Ecosystem modeling with goal models was helpful in mapping out and understanding the domain, including participating actors, and in solidifying details. Drawbacks included model complexity.

Ecosystem Mapping with API Layers. The teams made an effort to take each of the models created as part of the API analysis and sort the model elements as per the layers of the strategic API architecture described in Sect. 2. We found that making this division for workflow or e^3 value models was difficult (more detail described in [13]). However, this was possible for goal models by assigning API-related actors to layers. See Fig. 3 for a high-level view of how this was performed for C1. Note that the details of this figure are deliberately obscured to hide details of the company analysis. Figure 4 (described in next section) gives a simplified view showing only the actors.

The participant companies found that mapping API ecosystem actors to API layers was a helpful exercise in understanding the roles of the ecosystem actors, as they related to the API. In some cases, this mapping revealed significant gaps

in our ecosystem goal models. For example, in the initial C1 model, we had neglected to include the assets protected and managed by the device API.

Summary: Mapping ecosystem models to API layers helped to identify the API-centered roles of the actors, and in some cases to reveal missing actors. The drawbacks were that this was not easily possible with e^3 or workflow models.

Changing API Ecosystem Perspectives. When mapping our API ecosystem models to the API layers, it was challenging to map actors to layers, as their role was contextual. This was particularly true when an ecosystem contained more than one API. For example, in C1 the C1 ecosystem contains three APIs: the low-level device API which allows one to access content on the device(s), the raw content cloud API which provides device content to third party cloud developers, and the processed content cloud API which takes processed content from third party developers and provides it to the end customers.

In this case, ecosystem actors such as third party developers (gray circle in Fig. 4) could be placed on many layers depending on the API of focus. From the perspective of the Device API, this actor is part of the domain (left figure), a user of the raw content cloud API which uses the Device API. From the raw content API perspective, it is part of the App SW layer (middle), as the software which uses the raw content cloud API. While from the perspective of the processed content API, it is part of the Business Asset layer (right), as an asset used by the processed content cloud API. We show the different allocation of domain actors to layers depending on the API of focus at a high-level in Fig. 3 and with more abstraction in Fig. 4. In Fig. 4, each actor is a different color, so that the position of the actors can be traced across figures.

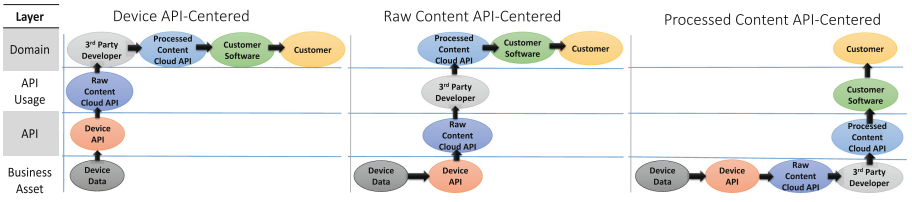

Fig. 4. API layers from [12] showing different API-centric views from the goal model for C1 (simplified view of Fig. 3 where each circle is represents a different actor) (Color figure online)

This type of analysis allows companies to understand the dynamic roles of the actors in their API ecosystem, particularly if the ecosystem is complex and contains different APIs. Similar situations can be found in some of the other companies, such as C3, with a communication API at a low level and a profile API (referenced in Fig. 2) at a higher level.

Summary: Mapping ecosystem models to strategic API layers allowed us to understand the changing roles of ecosystem actors, depending on the API of focus. Drawbacks were that this would only apply in cases with multiple APIs.

Incremental API Planning. In the case of C2, the analysis focused on planning a new API. C2, a large, international company, found planning global API deployment to be a particular challenge. Should the API be deployed globally in all locations? In some locations? With partial functionality? Should the old way of accessing data be preserved in parallel?

In this case an incremental planning approach using the layered ecosystem mapping models proved useful for the company. We took our initial ecosystem model, and with company input, modified it to show the incremental development and deployment of the API. A high-level view of the output is shown in Fig. 5. In the current (as-is) situation, there is no API (the API layer is empty), and several problems exist (e.g., customers often request custom reports, labor intensive for C2 employees). In the near future, the API is available as an option, some employees may use it, alleviating some problems, but others may choose to access data in the old way. In the near future, only the API is available to access data, thus there is a transition period to ease employees into API use. In the future, customers may use the reporting API directly. We show the details and changes for one actor in Fig. 6.

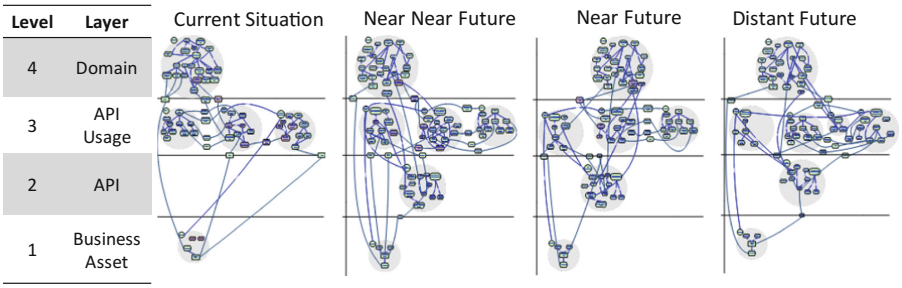

Fig. 5. Incremental planning of case API for C2 via goal models (high-level view)

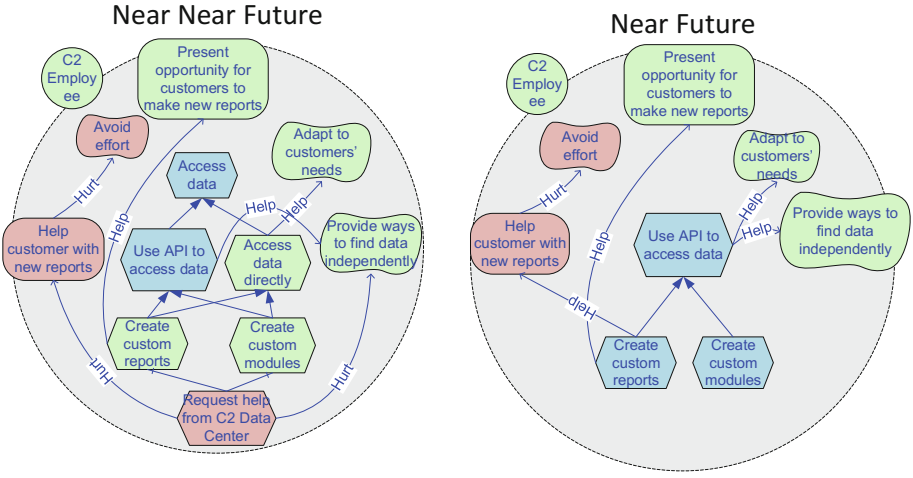

Fig. 6. View of one actor from Fig. 5, C2 case, near near vs. near future

In this case, colors (teal, orange) were used to show problematic goals, and how these problems were mitigated with subsequent API deployment steps. Although C2 found the process of creating these figures helpful in understanding the benefits of this form of incremental deployment from an ecosystem actor-centric view, such models were not easily able to cover technical details. For example, which data to expose via the API first? It is possible the model could help to evaluate this question, but detail would need to be added, which is difficult given the complexity of the model as is.

Summary: Goal ecosystem models allowed us to show incremental changes in plans over time, including goals that were satisfied/problems that were solved. Drawbacks include difficulty in representing technical detail.

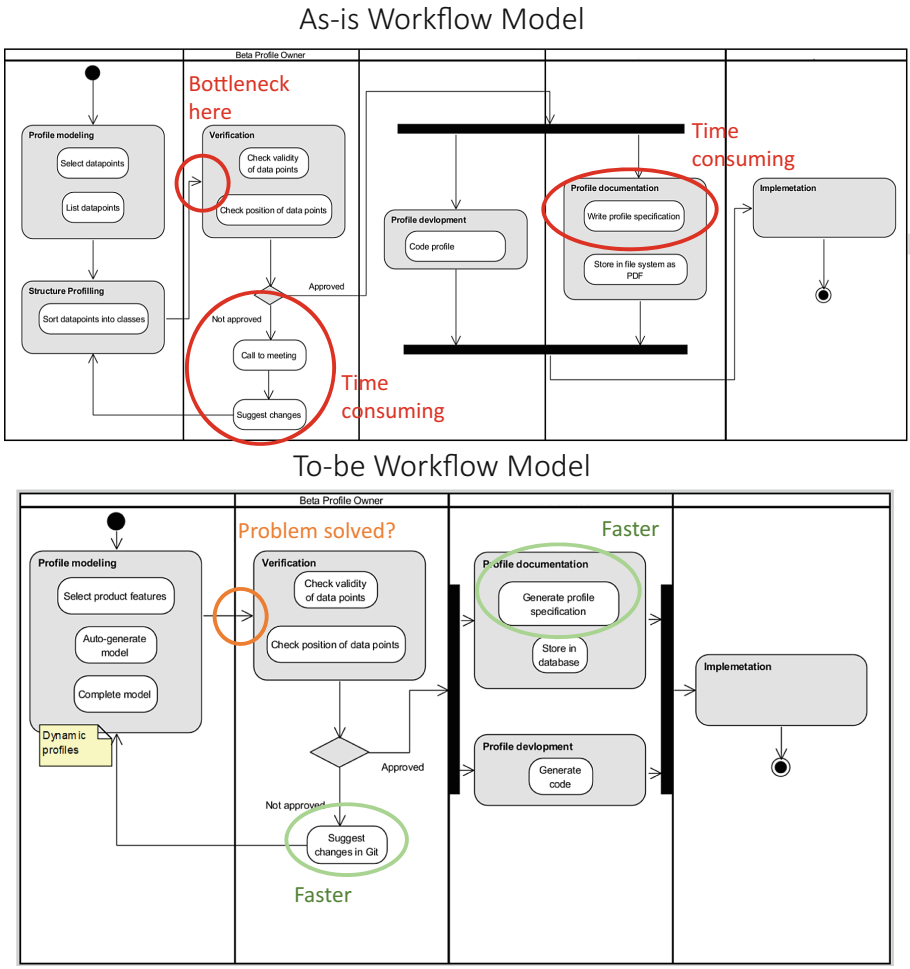

Fig. 7. As-is and to-be API modification process for C3 (some details removed)

API Workflow Analysis. C3 was particularly interested in understanding the workflow of processing requested changes to their API. In this case, they had an existing workflow and a new, planned workflow for their profile-based API. In addition to ecosystem modeling via goal models (Fig. 2), we performed workflow modeling for the as-is and to-be (planned) cases. Although we were not able to perform formal or quantitative workflow analysis (e.g., simulation, see [13] for more details), we were able to use the models to indicate bottlenecks and problematic flows for the as-is scenario, and to show how these problems would be at least partially alleviated in the new plan. See Fig. 7 for details. For example, in the as-is flow, the steps when the profile (API) owner rejected new additions or changes was time consuming. In the planned workflow, this part of the process was sped up by use of git. In other cases it was not clear if or how bottlenecks were solved by the new process design, and these cases were discussed with C3. Mapping back to qualities in the goal model, we found that the new design achieves maintainability, extensibility, and (partial) flexibility, but did not directly improve accuracy, consistency or fast approval.

Summary: Applying workflow modeling to APIs helped to understand the process of changing an API, including bottlenecks and planned process improvements. Drawbacks include an inability to perform workflow simulation.

API Value Analysis. In the case of C4, the company wanted to know why internal partners were motivated to use or not use an internal API enforcing software design rules. Use of the API was considered a good practice, but was not mandatory. In this case, it was agreed with the company that it would

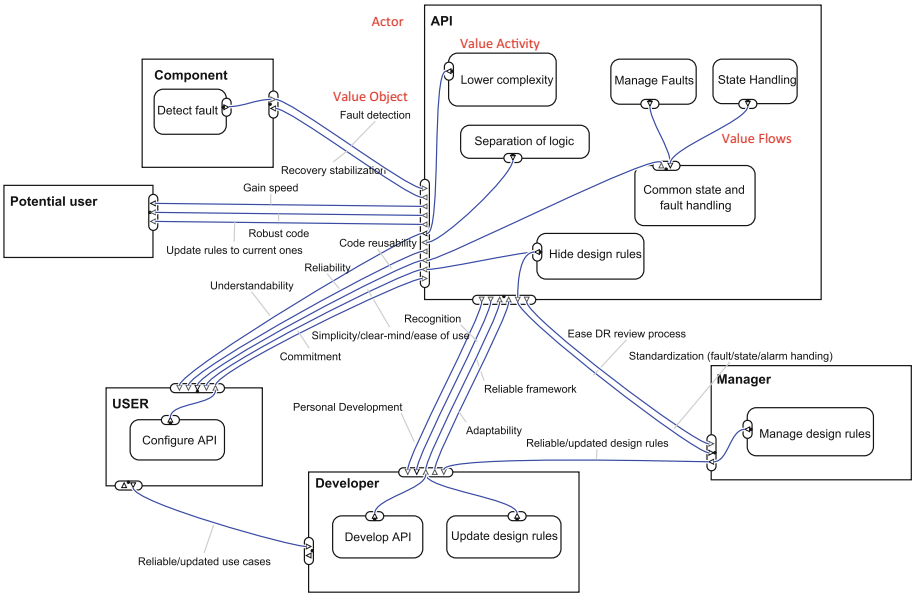

Fig. 8. e^3 value model for case API for C4 (some details changed) (Color figure online)

be useful to evaluate the value of the API, and as such, e^3 value modeling was applied. The result (some details changed), is shown in Fig. 8 (language annotations in red, more detail in [8]). The process of value modeling helped to highlight advantages/values of the API, showing why various actors would be motivated to use the API (e.g., robust code, standardization).

We note that the model did not necessarily capture the disadvantages of the API, including the learning curve, and difficulties in testing. We are currently continuing the project from both a goal and value-oriented perspective, at the request of the companies.

Summary: Value analysis was useful for understanding why people were motivated to use an API, in a relatively simple view. Drawbacks included an inability to show problems or missing values.

5 Discussion

Modeling Summary. We answer our initial research question, *what are the benefits and drawbacks of applying established modeling notations to strategic API analysis?* by summarizing the benefits (B) and drawbacks (D) of applying selected established modeling techniques to analysis of APIs from a strategic, business perspective (Table 2).

Table 2. Summary of benefits and drawbacks for API modeling

Benefits	Drawbacks
B1: Iterative modeling helped to reveal gaps in knowledge, both for the researchers and company partners	D1: Modeling, particularly goal modeling, can be complex and take some time to understand
B2: Mapping of ecosystems of goal modeling facilitates an understanding of API actors and motivations	D2: Goal models could not easily capture some of the specific technical detail needed for strategic analysis
B3: Division of ecosystem maps into strategic API layers helped to find gaps in knowledge and better understand the roles of various API actors	D3: Workflow models needed additional annotation to indicate the presence of problems or bottlenecks
B4: Dividing ecosystem models into layers helped to show the dynamic API-specific roles of various actors when multiple actors are involved	D4: e^3 value models could not easily capture drawbacks or reasons not to use a particular API
B5: Ecosystem models can be used to show incremental API planning, including gradual access to and use of the API	
B6: Issues in API workflows, such as bottlenecks, can be understood and addressed via workflow modeling	
B7: e^3 value modeling can help to understand the value (or lack of value) of APIs for actors in the ecosystem	

We have shown that by using structured models, we gain analysis benefits beyond existing API-related work (Sect. 2), including a visual mapping of API ecosystems, visualized incremental API planning, an analysis of the dynamic roles of API actors, API workflow and value analysis.

API Maturity. Our cases covered APIs of differing maturity. We can observe that ecosystem mapping was particularly useful for APIs in the planning stage (C1, C2), although it still provided some value to the other companies. Workflow modeling was applicable when workflows were established or planned (C3); such modeling would have been less applicable to those cases in the early planning stages (C1, C2). Value modeling was useful to capture value exchanges for an established case (C4), but could also be useful in planning stages.

Open/Closed APIs. Most of the APIs in our cases were internal (closed). However, because the companies were large, internal users could in some ways be treated as external: they were often geographically distributed and not personally known to our business contacts. As such, they were similar to open APIs.

API-Specific Results. One could question how many of our benefits and drawbacks are API-specific, or could occur in any strategic modeling exercise for a software-intensive company. B1 and B2 could be considered quite general, findings which are likely to appear in a software-related modeling exercise, while B3-B5 are more specific to APIs. B6 and B7 could be found for the analysis software in general; however the issues of bottlenecks in B6 is exaggerated by the presence of an API guardian, which is less common for general software. Value analysis in B7 is particularly useful in the case of APIs, as API value is often less obvious than with regular software. In this case, the value is not in a direct provision of functionality, but more subtly, in the enforcement of good coding principles. All of our found drawbacks are general, not specifically related to APIs. We consider this positive: use of the modeling techniques for APIs does not introduce additional significant challenges compared to general use of the modeling notations.

Threats to Validity. In terms of *Construct Validity*, the student modelers and the company representatives may have misunderstood the syntax or semantics of goal, workflow, or value models. We mitigated this threat by giving an overview of the modeling language at multiple points in the study. In this study, we worked with companies who were particularly interested in API analysis, including ecosystem, value and workflow analysis. In other cases, with less company buy-in, the modeling activities may be less fruitful.

Considering *Internal Validity*, all groups used some form of triangulation, collecting data from workshops, documents, archival data, and interviews. For C1, C2 and C3, validation rounds after the interactive workshop only involved one person per company. However, we mitigate this effect by collecting impressions from more people and multiple companies in the final cross-company workshops.

Examining *External Validity*, all case companies are located in Scandinavia; however, the companies are international. Applying the same modeling process to different companies with different APIs may produce differing observations.

However, we mitigate this possibility by involving a number of people from four different companies. Furthermore, all companies are in an embedded system domain, involving hardware in their products. We believe our results would be transferable to pure software companies. Our companies are also large; API modeling may be less beneficial for smaller companies. It is also likely that our results would not hold if different modeling notations were used.

Finally, considering *Reliability*, our study had the participation of a goal model expert, a co-author of [11]. However, most models in the studies were created by the student modelers.

6 Conclusions

We have used existing established modeling techniques in a novel way, demonstrating novel types of API analysis from a strategic business perspective. We have found several benefits of this application, including API-specific benefits, often related to the combination of structured modeling with our API layered architecture introduced in previous work. Drawbacks are common to most modeling efforts, including model complexity and expressive limitations.

Our findings are being incorporated into our framework for strategic API analysis. We are currently working with our industry partners to develop modeling methods, API metrics, and guidance in terms of API governance.

Acknowledgments. Thanks to company contacts and the Chalmers Software Center for support.

References

1. Hammouda, I., Knauss, E., Costantini, L.: Continuous API-design for software ecosystems. In: Proceedings of 2nd International WS on Rapid and Continuous Software Engineering (RCoSE 2015 @ ICSE), Florence, Italy (2015)
2. de Souza, C.R.B., Redmiles, D.F.: On the roles of APIs in the coordination of collaborative software development. CSCW **18**(5), 445 (2009)
3. Nordic API: Developing the API mindset: a guide to using private, partner, & public APIs (2015). https://nordicapis.com
4. IBM Institute for Business Value: Evolution of the API economy. Adopting new business models to drive future innovation (2016)
5. Oracle Communications: Making money through API exposure. Enabling new business models (2014). http://www.oracle.com/us/industries/communications/comm-making-money-wp-1696335.pdf
6. Aitamurto, T., Lewis, S.C.: Open innovation in digital journalism: examining the impact of open APIs at four news organizations. New Media Soc. **15**(2), 314–331 (2013)
7. Eric, S.: Social Modeling for Requirements Engineering. MIT Press, Cambridge (2011)

8. Gordijn, J., Akkermans, H., Van Vliet, J.: Designing and evaluating e-business models. IEEE Intell. Syst. **16**(4), 11–17 (2001)
9. Dumas, M., ter Hofstede, A.H.M.: UML activity diagrams as a workflow specification language. In: Gogolla, M., Kobryn, C. (eds.) UML 2001. LNCS, vol. 2185, pp. 76–90. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45441-1_7
10. Yu, E., Deng, S.: Understanding software ecosystems: a strategic modeling approach. In: Proceedings of the Third International Workshop on Software Ecosystems, IWSECO-2011 Software Ecosystems 2011, Brussels, Belgium, pp. 65–76 (2011)
11. Dalpiaz, F., Franch, X., Horkoff, J.: iStar 2.0 language guide. arXiv preprint arXiv:1605.07767 (2016)
12. Lindman, J., Hammouda, I., Horkoff, J., Knauss, E.: Emerging perspectives to API strategy. IEEE Software (Under revision). https://tinyurl.com/yaofetrx, https://www.computer.org/csdl/mags/so/preprint/08501965-abs.html
13. Horkoff, J., et al.: Goals, workflow, and value: case study experiences with three modeling frameworks. In: Poels, G., Gailly, F., Serral Asensio, E., Snoeck, M. (eds.) PoEM 2017. LNBIP, vol. 305, pp. 96–111. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70241-4_7
14. Boucharas, V., Jansen, S., Brinkkemper, S.: Formalizing software ecosystem modeling. In: Proceedings of the 1st International Workshop on Open Component Ecosystems, pp. 41–50. ACM (2009)
15. Bosch, J.: From software product lines to software ecosystems. In: Proceedings of the 13th International Software Product Line Conference, pp. 111–119. Carnegie Mellon University (2009)
16. Jansen, S., Finkelstein, A., Brinkkemper, S.: A sense of community: a research agenda for software ecosystems. In: 2009 31st International Conference on Software Engineering-Companion Volume, ICSE-Companion 2009, pp. 187–190. IEEE (2009)
17. Handoyo, E., Jansen, S., Brinkkemper, S.: Software ecosystem modeling: the value chains. In: Proceedings of the Fifth International Conference on Management of Emergent Digital Ecosystems, pp. 17–24. ACM (2013)
18. Sadi, M.H., Yu, E.: Modeling and analyzing openness trade-offs in software platforms: a goal-oriented approach. In: Grünbacher, P., Perini, A. (eds.) REFSQ 2017. LNCS, vol. 10153, pp. 33–49. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-54045-0_3
19. Biffl, S., Aurum, A., Boehm, B., Erdogmus, H., Grünbacher, P.: Value-Based Software Engineering. Springer, Heidelberg (2006). https://doi.org/10.1007/3-540-29263-2
20. Gordijn, J., Petit, M., Wieringa, R.: Understanding business strategies of networked value constellations using goal-and value modeling. In: 14th IEEE International Conference on Requirements Engineering, pp. 129–138. IEEE (2006)
21. Debbiche, J., Strömberg, A., Liao, P.: Applying goal modeling to API ecosystems: a cross-company case study. Bachelor thesis (2017). http://hdl.handle.net/2077/52649
22. Bedru, F., Freiholtz, M., Mensah, S.: An empirical investigation of the use of goal and process modelling to analyze API ecosystem design and usage workflow. Bachelor thesis (2017). http://hdl.handle.net/2077/52648

23. Hussein, M., Lundén, A.: An industrial assessment of software framework design: a case study of a rule-based framework. Master's thesis (2017). https://tinyurl.com/y9td34v6
24. Harper, M., Cole, P.: Member checking: can benefits be gained similar to group therapy? Qual. Rep. **17**(2), 510–517 (2012)
25. Horkoff, J., Yu, E.: Interactive goal model analysis for early requirements engineering. Requir. Eng. **21**(1), 29–61 (2016)