CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

An Empirical Investigation of the Use of Goal and Process Modelling to Analyze API Ecosystem Design and Usage Workflow

Bachelor of Science Thesis in Software Engineering and Management

FEKI MUNIR BEDRU
MARTINA FREIHOLTZ
STEPHEN MENSAH

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

An Empirical Investigation of the Use of Goal and Process Modelling to Analyze API Ecosystem Design and Workflow

FEKI M. BEDRU
MARTINA FREIHOLTZ
STEPHEN MENSAH

Supervisor: JENNIFER HORKOFF
Examiner: SALOME MARO

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

An Empirical Investigation of the Use of Goal and Process Modelling to Analyze API Ecosystem Design and Usage Workflow

Feki Munir Bedru
Department of Computer Science and Engineering
University of Gothenburg
Email: gusbedfe@student.gu.se

Martina Freiholtz
Department of Computer Science and Engineering
University of Gothenburg
Email: m@rtina.be

Stephen Mensah
Department of Computer Science and Engineering
University of Gothenburg
Email: gusmenst@student.gu.se

***Abstract*—The case company in this study develops different mechatronic devices for its customers around the world. Customers who use these products, do so with an application software to access the business assets of the company through an API (Application Programming Interface). The company wants to improve on its API management strategy which is expected to solve some of the bottlenecks they have in their current workflow.**

Our focus is to investigate how goal- and process models can be used to effectively capture dynamic and static aspects of an API ecosystem. We also investigate how the models can be improved for the purpose of analyzing an API ecosystem design. Four models were created, modelling the current and future workflow and API ecosystem design of the case company (one goal model and one process model for each scenario). To help answering the research questions, the models were analyzed using systematic forward goal model analysis. An evaluation of the models in terms of effectiveness and expressiveness was also addressed.

Keywords*—*software ecosystem, goal model, process model, API design, workflow

I. Introduction

The thesis is part of a larger research project in the domain of ecosystem-driven development, conducted at University of Gothenburg and Chalmers Software Center by I. Hammouda et al [1].The aim of the research project is to provide companies with an analytical framework for designing and managing API ecosystems effectively and efficiently [2].

We conducted a case study with a company in the embedded systems industry. The case company is currently moving towards the implementation of a new suggested API ecosystem workflow, which is expected to solve some of the bottlenecks they have in their current business process. The company wants to be able to analyze the effects these changes have on their API ecosystem design.

To that end, this thesis investigates how goal modelling and process modelling can be used together to capture both the static and dynamic aspects of an API ecosystem. In addition, to find out how these models can be improved for the purpose of analyzing an API ecosystem design. We aim to answer the following questions:

RQ1 Can goal modelling and process modelling be used together to effectively capture the static and dynamic aspects of API ecosystems?

RQ1.1 *How well can goal modelling be used to capture the static aspects of API ecosystems?*

RQ1.2 *How well can process modelling be used to capture the dynamic aspects of API ecosystems?*

RQ2 How can goal models and process models be improved for the purpose of API ecosystem design and workflow analysis?

RQ2.1 *How can goal models be improved for the purpose of API ecosystem design analysis?*

RQ2.2 *How can process models be improved for the purpose of API workflow analysis?*

To answer these questions, we modeled an API ecosystem using goal- and process modelling. The choice for goal and process model is the fact that, goal models have the capability of capturing and helping the viewer understand early requirements of a system [3]. The goal model expresses the relationship between actors and dependencies between goals and tasks in a static state, while process models are used to capture the activities of the workflow in the dynamic state [3].

To answer the first research question, we tested how well the models worked for the designated purpose by evaluating the models based on criteria of expressiveness and effectiveness. By expressiveness, we refer to the models' ability to capture the aspects of the software ecosystem which we want to capture. Effectiveness here refers to the ease of which these aspects can be captured [4].

To answer the second research question, we combined the results of the expressiveness and effectiveness analysis (uncovering uncaptured aspects and modelling challenges) with observations from the model iterations and the feedback received from the case company.

Creating visualizations of the API system and the workflows through a combination of goal modelling and process modelling,could lay the groundwork for a better understanding of the current (as-is) and proposed (to-be) API ecosystem

design. By making it easier to discover possible flaws (failures to meet requirements, bottlenecks, etc.), such models could potentially assist in improving the design of API ecosystems.

Goal models and process models have been used earlier to visualize software ecosystems, but not for the visualization of API ecosystems specifically. An API enables actors in the domain to access business assets [2], which means that it has to be both reliable and secure. Our contribution to the existing body of literature will be an example of how goal modelling and process modelling can be used together to capture the static and dynamic aspects of API ecosystems, with the ambition of ecosystem analysis and improvement.

Throughout the report we refer to a number of technical terms, which are discussed in more detail below.

1) Workflow: The workflow refers to actions, a sequence of activities, or tasks set in a certain order enabling a system to serve its purpose [5]. The workflow focuses on the use of entities (e. g. actors and activities) to explain the sequence of operation or work procedure.

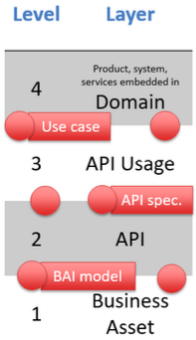

Fig. 1. Example of API Framework.

2) API: An Application Programming Interface (API), is defined as a multi-layered digital innovation object (Fig. 1) that allows developers to access business assets [2]. This differs from the typical technical use of the term. Four layers are defined: Business assets, API, API Software/User, and the Domain.

Boundary objects are instruments that are used to manage or govern communication between adjacent layers and determine the way that communication is established. For example, a use case indicates how actors in the domain (users) can communicate or access the API layer to fetch information from the business asset (BA) layer based on an API specification. The API specification in turn determines or provides business asset information to the user based on an agreement between stakeholders and BA provider. An API model, a boundary object between the API and business asset layer, manages the amount of assets that can accessible by the API layer [2].

To limit the scope of this study, we are not consciously attempting to model boundary objects shown in Fig. 1. The framework is presented and discussed in detail in the paper by Hammouda et al [2].

3) Software Ecosystem: A software ecosystem can be defined as a software system characterized by continuous evolution and independence with other systems [6]. Software ecosystems can support the development of large systems using components developed by actors [6]. This phenomenon includes software development activities outside traditional boundaries [7]. A large software ecosystem provides diversity and this diversity is also used to implement different techniques such as service-oriented architecture enabling a user add more features or configurable systems [6].

4) Static and Dynamic Aspects: In our analysis of the goal- and process models, we talk about static and dynamic aspects of the API design and workflow. To clarify what we mean with these terms, we define them in the context of this report.

Goal models are static, that is, they present "what things exist, their attributes and interrelationships [8]". We think of them as system snapshots, which in our case show how the API ecosystem is structured. By comparing two snapshots in the context of this study (the as-is goal model and to-be goal model) we analyze the changes between the two scenarios. Since the model is static, it does not give the viewer any direct information about the underlying processes. In order to get a more complete picture of the system, a dynamic model is needed.

Dynamic models show the states, state transitions, and processes of a system [8]. We use process models in the form of UML activity diagrams to visualize the workflow of the case company. Just as with the goal models, we compare the differences between the as-is and to-be process models by analyzing the effect of change on the workflow, goals, system requirements, and bottlenecks.

II. Background

A. API Design

In an empirical study conducted at Software Center as a part of this project, the researchers aim to provide companies with an analytic framework for designing and managing API systems. The paper [2] (under submission) presents APIs as one layer within a context of other layers relevant to API management and design: domain, app software/API user, and business assets. Each layer has a separate set of concerns. The advantages with loosely coupled separate layers are the ability to innovate at layer level, and the decoupling of design decisions made for a specific layer. In this paper, we use the layered framework to think about the different actors' role in the API - especially in the context of the goal models, where the actors are visually sorted into the defined layers.

The authors also identify three kinds of artifacts in the layer boundaries: Use cases (between the domain and API user), API specification (between API user and API), and API model (between the API and the business assets). In the scope of this study, these boundary objects were not identified.

B. Goal Modelling

The i* framework was selected for the goal modelling effort in this study. Yu [9] provides an outline for the i* framework for modelling of requirements in the early phases of requirements engineering. An overview of the basic i* goal model syntax can be found in Fig. 2.

The framework is presented as an effort to make requirements "precise, complete and consistent [9]", and increase the understanding of a domain to decrease the risk of failure. It also serves the purpose of helping the stakeholders get a better understanding of various design possibilities.

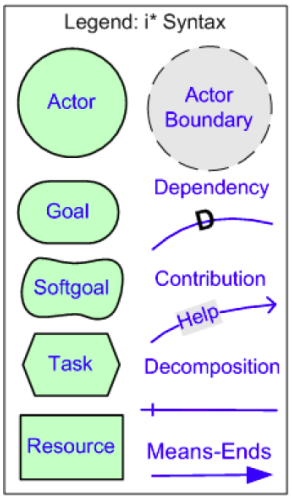

Fig. 2. i* Goal Model Syntax

C. Using Goal- and Process Models Together

J. L. de la Vara et al. [10] write about the use of goal models and process models as complements, and draw a connection between the two different perspectives by providing guidelines on how to derive goal models from process models. They discuss how one perspective supports the another, and how they can be equivalent in some aspects. For part of the development of our goal models, the derivation method has been used to achieve a level of consistency between the goal models and the process models.

The authors [10] argue that goal models and process models should be combined when "organizational procedures are (more or less) well-defined, but an organization expects a change in them as a result of the development of an [Information System] and the system must also support fulfillment of some strategic goal". We judge that this situation corresponds to the one at the case company.

D. Systematic Analysis of Goal Models

Horkoff et al. [11] describe three different methodologies of systematic goal model analysis: Alternative effects (forward analysis), achievement possibilities (backward analysis), and domain-driven analysis (mixed). In forward analysis, the leaf intentions in the model are first identified. Different implementation alternatives are evaluated by first attempting to satisfy all leaves, then none of them, and evaluate both situations. The elements in the models are given labels based on their level of satisfaction. The label notation can be found in Fig. 3.

In backward analysis, the roots of the model is looked at. It is checked if and how it is possible for all roots to be fully satisfied, and if and how minimum targets can be achieved. From the latter part of the analysis, the targets can be increased gradually to find the maximum targets which will still allow a solution. The mixed approach uses the model to attempt to answer questions about the domain. If the model cannot be used as such, the possibility of expanding the model is considered.

Source Label		*Contribution* Link Type						
	Name	Make	Help	Some+	Break	Hurt	Some-	Unkn.
✓	Satisfied	✓	✓.	✓.	✗	✗.	✗.	?
✓.	Partially Satisfied	✓.	✓.	✓.	✗.	✗.	✗.	?
↯	Conflict	↯	↯	↯	↯	↯	↯	?
?	Unknown	?	?	?	?	?	?	?
✗.	Partially Denied	✗.	✗.	✗.	✓.	✓.	✓.	?
✗	Denied	✗	✗.	✗.	✓.	✓.	✓.	?

Fig. 3. Label descriptions.

In our case study, we use forward analysis as described by Horkoff et al. [11] to analyze the two different implementation alternatives (as-is and to-be) of the goal models.

E. Evaluation of Modelling Languages

Horkoff et al. [4] conduct case studies where they evaluate the expressiveness and effectiveness of a requirements modeling language, and discuss evaluation of modelling languages in general. In their study, they find several challenges relating to both expressiveness and effectiveness of the modelling language. Since we refer to this paper to evaluate our models and answer our research questions, the procedure is explained in more detail in section 4: Methodology.

III. Related Work

A. API Ecosystems

An API is a communication channel that provides business assets and services to the end user through application software based on the given specification to boundary objects[2] [12] and supports third party to participate in the development of application software. Therefore, APIs have ability to attract new actors to the existing platform. The architecture of APIs depends on the platform, and relies on continuous development based on technology and API usage [2] [12]. The goals of developers and other stakeholders have to be considered during the development of an API architecture. Actors need to receive continuous value from the API in order for the software ecosystem to be said to be suitable and successful. An inefficient API design may reduce efficiency of the existing platform and impede its service provision [12].

Considering the importance of the API ecosystem design, a good analytic tool such as a model or a combination of models could be highly valuable when developing or making changes to an API.

Knauss et al. [13] discuss trade-offs caused by the openness in software ecosystems: the trade-off of keeping communication open and transparent while keeping some things confidential, and the trade-off between acting globally (on a long-term strategy) and locally (answer current context specific needs).

Architectural ecosystem platform openness is characterized by the ability to provide customer access to business assets and get free technical service [6]. Extreme openness facilitates users to modify and enhance the platform to use and run

their instants, while closed openness has limited and carefully controlled instances that can be run on the platform [6].

A systematic literature review conducted by Manikas et al. [7] describe that a software ecosystem have an advantage when constructing large software systems. The paper analyses the different definitions of an ecosystem and how they behave on a different platform.

Berger et al. explain a conceptual framework about variability mechanism across the ecosystem [6]. The paper mainly focuses on framework using data for each ecosystem, static analysis of data and the existing dynamic and static variability mechanisms.

B. Goal Models and Software Design

There are many papers available discussing the use of goal modelling in the context of requirements engineering. In addition to the paper by Yu [9], Lamsweerde [14] shows an example of goal-oriented requirements engineering in his case study from 2001. In the paper, he defines goals as the objectives a system should achieve. He argues for how goal modelling can help with verification, validation and elaboration of goals, as well as with elicitation of requirements.

There are also a number of papers discussing modelling and design of software ecosystems. Yu and Deng [15] use a strategic modelling approach based on the i* framework with the purpose of the understanding of software ecosystems. Their models are based on examples taken from earlier work by Jansen et al. [16], Bosch [17], Popp and Meyer [18], and Popp [19], and are not meant to be an accurate reflection of an existing ecosystem. The authors state that the quality of the models can be improved by applying a "systematic method for knowledge acquisition and validation with domain experts".

In a paper by Yu and Mylopoulos [20], the authors illustrate how the i* framework for goal modelling can be used in a design context, though for software process design.

In a recent paper, Sadi and Yu [21] propose a goal-oriented approach for designing open software platforms. The authors' outlined approach to model and analyze requirements focuses on requirement trade-offs, and aims to help with the decision of which platform design is the preferred one, based on a compromise between the assessed trade-offs. This paper outlines a series of steps for describing design decisions, modelling, and evaluation of the models (provided by the non-functional requirements framework) which were useful to us.

In a systematic literature review, Horkoff et al. [3] provide a detailed overview of the literature discussing the use of goal models downstream. The approaches of using goal models throughout the entire life cycle (e. g. for elaboration and validation of requirements) are discussed, and the authors point out the tendency to focus on new solutions instead of validating previously proposed solutions. In our thesis, we propose to apply previously discussed solutions (e. g. the i* framework) on a new domain (an API ecosystem).

IV. Methodology

A. Research Context

In this case study, we are looking at the case of one company's API ecosystem. The case is a company in the embedded systems industry, which manufactures and distributes mechatronic devices. Annually, the company is estimated to manufacture more than sixteen million units. The company develops profiles for the devices, which are unique for each model. The profiles can be described as sets of detailed data which decide the functionality of each device, and how they will interact with the communication protocol.

The communication protocol is used to transfer data between devices. It establishes communication among the applications of different devices and allows the end-user to manage their system (a number of mechatronic devices controlled by one or several control units).

B. Data Collection Procedure

The investigation of the company's current and proposed API ecosystem design and workflow was done through qualitative data collection through interviews. The initial contact with the company generated data in the form of system architecture documentation, technical specification documents, and audio recordings from two workshops at the location of the company. The workshops sum up to nearly five hours of collected audio. These recordings were partly transcribed manually to provide us with an overview of the information given at the workshops. As the dialogue was not noted down word for word, time stamps were added to information with high relevance for the scope of this paper, making it easy to go back to the original recording as needed.

From the qualitative data, first drafts of the as-is and to-be goal models and process models were created. The as-is goal- and process model were developed to capture the challenges and bottlenecks identified through communication with the case company and discussions with the senior researchers. The improved goal- and process model describing the possible future API design and workflow (the to-be models) should capture suggested solutions to the challenges we defined in the as-is models, as well as their effects.

The i* framework [9], [15] is used to create the goal models. The i* goal model notation is summarized in Fig 2.

UML activity diagrams (with swimlanes for actors) is used to model the workflow of interest for the analysis, e. g. the device profile development process. We used the layered API modelling framework defined by Hammouda et al. [2] to sort the actors within the goal models (see Fig. 8 and 10 for high-level views of the finished models). As seen in the models, the actors are arranged in four rows, each row representing a layer. The business assets are modelled at the bottom of the model, followed by the API layer, API SW layer, and Domain layer at the top.

Our intention for structuring the goal models this way is to provide a comprehensive overview of the current and future suggested API design, and to test the application of the layered API framework as an analytic tool for API design analysis.

The models were shown to the senior researchers as well as our contact person at the company, who is one of the case company's API owners and involved in the profile development process. From the feedback received from the supervisors on these initial models, we corrected and refined them to better capture the design and workflow of the API

ecosystem. Meetings with the senior researchers were held weekly, and additional meetings with the supervisors were planned as needed. Our contact person at the company was notified of our progress after each model iteration (four in total) in order to create an opportunity for feedback through written communication or remote meetings.

At the end of the study, a remote meeting through video was held with the researchers and our contact person at the company. The aim of the meeting was to verify the models in order to create the final versions. The data collection activities have been listed below and summarized in Fig. 4.

1 Investigate objectives of the company during the cross company workshop. The workshop was held with the different organizations and senior researchers involved in the research project conducted at Software Center [1].

2 Investigate previous work from research project from the software centre at Gteborgs universitet.

3 Analyze technical specification documents from the company.

4 Collect qualitative data to understand companys profile development workflow with company representative and senior researcher. This was done at a workshop at the location of the company.

5 Collect related literature to understand modeling design in relation to API ecosystems.

6 Communicate with companys representative through email for validation of the first draft as is models.

7 Sent questions for further clarification of previous answers.

8 Held cross company workshop with the company representative and the senior researchers.

9 Received a feedback on the second draft of models from the company representative.

10 Online video meeting with the company representative and the senior researchers to validate the final models.

Goal	Data Collection method	Subject
1	Cross company workshop, voice recording	Company representatives, Senior researchers
2	Previous related work	Göteborgs Universitet Software Centre
3	Cross company workshop,email request	Company documents
4	Personal Interview at company workshop, voice recording.	Company representative
5	Literature review	Collected literature related to the research topic
6	Communication via email	Company representative
7	Sent via email	Company representative
8	Cross company workshop	Company representatives, Senior researchers
9	Received via email	Company representative
10	Online video conference call	Company representatives, Senior researchers

Fig. 4. Data collection procedure.

C. Analysis Procedure

For the analysis of the goal models, we used the method of forward systematic goal model analysis as defined by Horkoff et al. [11]. Each goal and softgoal in the model is given a label showing its satisfaction level, which is dependant on how the tasks in the model contribute to each goal. Using this form of systematic goal model analysis helps us to understand what effects different goals and interrelations have on the ecosystem, as well as revealing previously unidentified issues with the models. Part of the analysis is automatic depending on the various types of contribution links, but human judgment is sometimes necessary when a goal has several contribution links - sometimes contradictory - leading to it.

The initial question to be asked for the forward analysis, as proposed by Horkoff et. al [11], is of the form ”How effective is an alternative with respect to goals in the model?” After labeling the as-is and to-be models’ goals, tasks and softgoals somewhere on the spectrum between fully satisfied and fully denied, the two different alternatives can be compared and the original question answered.

In summary, we check if the changes made in the to-be API ecosystem design and workflow provide solutions to the challenges and unmet criteria from the case company’s previous system design, and if any additional bottlenecks, unmet goals or new challenges can be elicited from the models.

In order to evaluate the ability of our models to capture static and dynamic aspects of the API ecosystem, we refer to the work of Horkoff et al. [4] on evaluation of modelling languages. Keeping our first research question in mind, we want to investigate how goal models and process models respectively can be used to capture static and dynamic aspects of the API ecosystem, and if they can effectively be used together. We evaluate the following attributes:

1) Expressiveness: The ability of the models to capture aspects of interest in the API ecosystem. As we model, we take note of if we are able to model everything we want to model. We take note of

- aspects we are able to capture in either the goal models or the process models but not both, indicating that the models work as complements,
- aspects we are able to capture in both the goal models

and the process models, indicating an overlap between the models, and

- aspects we want to model but cannot find a way to capture in the goal models or the process models.

2) Effectiveness: Effectiveness in this case refers to the simplicity with which the models can be used to capture the aspects of the ecosystem. We take note of

- how easily information can be captured in the models using the syntax of i* and UML respectively, and
- how easily the information corresponding to model concepts can be elicited.

In order to answer the second research question (aimed at exploring how the models can be improved for the purpose of API ecosystem design analysis), we complement the above evaluation with other relevant observations made throughout the model iterations. What changes did we make to the models, and why? How did they improve the models? Did they help us discover any previously unmentioned challenges?

V. Results

A. Overview

With the information gathered from the company at the workshops (see Appendix for interview questions and partial answers), we created four models visualizing our understanding of the current and future API ecosystem design and workflow at the case company:

- As-is goal model, visualizing the current API ecosystem design sorted in layers (Fig. 8)
- As-is process model, visualizing the current API workflow (Fig. 7)
- To-be goal model, visualizing the API ecosystem design sorted in layers with implemented suggested changes (Fig. 10)
- To-be process model, visualizing the API workflow with implemented suggested changes (Fig. 9)

The identified actors in each goal model have been sorted into layers as shown in Fig. 5. Descriptions of the actors in each layer can be found in the Appendix, and the reasoning for their placement in the API ecosystem is discussed in more detail later in this paper.

The models have been iterated with the feedback we received from the senior researchers and the company representative to more correctly and clearly display the different aspects in the API ecosystem and the relationships between them.

B. Iterations of Models

It is interesting to note that throughout the study, the models were used at different points in time. More focus was put into the development of the process models at the start. This enabled us to gain an understanding of the actors involved in the profile development process and their different actions. While the goal models were developed in parallel to the process models, these were initially incomplete and largely independent from the process models. The main focus from the goal models at the start was to identify actors, sort them into layers, and reason around possible softgoals for each actor.

Goal Modeling for APIs			
Level	**Layer**	**Actors in As-Is Model**	**Actors in To-Be Model**
4	Domain	• Mechatronic Device • End User • App Developer • Profile Specialist • Profile Owner • API Consumer • Communication Device Stakeholder	Unchanged
3	App SW	• Mobile App • Controller	Unchanged
2	API	• Communication Protocol • Profile	Unchanged
1	Business Asset	• MD Data • MD System	• Database • Feature Model

Fig. 5. Company-Specific API Ecosystem Layers.

Fig. 6 provides a high-level view of an early version of the as-is goal model (legend can be found in Fig. 2). The details in the model are not important for this section, but the relative lack of detail (compared to the later version) should be obvious. The process models were made more complete from the start, although these were also iterated with received feedback from the senior researchers (concerning visual representation) and the contact person at the case company (concerning workflow validity).

As our models work as visual representations of the API ecosystem design and workflow, it is important how they look. Our initial goal models (e. g. Fig. 6) were created using Creative Leaf [22], a free online i* modelling tool. As seen in the high-level view, Creative Leaf makes it easy to distinguish between softgoals, goals and tasks at a glance by giving them vastly different colours by default.

As the goal models became more complex, we moved over to another tool, OpenOME [23]. OpenOME is an open-source requirements engineering tool which allows for more refined shaping of dependency-, contribution-, and decomposition links, giving the opportunity to make the models more readable even as they grow more complex. This became important as the actors and the dependencies between them increased with each iteration, threatening the readability of the goal models.

C. As-Is Workflow and API Design

The workflow model (Fig 7) shows the current process which is followed in order to develop the device profiles, which work as the API for the units in a mechatronic device system. The process starts with a profile specialist, who works in a team of developers to create a profile model. The model presents an idea of what the profile is going to look like, and its creation involves several steps. After the requirements of the physical product the profile is going to be used by have been defined, corresponding data points have to be selected and sorted into appropriate classes.

When the model has been developed, it is sent to an actor called the Profile Owner (PO), who verifies the model. The PO checks that the positions of the data points make sense, and

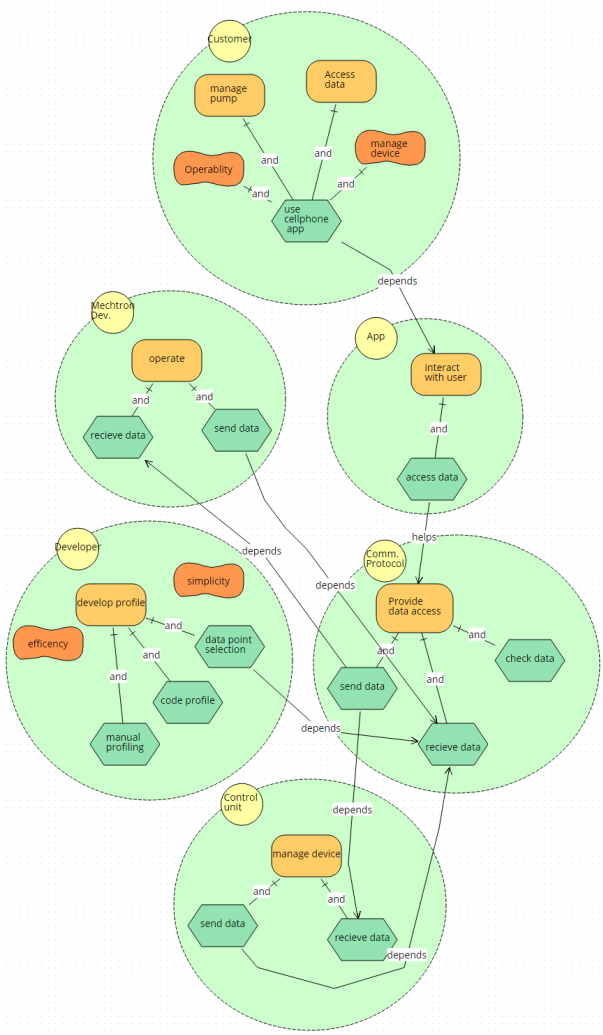

Fig. 6. Goal model: First draft. High-level view.

that their positions are consistent with the placement of similar data points in previously developed profiles. If the PO does not approve of the profile model, they communicate suggested changes using Git, and the development team reworks the model. The PO also keeps a record of these changes in a system. Up until recently, the PO and development team would have a physical meeting instead of handling change requests and change tracking digitally. We have chosen to model the older process in order to analyze the effects of these newer changes to the workflow.

When the model is approved, the profile is implemented and a profile specification is written. The specification is an extensive document which can take months to complete, but the implementation is only dependant on the finalization of the data points and their position, and can start as soon as that part of the specification document is ready. It is the PO's responsibility to describe the functionality of the implemented profile. Product ID's and serial numbers are hard-coded into the profile, and the completed specification document is saved as a PDF in a file-folder structure. The profiles are not publicly available, but neither are they confidential; a costumer can request a specific profile, even though it's not a common occurrence.

The as-is goal model is partly derived from the process model since the actors involved in profile development play a vital role for the API design, and are affected by most of the proposed changes. Fig. 8 is a high-level view of the model, and is not meant to be read in detail. Fig. 5 shows which actors are present in each layer, while detailed overviews of selected actors can be found later where we analyze the difference between the models. The PO, profile specialist and other developers can be found in the domain layer of the model together with the end-users of the product and other stakeholders.

The API Software layer contains a cellphone application, which enables the end user to manage the devices and collect data. We also consider the controller, which is a control unit used to manage a network of mechatronic devices, to be part of the API SW layer. The devices and their control until is referred to as a mechatronic device system (MD system). The controller contains a special profile which is a subset of all the device profiles in the system. The controller profile is produced by another department within the company.

The API layer has two actors: The communication protocol (a static communication protocol used to convey data between devices) and the profiles. The controller in the API Software layer depends on the profiles to work, and has two-way dependencies to the communication protocol.

Lastly, the business asset layer in the as-is goal model contains the aforementioned MD system. Data, which is stored on the cloud, is also considered a business asset.

D. Challenges and Proposed Changes

In the current workflow, the main challenges we elicited from the data are to:

1) Limit the time spent on creating a profile model.
2) Limit the time it takes to approve a profile model.
3) Keep a level of consistency between different profiles.
4) Limit the time spent on documentation.
5) Limit maintenance costs.

Since much of the process is done though manual work (defining data points, sorting them into classes, checking consistency, writing the specification), the development of a profile is time-consuming. The profile is validated by one person, and controlling the position for a thousand data points and making sure they are consistent with previously developed profiles is not a simple task. Occasionally (by coincidence), the PO have several profile models lining up for validation at once, creating a queue.

After a profile has been validated, the profile specification has to be written from scratch and the profile needs to be implemented before being utilized. If a profile needs to be changed, it goes through much of the same process again (although more reuse oriented).

To at least partially overcome these challenges, the implementation of profile feature modeling has been proposed. This would allow the profile specialist to select features rather than defining the data points, and having much of the artifacts

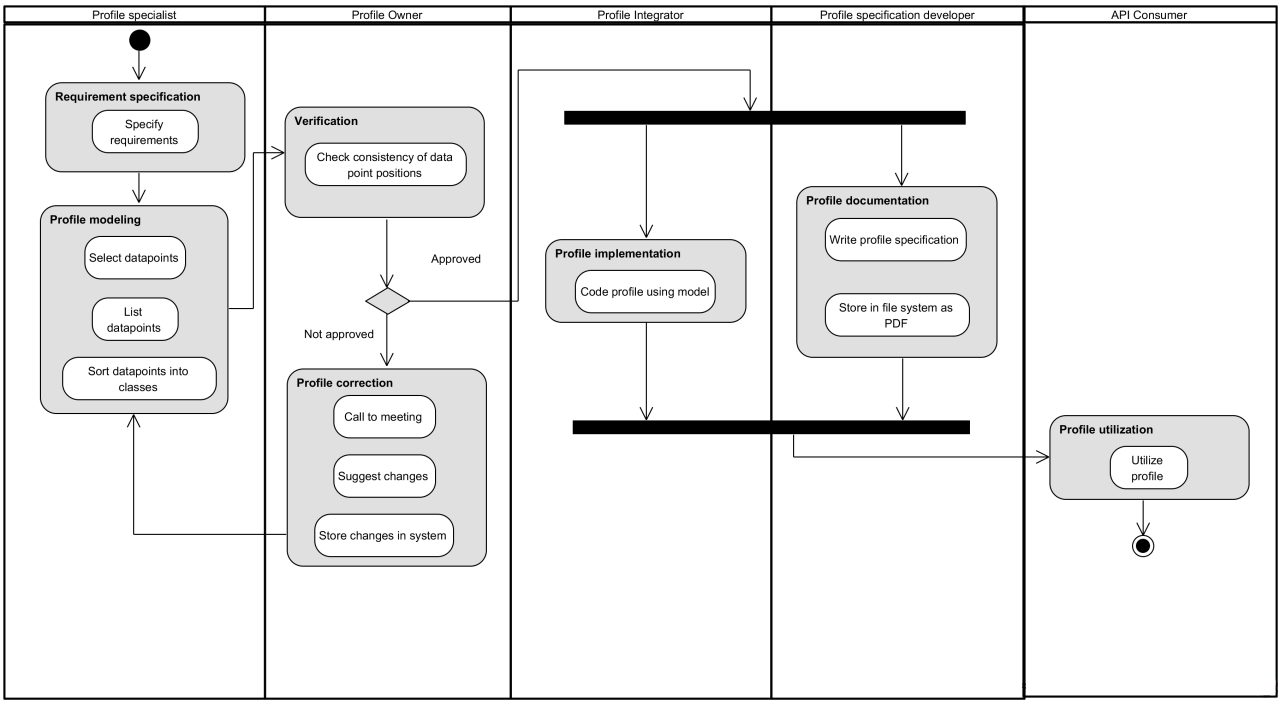

Fig. 7. As-is activity diagram.

(profile model, profile, specification) generated automatically. In its initial stage, this solution would only include the most common features. In an ideal future however, minimal manual labour would be required to develop and maintain the profiles.

E. Benefits and Drawbacks of Proposed Changes

The to-be workflow (Fig. 9) shows the changed process after the implementation of feature modelling and digital profile model change tracking.

After the profile specialist has selected the desired product features, the profile model is generated through feature modelling. The verification process of the PO remains the same, but the reporting of the results is as previously mentioned handled differently (Git, changes stored in system).

The feature model system has taken over the Profile Integrator role, as well as creating the profile specification. With the Profile Specialist's role made simpler, there is a possibility that the PO could take over the activities of requirement specification and feature selection. This would eliminate the need for communication between these roles, and remove potential bottlenecks tied to the interaction.

The to-be goal model was analyzed using forward systematic goal model analysis as described in the Methodology section in this paper. Fig. 10 shows a high-level view of the model. (More detailed views are presented later in this paper.) Special attention was paid to softgoal satisfaction in the to-be model compared to the as-is model.

Fig. 11 gives an overview of how the softgoals in the goal models are affected when the changes are implemented. The table only includes the actors affected by the changes made to the models. The initialisms stand for:

D	Denied
PD	Partially Denied
C	Conflict
PS	Partially Satisfied
S	Satisfied
U	Unknown

In the remaining part of this section, we explore the to-be alternative API ecosystem design and its implication on softgoals. The softgoals represent the qualitative goals for the different actors in the API ecosystem. In our collected data, most of the desired changes in the API ecosystem design was phrased in a form which could easily be modelled as softgoals (e. g. "[A development activity] should be as fast as possible"). Therefore, we chose to focus on the affect on softgoals for the two alternative API ecosystem designs. To compare the models, we focus on the actors in which the softgoals are contained, included in Fig. 11.

1) Profile Specialist: The team of profile developers, the profile specialist included, wants the profile modeling process to be as fast and easy as possible. In the as-is model, this goal is hurt by the manual work of defining data points and sorting them into classes (Fig. 12). If the profile is built upon earlier work, and does not have to be developed from scratch, this helps the goal of fast profile modeling. Since its not uncommon for profiles to be developed from scratch, the goal is considered partially denied in the as-is model.

In the to-be model (Fig. 13), the profile specialist simply selects the desired device features, and feature modelling is used to generate the data points and the profile model (Fig. 14). This saves time for the profile specialist, and the goal is at least partially satisfied.

In addition to the modeling of new profiles being fast,

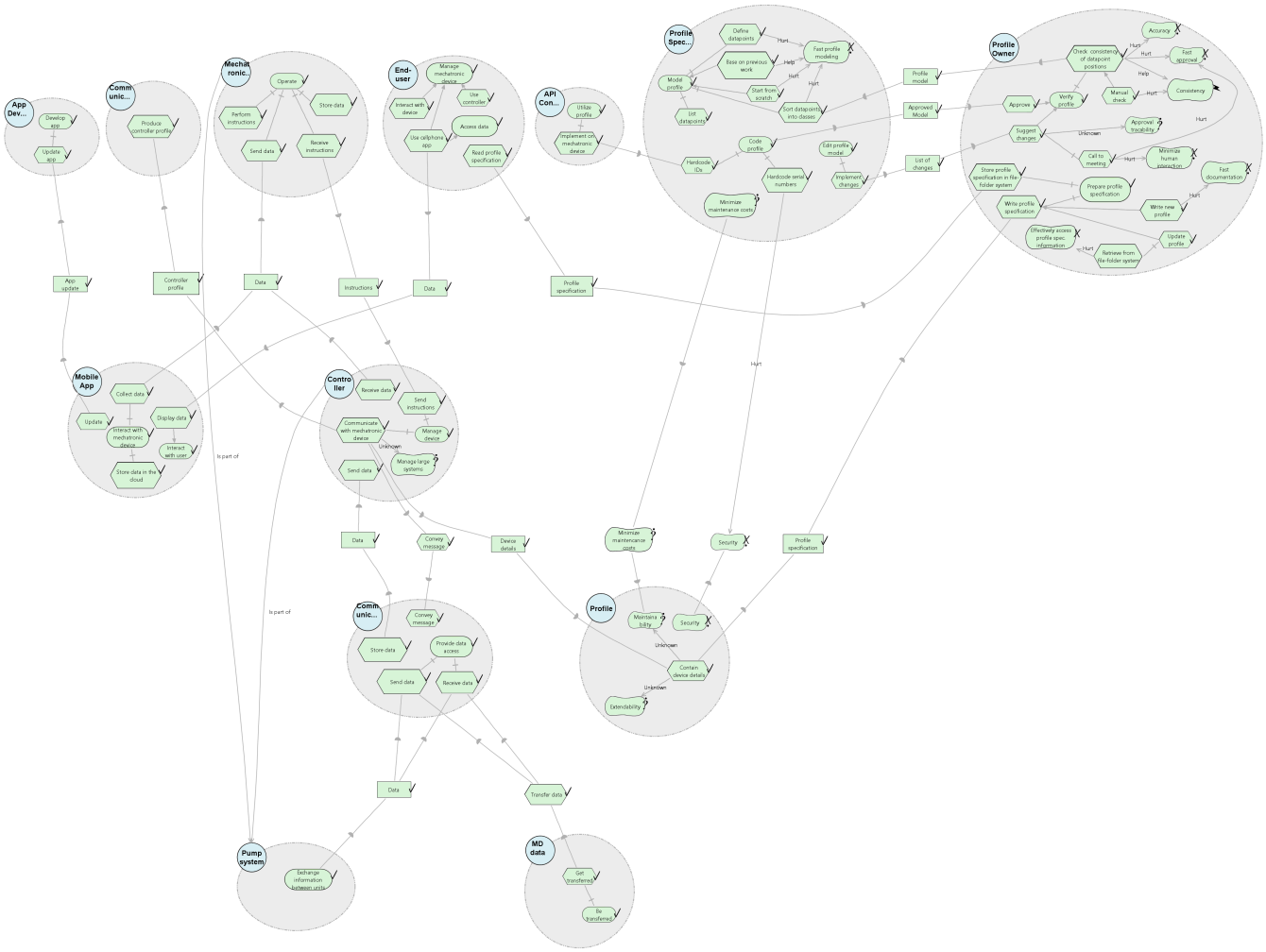

Fig. 8. As-is goal model, high-level view.

updating existing profiles should not be resource-intensive. The goal to minimize maintenance costs depends on the maintainability of the profile itself. In the as-is goal model, the degree of satisfaction of this goal is unknown. In the to-be model however, the devices profile should be automatically extended to include new modules when needed, helping profile maintainability and decreasing maintenance costs.

2) Profile owner: Profile verification: When the profile owner (Fig. 15) verifies a profile model, they should check that the data points are consistent with previous models. This is done to help consistency, but doing so manually could hurt consistency due to different profile owners having different ideas of in which class a specific data point belongs, unclear sorting standards, etc. Accuracy is hurt for the same reason, and the large number of data points which needs to be checked means the verification process takes time. Therefore, accuracy and fast approval are at least partially denied in the as-is model, while there is a conflict in consistency.

In the to-be model (Fig. 16), the verification process is left unchanged. However, since feature modeling is used to generate the model automatically, a smaller effort is required by the profile owner. Consistency is helped by the feature model making consistent choices when sorting data points into classes, indicating the profile owner no longer needs to check all data points if the feature modelling system works. Consistency is therefore marked partially satisfied in the to-be model, while theres now a conflict in fast approval; the satisfaction rate depends on how much in detail the profile owner checks the data points. Accuracy remains unchanged in the model, though it can be argued that theres less of a need for the profile owner to be accurate if feature modeling can accurately sort the data points into classes.

To minimize human interaction in the verification process, the use of Git is introduced in the to-be model (in practice this has recently been implemented at the case company, but not included in the as-is model in order to analyze the benefits). When suggesting changes to the profile model, the product owner would previously call for a meeting. The possibility of a meeting depends on the availability of the stakeholders (profile owner and profile specialist), which can slow down the verification process. Suggesting the changes in Git instead helps to minimize human interaction in the to-be model. As a part of the changed routine, the model changes are also reported in a system which makes it possible to trace the

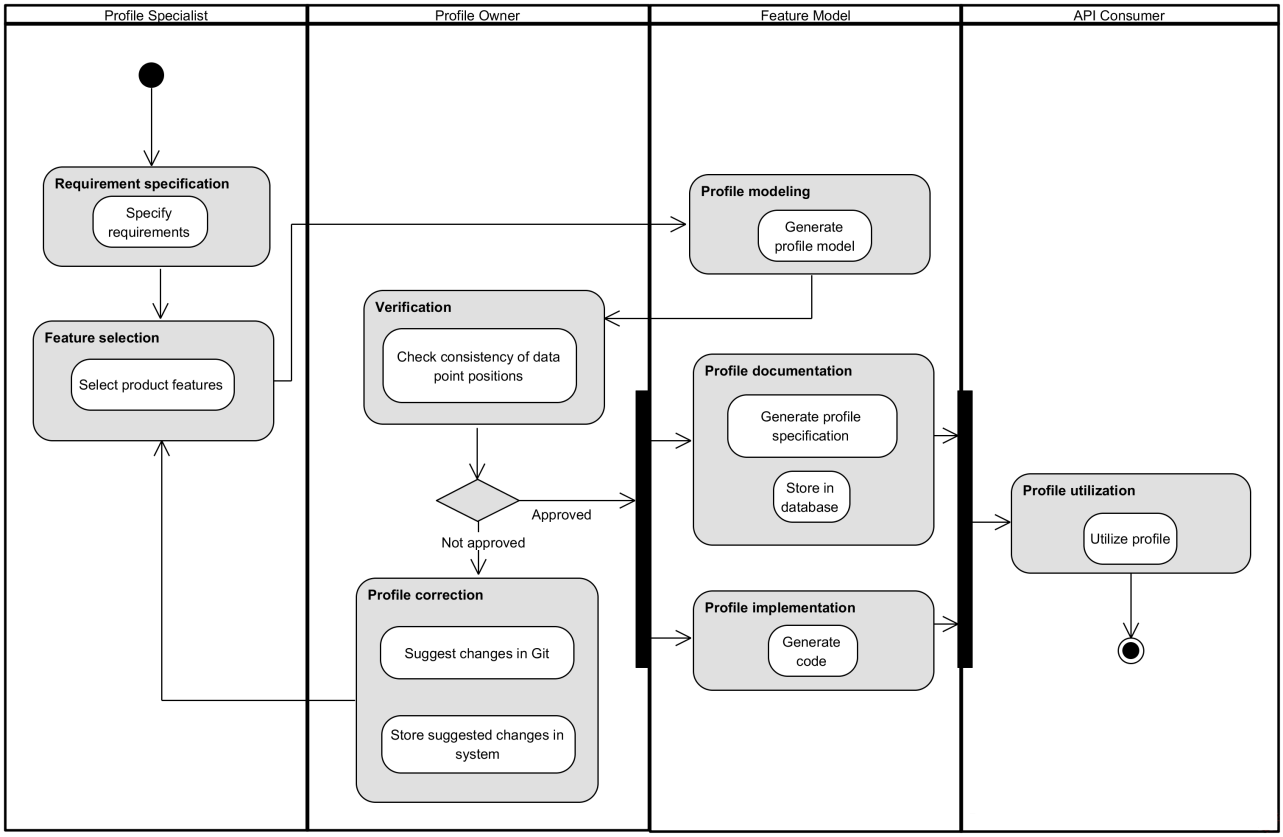

Fig. 9. To-be activity diagram.

changes to the profile.

3) Profile owner: Profile specification documentation: The profile specification is an extensive document which takes a long time to write if written from scratch, as is the case in the as-is model (Fig. 15). The goal of fast documentation is partially denied. In the to-be model, the specification is at least partially generated together with the profile model (Fig. 14), which helps the goal of fast documentation (Fig. 16). The goal is also partially satisfied in the to-be model by an improved way of storing the profiles. The addition of a database (Fig. 17) makes it easy to access and change specific profile specification information by querying the database. In the as-is model, the specifications are stored as PDF-files in a folder system, which hurts the goal of effectively accessing the information needed to update an existing profile.

4) Controller: The controller should be able to manage large systems, which is currently a limited ability. The controllers profile is a subset of all the profiles of the mechatronic devices in the system, which means the system can only contain as many devices per controller as the controller profile allows. If the profile can be extended past its current limits it would partially satisfy the goal (Fig. 18), although there is likely to always be an upper limit for how large a system one controller can handle.

5) Profile: In the as-is model, serial numbers are hard-coded into the profile (Fig. 19). This hurts security, so in the to-be model, serial numbers are accessed from an external source (Fig. 20). It is unclear how the current API system affects maintainability and extendability. In the to-be model, the profile can be automatically extended to include additional functionality of new modules, which satisfies the extendability goal. Maintainability are also helped by this change.

VI. Discussion

We divide the discussion into two sections. First, we attempt to answer the research questions with the help of the insight gained during the model development process. After that, we discuss the different concepts which led to the answers in more detail.

A. Answers to the Research Questions

1) RQ 1.1: How well can goal modelling be used to capture the static aspects of API ecosystems?: The goal models in this study captured most of the relevant aspects elicited from the data, were received well by the case company because of their level of detail. The contribution links to softgoals and the dependencies between actors in the API ecosystem provided especially valuable information to the company.

2) RQ 1.2: How well can process modelling be used to capture the dynamic aspects of API ecosystems?: The process models were important tools for the researchers to understand and get an overview of the current and planned workflow. The models, which are UML activity diagrams, effectively communicated how different actors in the API ecosystem interact. However, they were not frequently used in the analysis of the ecosystem once the workflow was understood by the

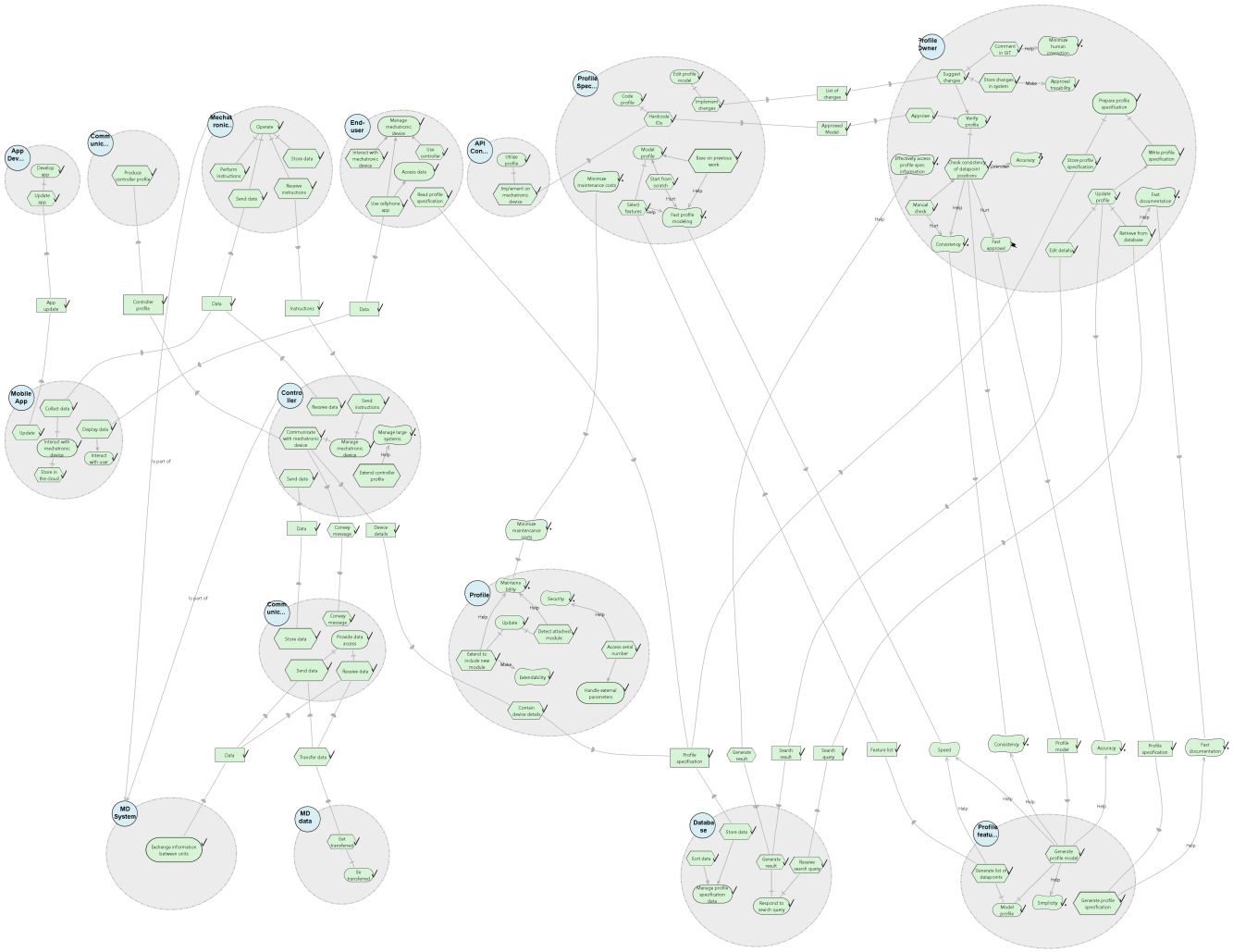

Fig. 10. To-be goal model, high-level view.

Actor	Softgoals	As-is	To-be
Profile specialist	Fast profile modeling	PD	PS
	Minimize maintenance costs	U	PS
Profile owner	Accuracy	PD	PD
	Fast approval	PD	C
	Consistency	C	PS
	Approval traceability	U	S
	Minimize human interaction	PD	PS
	Effectively access profile spec. information	D	PS
	Fast documentation	PD	PS
Controller	Manage large systems	U	PS
Profile	Security	PD	PS
	Flexibility	U	PS
	Maintainability	U	PS
	Extendability	U	S

Fig. 11. Softgoal Analysis.

researchers. The aspects captured by the models did not provide new insight for the case company, i. e., the activity diagrams were less expressive than the goal models.

3) RQ 2.1: How can goal models be improved for the purpose of API ecosystem design analysis?: The main challenge in the analysis of the ecosystem using goal models was instances of conflicting contribution links between tasks and softgoals. In many cases, the level of fulfillment of a softgoal is not clear to the researcher. For the analysis to be accurate, more qualitative data is required from the modelled actors in order to make an appropriate judgment and compare two API ecosystem design alternatives.

4) RQ 2.2: How can process models be improved for the purpose of API workflow analysis?: In our process models, action duration (the time it takes to complete an action) is not modelled. Many of the softgoals modelled in the goal models concern time efficiency, and being able to elicit potential bottlenecks causing time inefficiency from the process models could improve the analysis of the workflow. As previously mentioned, the order of events is something which can not be captured in a goal model (as the latter shows static aspects), but once the goal models had been finalized, the process models as they are designed in our study did not offer much valuable information for the analysis of the company's API ecosystem.

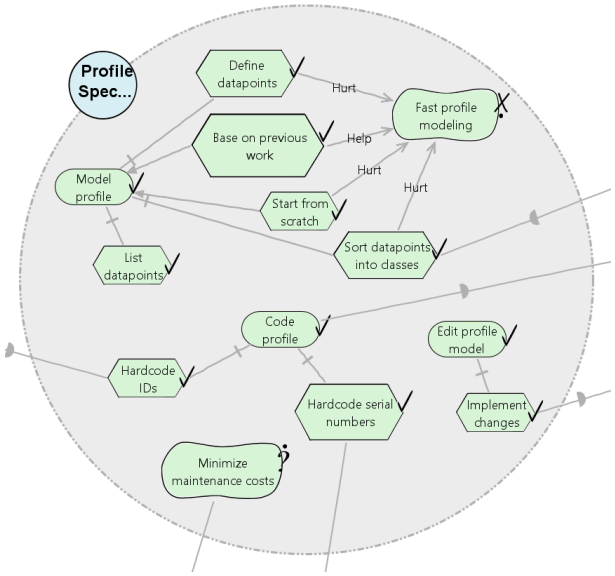

Fig. 12. Profile Specialist in the as-is goal model.

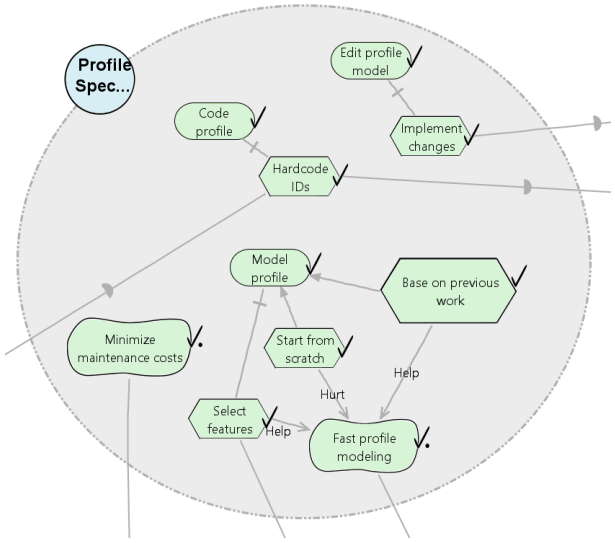

Fig. 13. Profile Specialist in the to-be goal model.

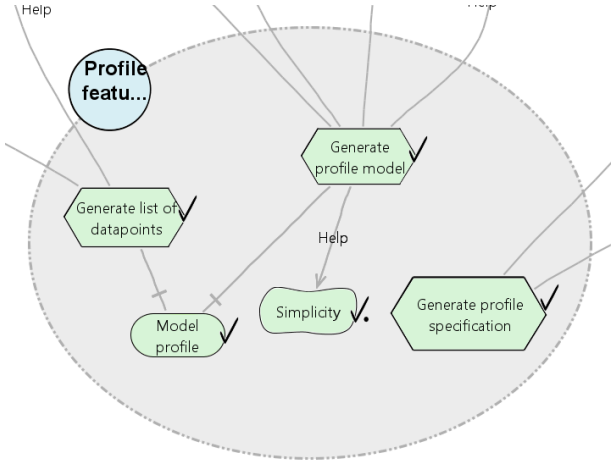

Fig. 14. Profile Feature Model in to-be goal model.

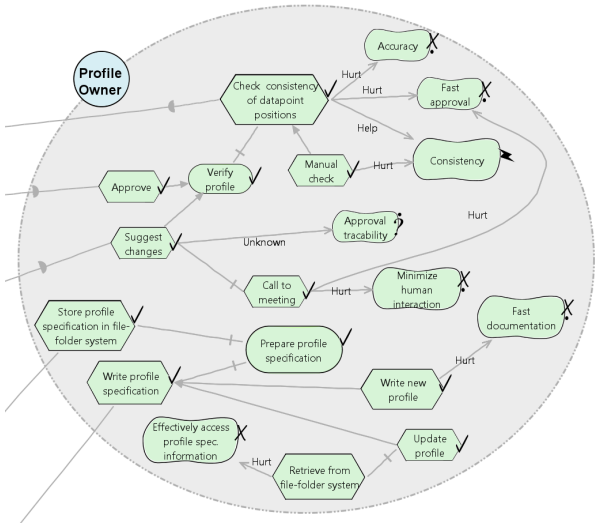

Fig. 15. Profile Owner in the as-is goal model.

B. General Discussion

1) Expressiveness of Models: The goal models can be partially derived from process models [10], and have therefore some aspects in common, providing a way to check the consistency between the models. The tasks in the goal models correspond to the actions in the process models, and the actors in the goal models are represented as swimlanes in the process models. De la Vara et al. propose that the process (or sub-process) itself denotes a goal in the goal model (e. g. 'Profile Verification' in the process diagrams can be found in the goal models as a goal decomposed into tasks). Softgoals, however, are only found in the goal models.

Modelling the different actors in the API ecosystem using i* goal modelling was occasionally challenging. While we found it easier to show the structure of the API ecosystem using goal modelling rather than using process modelling, it was not always obvious where in the layered API framework a specific actor belongs. A mechatronic device, for example, can be a business asset. It can also belong to the API Software layer, if we look at it from the perspective of its software which is accessing the API layer. In our final goal models, it has been included in the Domain layer since it is communicating with the Controller, which in turn is using the API.

The decision of where in the API ecosystem an actor belongs could be discussed at length (and in fact has been), and would benefit from several models where the actors are placed in different layers depending on perspective and different ways of reasoning about their most important function in the ecosystem. Only one set of final as-is and to-be models was developed within the scope of this study.

We believe that using the layered API framework to model the API ecosystem as a goal model increased the readability of the models. Extensive goal models tend to spread out like a web, sometimes making it difficult to get an overview of the models. By structuring the actors into clearly defined layers, actors and shared resources are easy to locate, both while

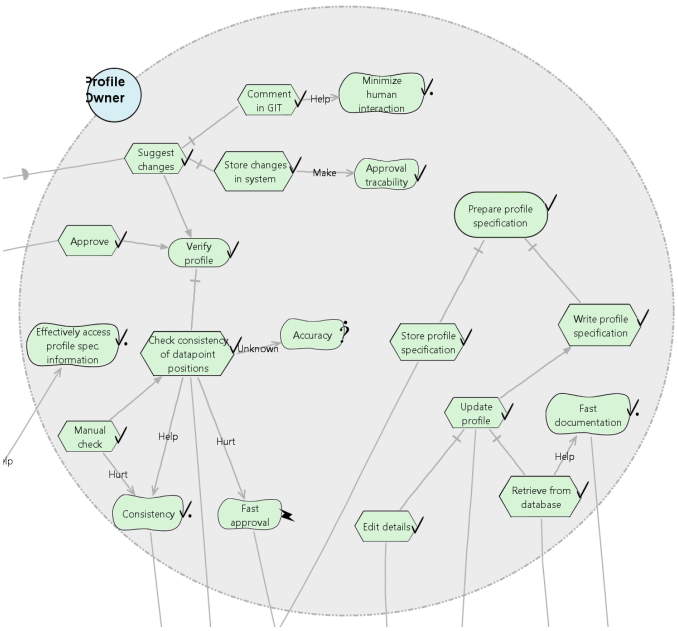

Fig. 16. Profile Owner in the to-be goal model.

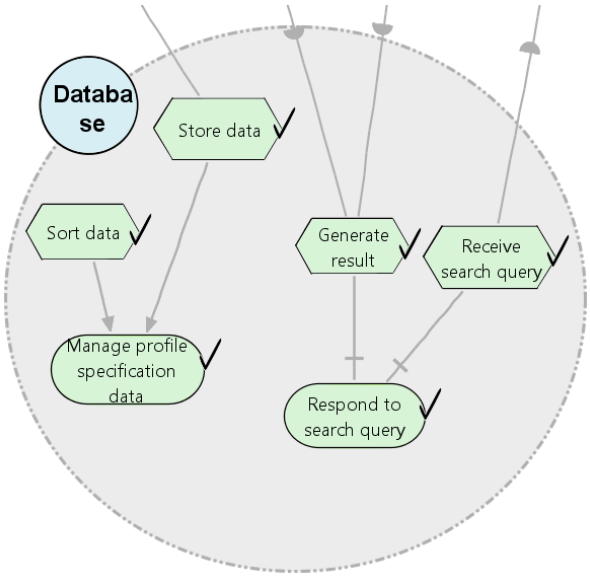

Fig. 17. Database in the to-be goal model.

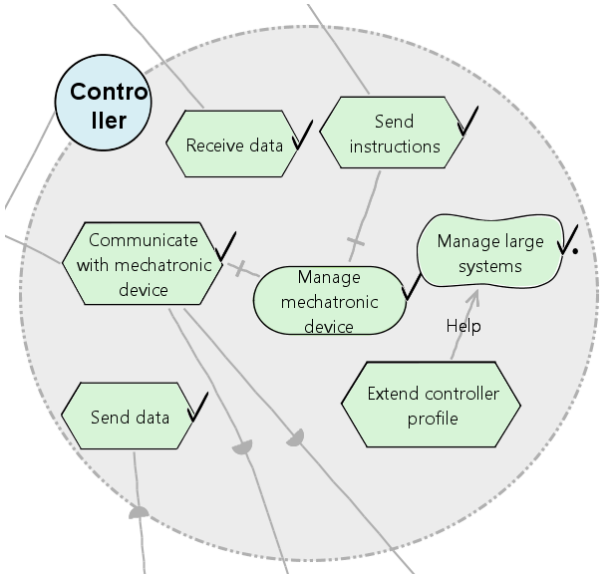

Fig. 18. Controller in the to-be goal model.

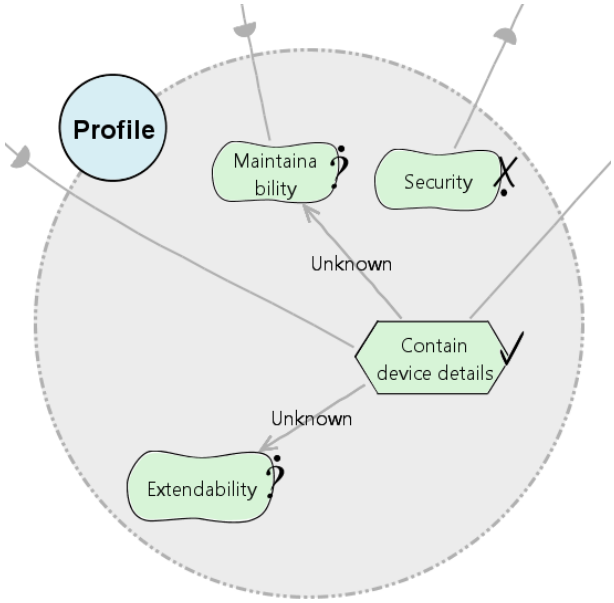

Fig. 19. Profile in the as-is goal model.

communicating about the models and during the modelling effort itself - if we know what role an actor has in the API ecosystem, we know where to look for it.

However, the layered structure provides limitations for the direction of dependency- and contribution links. This is made especially clear in the to-be goal model (Fig. 10) where there are many links between the Domain and the Business Asset layers. Links running too close in parallel and sometimes crossing each other might decrease readability of the models. If we did not care about the layers, we might have been able to get around that problem by placing connected actors closer to one another, making the links easier to follow. Respecting the layers, this is not an option.

For the sake of comparison between the as-is and the to-be goal model, it was tempting to model aspects in the system as actors, even though they are not particularly 'active'. The database in the to-be model replaces the current file-folder system. While the database can perform an action (e. g. respond to a search query), the file system is passive. This results in the database showing up as an additional actor in the to-be model, and the notion that it's replacing another form of file-storing system is not clear, which gives the as-is goal model a sense of incompleteness.

As we iterated the models to make them more accurate and detailed by accounting for elicited data, we sometimes had to make design decisions based on a trade-off between completeness and simplicity. This is especially true for the

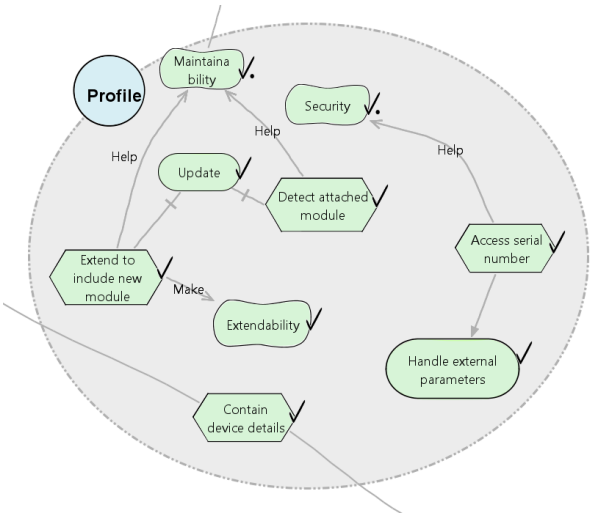

Fig. 20. Profile in the to-be goal model.

goal models. The models should be complete enough to be useful for analysis, e. g. we should be able to infer challenges and bottlenecks from them. At the same time, they should be understandable. As we add more tasks or find more softgoals and goals for an actor, and add more actors to the API ecosystem model, the models became more difficult to understand (possibly without gaining much in terms of usefulness of the models in an analysis), which has a negative impact on the effectiveness of the model.

We have limited the scope of the goal models by excluding actors which are judged to be too far from the actors in the API layer and the changes implemented in the to-be model. An example of such an actor is the Mechatronic Device Manufacturer. Within the actor boundary, the number of goals and tasks are limited by not showing more than one or two levels of abstraction (a task can have subtasks, but the subtasks do not have subtasks themselves). The softgoals derived from the data is included in the models, while highly probable but unmentioned ones are excluded unless the changes in the to-be model are expected to have a considerable effect on the softgoals.

In general, human judgment was frequently used both while developing the goal models and when analyzing the models using systematic forward goal model analysis [11]. As previously mentioned, the contribution links tells us (and the semi-automatic analysis software) how a task contributes to a softgoal, but not how much. Does several “hurt” contribution links make a softgoal denied, or partially denied? Does several “help” contribution links tip the scale to “partially satisfied” even with an active “hurt” link present? These decisions can only be made subjectively, and is up to the actor to judge (in the case where the actor is a person) rather than the person doing the modelling. In the more uncertain cases, we chose to use the ”Conflict” label.

2) Effectiveness of Process Models: The layered API framework [2] introduced in the first section of this paper has only been applied to the goal models developed in this study. Using the layers in the process models as well was found to be more tricky, especially since we chose to only model the workflow for part of the API ecosystem (namely, the development of a mechatronic device profile). The process models involve at most two layers - the Domain (developers and PO) and the Business Asset layer (the profile feature model).

Since we were mainly interested in the differences between the as-is and to-be models, other workflows were not modelled as these are expected to remain largely unaffected by the changes in the profile development workflow. However, had we had time to develop another process model spanning over all the layers in the ecosystem (e. g. how the user accesses data using the mobile app) a possible solution for displaying the layers in a UML Activity Diagram could be to use swimlanes. This would likely happen at the cost of displaying the actors as swimlanes, as creating a grid (with actors on vertical swimlanes and API layers on horizontal swimlanes) seems unnecessary as an actor only perform its role in one layer.

3) Compatibility of Models: The use of the process models and goal models together provided some value for the analysis of the API ecosystem. The models essentially represent two different ways to look at the same thing (the API ecosystem in terms of goal fulfillment and the API ecosystem in terms of activities), and share similarities as discussed by de la Vara et al. [10]. Even so, the use of process models in addition to the goal models made it possible to gain an initial understanding of the underlying process of profile(/API) development of the company. In other words, the natural tendency within the study was to use the two types of models sequentially rather than together.

If the vocabulary is kept consistent between the models (i.e. tasks in the goal models being phrased exactly the same as actions in the process models) the different models can be used together to infer different information about the same aspect of an API ecosystem. For example, one of the goals of the Profile Owner is to verify the profile. The swimlane representation of the PO in the process diagram shows the actions they have to take in order to meet this goal, and in which order. In the goal model, we can see how these actions (or tasks) affect the qualitative aspects important to the PO (or softgoals). If the vocabulary is inconsistent between the models, this becomes much harder, which works as an argument in favour of using derivation techniques (such as the one suggested by de la Vara et al. [10]) to increase model consistency.

Dependencies are explicitly modeled in the goal models, which means that the necessary sequence of actions can be implied in the from the dependencies of shared resources created by an action. For example, the action ”Check consistency of data point positions” (undertaken by the Profile Owner) depends on the development of the profile model, which in turn depends on the task ”Sort data points into classes” (undertaken by the Profile Specialist or Feature Model).

4) Value of the Models to the Case Company: By creating the models based on the needs of a real company, we have been able to experience and evaluate the usefulness of the models as analytic tools. The models provide an overview of the ecosystem which makes it possible to identify the important actors and how they are connected.

In addition, the clear overview of the softgoals in the goal models provides an opportunity to develop a goal-oriented API

ecosystem design. The presence of unfulfilled softgoals helps us ask relevant questions which enables us to start thinking about ways to resolve unfulfilled requirements. Though the goal models were initially perceived as difficult to read by the company representative, the details in these models got a positive response and could serve as a foundation for further investigation into a different perspective of the API ecosystem design in the long run.

At the latter half of the study, most of the attention of the group of researchers and the company went to the goal models. The process models worked largely as an initial one-way communication tool between the company representative and the researchers: We modelled the current and suggested future profile development processes based on the data, and verified our understanding of the workflow by asking for feedback. The process models did not tell the company anything they did not already know, but served as a way for the researchers to get a clear idea of what is done when, and by who.

5) Suggested Improvements to the Models: To further improve the models for the purpose of API ecosystem analysis, they could benefit from a way to quantify the effect of tasks on goals (goal model) and a validity check of the goals with the concerned actors. At the same time, the level of complexity should not be so high that the models are difficult to understand, while still providing necessary information for an ecosystem analysis. A too simple or abstract model shows an incomplete picture, and is not enough to analyze the effect a change might have.

A complex model, such as our to-be goal model (Fig. 10) shows a number of actors which remain unaffected after the changes in the system has been implemented. Their lack of softgoals make them unimportant to the analysis unless their tasks influence the softgoals of other actors, which might imply that these actors should have either been excluded from the model (adding complexity without providing considerable analytic value) or have been researched more in detail to find out their softgoals, and how these are affected (if at all) by the implemented changes.

There are some aspects which were not modelled in either of the selected model types. An example of that is the challenge of the PO having many profile models to verify at one point in time. This is an important challenge which should be addressed when coming up with an alternative workflow. Modelling the many-to-one relationship between the "profile model" (i*-)resource/(UML-)object and "model profile verification" task/action could increase the models' potential to be used as an analytic tool for discovering bottlenecks in the API ecosystem design.

We think that displaying the duration of a task/action would also increase the ability to use the models to analyze the API workflow. In the goal models, time is an important factor (as shown by softgoals such as "fast documentation" and "fast approval"). We believe that the ability to tell at a glance where the time is spent would assist in the development of an improved workflow, by making it easier to find the bottlenecks. Rather than (or in addition to) using a UML Activity Diagram, which has the primary purpose of showing in which order activities happen, an alternative process diagram could be used to allow for activity duration to be showed. It could also be displayed via annotation over the model, as a way of getting around the limited modelling language grammar. In addition to time, the models do also not display other quantitative aspects. The goal models show in which direction a task influences a softgoal, but not by how much.

VII. Threats to Validity

A. Internal Threats to Validity

Most of the company-specific data used to create the models were collected from one person at the company, which implies that there could be some bias. However, at the very first workshop, where most of the challenges in the API ecosystem design were discussed, several people from the company were present. After that point, the feedback on the models has come mostly from our contact person at the case company.

Even so, our main focus in this study was to explore how two kinds of models can be used together as complements, and how they can be improved. We believe this is unaffected by the potential bias in the company-specific data. The research process and the resulting models have been reviewed in each iteration by other researchers in the project group. The implications of the models (individually and as complements) have been discussed openly among the researchers (students and senior researchers).

B. External Threats to Validity

There are some threats in the validity of study in order to come up with concrete and generalized conclusion to make a study workable to other company. One of the most and crucial validity threats in the study is data originates from a single source, there was no any additional data source for using a triangulation methodology to see how this study workable in similar situation in other company. One senior research and a student made a face-to-face interview with a company representative, who is profile specification specialist. The interview did not include profile developers to find other relevant information to this thesis.

C. Construct Validity

In order to find out how well goal and process models can be used together to analyze an API ecosystem design, we evaluated the models in terms of expressiveness and effectiveness. We believe that the analysis of these qualities are able to capture some of the traits which make the models suitable to use as analytic tools for an API ecosystem design purpose. However, expressiveness and effectiveness does not capture everything. For example, while the ability to easily capture relevant information in a model per definition means it is effective [4], it does not capture how well they can be understood. As more details are added to the model, the more expressive it becomes while also becoming more complicated to understand.

In addition, effectiveness in particular is used as a subjective measurement in this study. No form of quantifiable information was used to justify the ease of which an aspect in the API ecosystem was modelled. It is therefore possible that another researcher could formulate a different answer to the question of effectiveness of the models (and by extension how

well they can be used to capture aspects of an API ecosystem) based on the same data.

VIII. Conclusion

In this case study, we tested the application of goal and process models as analytic tools to discover and mitigate the challenges and bottlenecks associated with the change in the API ecosystem design at the investigated company. We investigated whether goal and process models can be used individually or as complements to effectively capture the dynamic and static aspects of an API ecosystem using two snapshots (the as-is and to-be models). The goal models were used to analyze the static aspects of the API ecosystem, while process models were used to analyze the activities of the case company's profile development workflow.

From our evaluation, the goal models worked especially well as communication tools for the redesign of an API ecosystem. Due to the expressiveness of the i* framework, they exposed challenges in the form of unfulfilled softgoals and enabled the viewer to understand how different actors depend on each other. They provided an opportunity to develop a goal-oriented ecosystem design by working as a base for discussion of how to satisfy denied softgoals. Sorting the actors into layers gave a structured overview of which role each actor has in the API ecosystem. On the other hand, the models quickly became more complex and difficult to understand as they were becoming more complete, and eliciting information about how task affect softgoals is a challenge.

The activity diagrams were mainly used at the beginning of the study to allow the researchers to get familiarized with the case company's workflow, and the company showed a greater interest in the goal models. The process models are believed by the researchers to have worked better as analytic tools to discover bottlenecks in the API ecosystem workflow if action duration (i. e. the time it takes for an action to be completed) would have been modelled. Limiting the amount of time spent on different areas of the API development process was an important goal of the API ecosystem change, and we believe that quantitative information in the models would help to discover and address more challenges and bottlenecks.

IX. Future Work

For future research, we propose to further explore the possibility of adding quantitative measures to the fulfillment of goals. We would like to see a way to analyze the degree of satisfaction of a softgoal without having to rely as much on human judgment.

There is also a potential of experimenting more with the placement of actors in the different layers. If we were to create additional goal models from different perspectives, this would potentially affect the outcome of the analysis, and enable us to make more discoveries in terms of positive effects and/or new challenges of implemented changes.

The boundary objects between layers, identified by Hammouda et al. [2], were not considered in the scope of this study. The artefacts (use cases, API specification, and API model) affect adjacent layers when they change, and the research has found that most problems in API management "come from unilateral change of the boundary objects. It is therefore important that these artifacts are known by the company in order to use them in API strategy. We propose that the boundary objects are defined and analyzed in future work.

We would also like to see a similar approach using other modelling languages to model an API ecosystem design and workflow. A sequence diagram allows for duration constraints, and there are also other process diagrams which offers a simple and grammatically correct way to denote time. The development of such a model could make the process model more valuable as an analytic tool for API ecosystem design.

Acknowledgment

We would like to thank the senior researchers Imed Hammouda, Jennifer Horkoff, and Juho Lindman for their guidance and support in completing this study.

Finally, we express our profound gratitude to our partners for their continuous encouragement and countless support throughout the period of writing this thesis.

References

[1] Software Center. http://www.software-center.se/research-themes/technology-themes/customer-data-ecosystem-driven-development.

[2] Imed Hammouda, Jennifer Horkoff, Eric Knauss, and Juho Lindman. Emerging perspectives to api strategy (in submission). 2017.

[3] Jennifer Horkoff, Tong Li, Feng-Lin Li, Mattia Salnitri, Evellin Cardoso, Paolo Giorgini, and John Mulopoulus. Using goal models downstream: A systematic roadmap and literature review. 2015.

[4] Feng-Lin Li Tong Li Jennifer Horkoff, Fatma Basak Aydemir and John Mylopoulus. Evaluating modeling languages: An exampleffrom the requirements domain. *Lecture Notes in Computer Science*, (8824):260–274, 2014.

[5] Rik Eshuis and Roel Wieringa. A formal semantics for uml activity diagrams - formalising workflow models.

[6] Thorsten Berger, Rolf-Helge Pfeiffer, Reinhard Tartler, Steffen Dienst, Krzysztof Czarnecki, Andrzej Wasowski, and Steven She. Variability mechanisms in software ecosystems. *Information and Software Technology*, 56(11):1520–1535, 2014.

[7] Konstantinos Manikas and Klaus Marius Hansen. Software ecosystems–a systematic literature review. *Journal of Systems and Software*, 86(5):1294–1306, 2013.

[8] John Mylopoulus. Information modeling in the time of the revolution. *Information Systems*, (23(3-4)), 1998.

[9] Eric S. K. Yu. Towards modelling and reasoning support for early-phase requirements engineering.

[10] Jose Luis de la Vara, Juan Snchez, and Oscar Pastor. On the use of goal models and business process models for elicitation of system requirements. 2013.

[11] Eric Yu Jennifer Horkoff and Arup Ghose. Interactive goal model analysis applied–systematic procedures versus ad hoc analysis. In *IFIP Working Conference on The Practice of Enterprise Modeling*, pages 130–144. Springer, 2010.

[12] Imed Hammouda, Eric Knauss, and Leonardo Costantini. Continuous api design for software ecosystems. In *Rapid Continuous Software Engineering (RCoSE), 2015 IEEE/ACM 2nd International Workshop on*, pages 30–33. IEEE, 2015.

[13] Eric Knauss, Daniela Damian, Alessia Knauss, and Arber Boricia. Openness and requirements: Opportunities and tradeoffs in software ecosystems. 2014.

[14] Axel van Lamsweerde. Goal-oriented requirements engineering: A guided tour. 2001.

[15] Eric Yu and Stephanie Deng. Understanding software ecosystems: A strategic modeling approach. 2011.

[16] S. Jansen, A. Finkelstein, and S. Brinkkemper. A sense of community: A research agenda for software ecosystems. *ICSE Companion 2009*, pages 187–190, 2009.

[17] Jan Bosch. From software product lines to software ecosystems. In *Proceedings of the 13th international software product line conference*, pages 111–119. Carnegie Mellon University, 2009.

[18] K. Popp and R. Meyer. Profit from software ecosystems. 2010.

[19] K. Popp. Definition of supplier relationships in software. *http://www.drkarlpopp.com/resources/ICSOBSubmission2.pdf*.

[20] Eric S. K. Yu and John Mylopoulus.

[21] Masha H. Sadi and Eric Yu. Modeling and analyzing openness trade-offs in software platforms: A goal-oriented approach. 2017.

[22] Creative Leaf. http://creativeleaf.city.ac.uk.

[23] OpenOME. https://se.cs.toronto.edu/trac/ome/wiki.

APPENDIX

A. Goal Models: Actors

1) Level 4: Domain

- Mechatronic device: A device produced by the company to perform a specific task.
- End User: Refers to users of the product (mechatronic device, MD system).
- App Developer: A to develop mobile application used for operating the device/system.
- Profile Specialist: A person who are develops the device profile.
- API Consumer: System which utilises the API.
- Profile Owner: A person who verifies a profile.
- Communication Device Stakeholder: A person developing special profile for the control unit (controller profile).

2) Level 3: App SW

- Mobile App: A mobile phone application used to access the device/system data.
- Controller: A device used to control the mechatronic devices in a system.

3) Level 2: API

- Communication Protocol: A static communication protocol used to convey data to between devices.
- Profile: This refers to the set of detailed device data.

4) Level 1: Business Asset

- Mechatronic Device (MD) Data: Refers to data generated when the device is in operation.
- MD System: Refers to a system of mechatronic units, controlled by a controller.
- Database: Refers to an electronic filing system.
- Feature Model: A model that capture the systems relevant for a stakeholder of a product line.