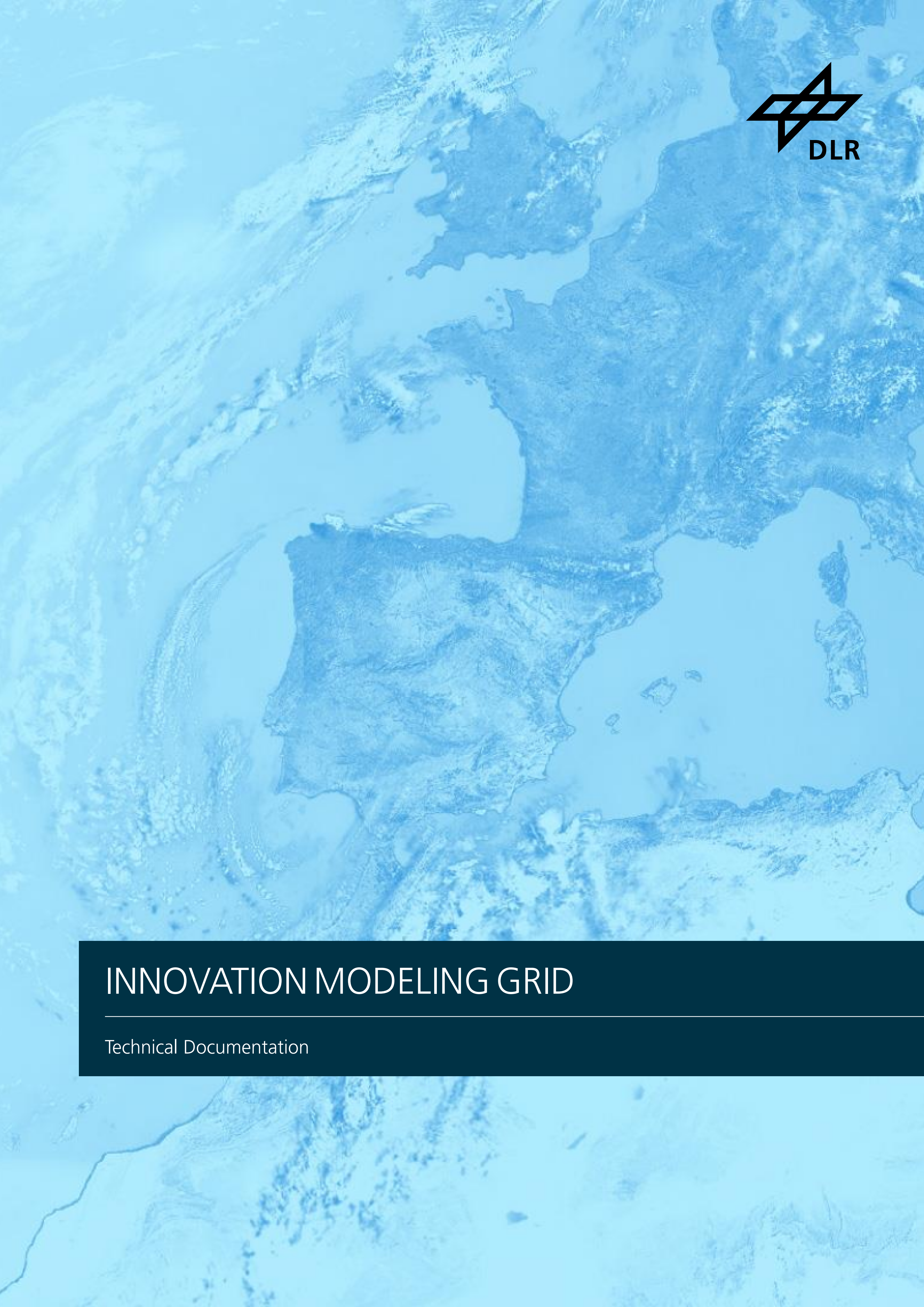


INNOVATION MODELING GRID

Technical Documentation

DLR German Aerospace Center
Institute of Systems Engineering for Future Mobility

Prof. Dr.-Ing. Axel Hahn
Escherweg 2
26121 Oldenburg
Germany
Tel: +49 441 770507-100
Fax: +49 441 770507-102
Web: https://www.dlr.de/se/en/

Oliver Klemp
Tel: +49 441 770507-331
Mail: oliver.klemp@dlr.de

Document Identification:

Title	Innovation Modeling Grid
Author(s)	Oliver Klemp
Filename	IMoG.tex
Last saved on	29th September 2023

Abstract

This technical document presents the committee driven innovation modeling methodology "Innovation Modeling Grid" in detail. This document is the successor of three publications on IMoG [6, 11, 17] and focuses on presenting all details of the methodology.

Acknowledgments

This work has been supported by the GENIAL! project as funded by the German Federal Ministry of Education and Research (BMBF) under the funding code 16ES0865-16ES0876 in the ICT 2020 funding programme.

Contents

Part I.

Overview over IMoG

1. Introduction

This document presents the modeling methodology Innovation Modeling Grid. The Innovation Modeling Grid (IMoG) targets the discussion and modeling of innovations in a committee. The methodology shall reduce the start-up time for innovation modeling by pre-structuring the innovation in the sense of advising what type of elements exist and how they relate to each other. The modeling methodology originates from a project within the context of the automobile industry, which is used here to motivate the methodology.

The automobile industry is undergoing a major transformation and is facing the following situation: First, there is the huge demand for autonomous and highly automated driving. Autonomous driving shall provide a safer and more efficient transportation, while allowing passengers to focus on other things. If the driver wants to enjoy driving, then highly automated systems shall support the driver with several assistance systems to ensure a safe journey. This demand is on a different complexity level than the typical innovations known in the automotive industry and will shape the future development.

Secondly, there is also the huge demand of more sustainability due to the climate change. The electrification of the transport sector and the limited amount of rare resources require new technologies and design principles.

Individualization is another demand. The passenger demands more comfort and custom functionality in vehicles. Individualization requires a rethinking towards data driven and software defined vehicles. Software defined vehicles also relate to high complexity and high loads of external communication. Additionally, data driven in-vehicle applications represent potential for new business models for software companies.

Referencing business models, mobility as a service is an uprising trend as a business model for car manufacturers. Not only conventional vehicles are considered, but also the whole transportation sector including trains, the aerospace and the last mile. This trend requires a rethinking of the structure of the automotive industry as a whole. The question raises of what is holding the automotive industry back of just addressing these demands today? Well, each of the demands represent a special challenge.

Autonomous driving / highly automated driving Source: DLR (CC BY-NC-ND 3.0)

The autonomous driving functionality relies on an accurate perception of the environment, an accurate localization of the car's position, an accurate prediction of the behavior of the other traffic participants – optionally by sharing the intends by communication with the other participants or with the infrastructure –, a sophisticated control of its own behavior, including a computation of trajectories, monitoring its own behavior and securing the vehicle from unauthorized manipulation and the driving functionality relies on an accurate planning and navigating of routes from point A to point B. This functionality poses a major challenge to the industry with a high level of complexity, including high safety and security requirements. The computation of this functionality is expected to be mainly implemented in software. To stay competitive the automotive value chain needs to adjust to this new software focus.

Electrification / sustainability Source: DLR (CC BY 3.0)

The climate change and the implied departure from combustion engines to electric engines is a another challenge. Combustion engines are decried as not acceptable anymore as a future transport solution for the masses. The expected technology shift goes towards the electrification of cars, which represents a far easier technology than highly optimized combustion engines for market newcomers. On the other hand, the electrification requires a lot of rare resources - like Lithium – and requires huge accumulators. Up-scaling the power net in the vehicle itself proposes a challenge of its own by considering electromagnetic conductivity, cable weight and so on. These factors increase the market pressure for all members of the automotive value chain.

Individualization / software defined vehicles Source: DLR (CC BY 3.0)

The next challenge is proposed by the individualization of software defined vehicles [4, 16, 7]. The deciding factor for autonomous vehicles lies in the software centered complexity as mentioned in the autonomous challenge. Individualization requires on top of that short market times. For example a user likes to connect the newest smartphone generation with the vehicle. These vehicles can not be simply called into the next workshop to update the software, so over the air updates are required. These demands further amplify the need for a software focus. The value chain has to face this software focus, the implied complexity and furthermore the implied safety and security requirements.

Mobility as a service
Source: DLR (CC BY-NC-ND 3.0)

Finally, the general trend of car manufacturers is their move towards new business models that focus on mobility as a service. Cause for the shift is the globalization and climate change, which demand a rethinking of mobility. Cities get more and more interconnected with various optimized solutions for mobility: from (underground and intercity) trains, to various models of bus systems, over motorcycles, e-scooters and e-bikes for short trips. Owning cars is thus not mandatory in larger citizens anymore and therefore, a lower demand for owning vehicles can be expected. Additionally, improvements in the drive train of new cars tend to be insignificant from the view of the end user equalizing the quality of cars in the market and decreasing the importance of the brand of the car. The car manufacturers have to think about how they can bind their current customers if they do not want to loose market stakes. The general trend towards retaining current customers lies in adapting to their new demands. Future business models are expected to be focused around the aspect of mobility as a service. This restructuring poses a major challenge to the whole value chain.

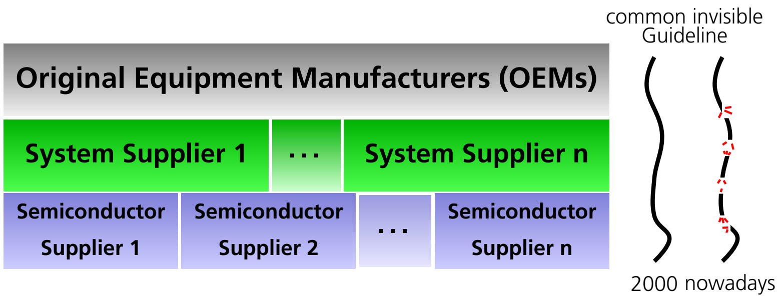
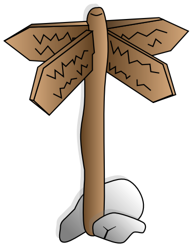


Figure 1.1.: **A broad view on the structure of the current automotive value chain. Its understanding of the common future is not as easy as in the last decades. Thus their common invisible guideline can not be simply assumed to exist anymore. Instead the automotive value chain has to collaborate and explicitly design it to achieve their maximum efficiency.**

There is another challenge that is not induced by customer demands. That is the current structure of the value chain (see Figure 1.1). The long established automotive value chain between the Original Equipment Manufacturers (OEMs), chip manufacturers (Tier 1) and semiconductor suppliers (Tier 2) is very fragmented and optimized for producing vehicles with long product- and lifecycles [13]. This structure works well for modular design with hardware elements that have a long cycle times. However, following this principle of modular design leads to a sequential working process, resulting in long communication times and slow innovation speed with no effective use of horizontal connections between the suppliers. Additionally, the structure does not address the new complexity, the software focus and a service oriented business model in a suitable manner. The demanded fast and safe realizations represent an enormous technological and methodical challenge. On the one side, the car manufacturer has to anticipate the very rapidly changing possibilities of future microelectronic platforms, sensors and semiconductor technologies already at the time of product definition to include them in the next generation. On the other side, the suppliers have to know early enough the requirements of future functionality to strategically invest into technology developments on a quantitative and reliable basis. The missing communication between the value chain makes it hard to understand and predict the future and slows the value chain significantly down.



Uncertainty

These challenges introduce together a lot of uncertainty - a well known challenge in requirements engineering: The complexity, including safety and security, adds uncertainty due to the vast exploration space, as not every solution can be explored. Additionally, this complexity makes it hard to predict the future market trends and new directions of technology. The limited resources, competition and time to market challenges add uncertainty in the sense of limited time with the pressure to find good solutions in time. The new business models and new demands challenge the partners of the value chain by being uncertain on how they will cooperate together in the future. Also directly related is the fact that the value chain has been indirectly guided by a common understanding of future technology over the past decade. With the raising complexity and uncertainty this invisible guideline more and more disappears. Overall, the uncertainty and missing of knowledge makes it hard to predict the future, plan innovations and do the right investment decisions. The pressuring question is therefore how the value chain can sustain the new business models, autonomisation, electrification and individualization with their high requirements.

(Storyboard figures are owned property of http://storyboardthat.com)

One way to cope with these challenges is by trying to boost innovation with a public roadmapping approach. This road mapping approach focuses on shaping the understanding of the innovation, which is in essence a requirements engineering problem. The goal of the roadmap is to better understand and communicate future innovations, the required future technologies and the decisions about their future directions of the other partners of the value chain. The expected gain of this synchronizing of the strategies across the value chain is an acceleration of the development of future innovative applications.

The approach can be understood as followed: The automotive value chain forms a committee for creating a public roadmap on a specific innovation (see Figure 1.2). This committee may include several car manufacturers (Original Equipment Manufacturers, also called

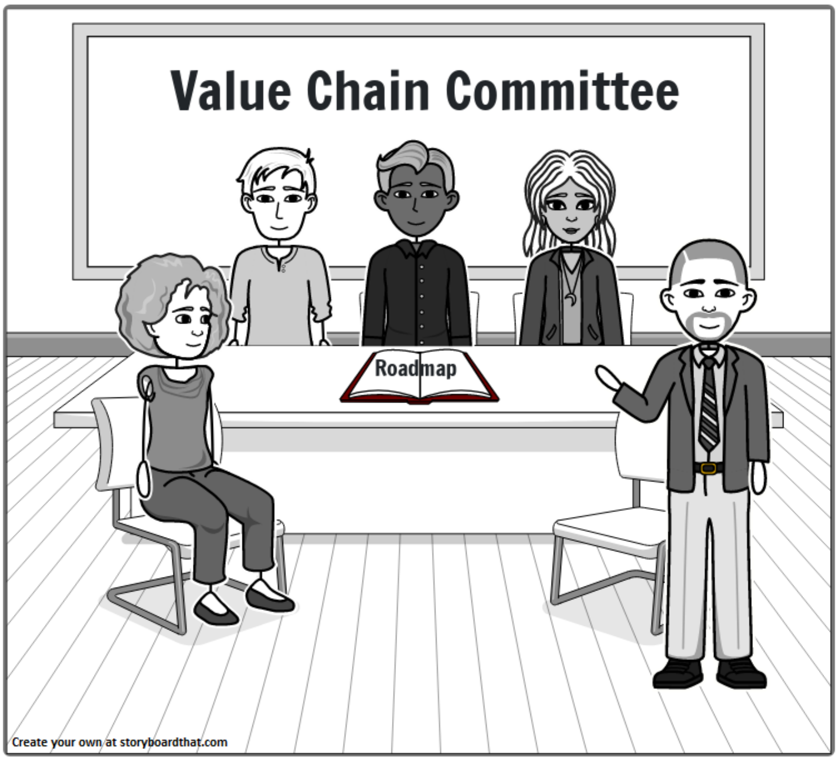


Figure 1.2.: **An example committee that aims to boost innovation with a public roadmapping approach.**

OEMs), several software and hardware component suppliers (Tier 1) as well as several semiconductor suppliers (Tier 2). The committee is open for the public and for new members to comply with the compliance laws. The committee meets and discusses the innovation by focusing on the strategies of the stakeholders, the features and functions of the innovation and by exploring the possible solutions of the innovation. It is crucial for the success to discuss on an appropriate level, which however varies from innovation to innovation. An appropriate level includes the understanding of the problems and its technical constraints, but it does not include too many details about the development of the innovation as the innovation in itself is not implemented by the committee.

The immediate question appears as for every approach: "How does a consistent public road maps based information transfer in the value chain tackle the challenges?". By understanding the future of the innovation and the value chain, the innovation becomes plannable, which reduces the uncertainty about the future for each partner and reduces the involved risks in investment decisions. The manufacturer can reliably plan with the discussed chip technology long before it is available. The suppliers gain an early insight into forthcoming requirements with the certainty that their newly developed chip technologies and components will suit an existing demand. The exploration of solutions directly helps with handling the complexity and boosting quality management. The competition aspect is covered by using the roadmap to prepare early for the future innovation. The discussion of the strategies helps with adjusting the value chain to a software defined focus and addressing the new business models. Overall the public roadmap enables to adjust the value chain to the new demands. This holistic road mapping is new to the value chain and requires to be open to the public to play within the rules of compliance. It also enables to communicate in a horizontal manner between suppliers, that boosts synergies in the innovations design.

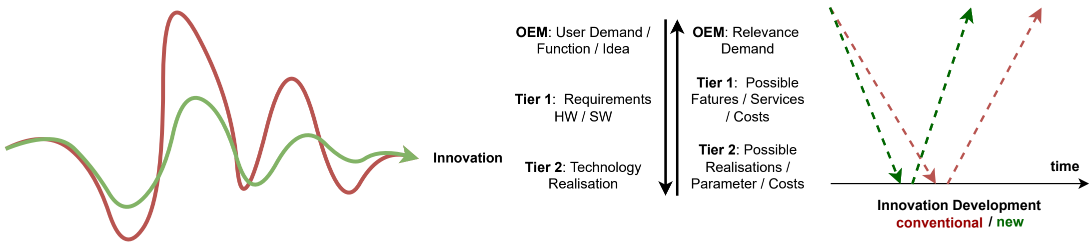


Figure 1.3.: **Expected gain of the approach.**

One major gain is the expected speed up in the innovation cycles that is crucial to meet the short time to market demand known from software development. This speed up can be imagined as followed (see Figure 1.3): A company without a roadmap explores an innovation by starting with a seemingly feasible direction. They may spend some time exploring, explaining and discussing with other suppliers and adjusting the direction as needed. They may find a detail that is not satisfiable by the chosen direction and thus try out the next feasible direction. They proceed this way - with some more minor adjustments here and there - until the innovation is sufficiently explored. A committee with a roadmap may be faster by discarding unsatisfiable directions earlier, which smoothens the path taken. This speed is achieved for several reasons: First, by discussing and sharing expectations with the committee, technological possibilities are better understood. This directly leads to an earlier discarding of unsatisfiable directions. Secondary, the value chain can parallelize the development. This parallelization is achieved through the already mentioned reduction of uncertainties and risks, which leads to the confidence to initiate investments at an early stage. Lastly, the committee has the opportunity to standardize common terms to significantly reduce the communication overhead via misunderstandings across the whole value chain. Finally, a short note about the limitations of this road mapping approach. The development and engineering tasks are unaffected by this approach. That means that the development still has to cope with the increased complexity and the technologies have to be developed as well.

Given this road mapping approach, we investigated the research question of what an appropriate process, methodology and tool for this approach would be. In our opinion a dedicated methodology supported by a process and a tailored tooling is required to efficiently handle this specific context. We developed the Innovation Modeling Grid (also called IMoG) [6, 10] as a methodology with a process and a tooling prototype to enable an open, fair and compliant communication along the value chain. The methodology's goal is to efficiently represent and model early microelectronic innovations to enable a consistent information transfer along the value chain. The process recommends who is doing what with which tool to produce which artifacts and the tooling supports as good as possible the above mentioned process and methodology. The process, the methodology and the tooling are the main focus of this document, which are described in detail in the further sections. The document will finish with the preliminary evaluation results and a closing.

2. Delimitation of IMoG to relevant thematic fields

IMoG relates to the umbrella term of innovation management, IMoG shares similarities with the general roadmapping approach and IMoG shares similarities with other well known fields like requirements engineering and systems engineering.

Innovation management refers to the systematic approach of planning, controlling and executing activities related to innovations. It includes the generation of ideas, evaluation of the feasibility of the ideas, managing development prototypes and guiding the whole process up to the product. Innovation management is used in companies to drive growth, competitiveness and sustainability. Furthermore, innovation management also refers to the management of innovation outside of companies. It provides means to communicate with the stakeholders of the corporation and supports the synchronization and harmonization of agreements between the corporation and external business units.

IMoG can be considered an innovation management technique, although it does not specifically target any particular company or emphasize decision-making activities. Thus, IMoG is only applicable for the specific class of committee applications in innovation management, where the focus does not lie on presenting the decisions of a company to their stakeholder.

Roadmapping is a technique used in innovation management that relates to IMoG. Roadmapping is a creative analysis procedure used to analyse, forecast and visualize the development paths of products, services and technologies [**?**]. Roadmapping is widely recognized as a strategic management tool to forecast the future development. Roadmaps serve various purposes depending on the involved stakeholders. They support achieving a robust and market-oriented technological positioning as well as for enhancing, protecting an utilizing the competence of the organization [**?**]. Furthermore, roadmaps play a crucial role in providing orientation for employees, for external stakeholder and for marketing when published. IMoG shares many similarities with roadmapping, however, IMoG's context slightly differs from the typical roadmapping context. The typical roadmapping approach requires perfect or quite complete knowledge about the scope of the roadmap, topics of interest and future directions of the object under consideration. However, this knowledge is often not given in a microelectronic value chain with a huge number of different stakeholder and varying expertise. Therefore, the IMoG methodology does not build on the assumption of perfect knowledge and includes a sophisticated investigation of the problem space before investigating the future possibilities. Furthermore, we assume, that the topic under investigation is quite complex and that the solution space is not yet fully understood by the committee. To tackle this complexity, the IMoG methodology recommends a model-based supported investigation of possible future solutions with dedicated tools. To better understand the solution space and decision making, the IMoG methodology recommends to parameterize and decompose the solutions until they are sufficiently understood. This parameterization and decomposition requires dedicated tools to handle the complexity. Nonetheless, many typical roadmapping techniques can be applied at the later states of the IMoG methodology when the problem is better understood and when the possible solutions are collected. The difference in imperfect knowledge and complexity also requires to handle the workshops with the committee differently to the typical roadmapping workshops. This document gives in the later chapters recommendations on how the IMoG methodology may be applied in these workshops and how the roadmap for the microelectronic value chain as a whole can be addressed.

IMoG also shares similarities with requirements engineering. IMoG divides the innovation modeling into the problem and solution space, which is a common approach in requirements engineering. Furthermore the alignment and understanding of the innovation is a crucial part of IMoG, which relates to the goal of requirements engineering to foster a better understanding between all stakeholder. IMoG distinguishes itself from requirements engineering by focusing on the class of innovation modeling in committees while requirements engineering covers the more general and abstract guidelines for stakeholder and system investigation.

Systems engineering focuses on how a system can be systematically developed and systems engineering does not specifically consider committees or abstract concepts. IMoG also investigates the system decomposition and shares similar concepts. However, IMoG does not require the level of detail known from systems engineering models, because the innovation is not developed by the committee members. The development of the innovation happens after the public committee phase internally in the corporations.

3. Process for IMoG

This chapter covers the process that is recommended for the committee to create a roadmap for their innovation. This section presents the recommended process for the committee to create an innovation roadmap, referred to as IMoG's process. The description of IMoG's process commences by introducing the various roles involved in Section 3.1. Subsequently, it outlines the process activities, the produced artifacts, and the tools involved in 3.2. Notably, IMoG's process does not propose any template for milestones. The decision to exclude such a template is based on the assumption that it would vary significantly for each specific innovation. The process description is finally illustrated with a "Mobility with an e-scooter" innovation from the time before e-scooters got popular in cities in Section 3.3.

3.1. Roles and Responsibility

The roles of the members of the committee are presented first. IMoG defines three disjunctive sets of roles. Each member of the committee may take zero, one or more roles from each role set. This implies that each member of the committee may represent several roles and that their roles may differ depending on the task.

The first set of roles defines roles of the corporation each member may represent. The corporation roles include the role of the OEM (Original Equipment Manufacturer), the role of the Tier 1 supplier and the role of the Tier 2 supplier. The three roles are defined as follows (inspired by Knauf [12]):

- The **OEM** (Original Equipment Manufacturer) is the manufacturer of the end product, which deals with the market launch of the vehicle.
- The **Tier 1** suppliers develop system solutions that are tailored to the end product without major changes.

- The **Tier 2** creates the components needed to be integrated into systems. This includes the production of semiconductors and microcontrollers.

The corporation roles of the automotive value chain are often more differentiated than in OEM, Tier 1 and Tier 2. However, the roles defined were evaluated as sufficient enough for automotive committees discussing microelectronic innovations.

The second set of roles defines the roles of the members of the committee. The roles are described in Table 3.1. The third set of roles are the roles of the corporation employees, which specialize the role of the "Corporation Representative" to execute the specific activities of IMoG. These (in-house) employees help the committee by providing and compiling information. These roles are described in Table 3.2.

Table 3.1.: **The involved roles in the automotive value chain committee**

Roles	Description
Committee Leader	The responsible person leading the roadmap committee.
Corporation Representative	The responsible person of a corporation to coordinate the corporation internal tasks to produce the needed inputs for the roadmap.
IMoG responsible Model Expert	The responsible person of creating and maintaining the IMoG model on the command of the committee members. The IMoG responsible Model Expert is also called IMoG Modeler.
Roadmap Manager of the Committee	The roadmap manager of the committee is responsible for the creation and maintenance of the roadmap.

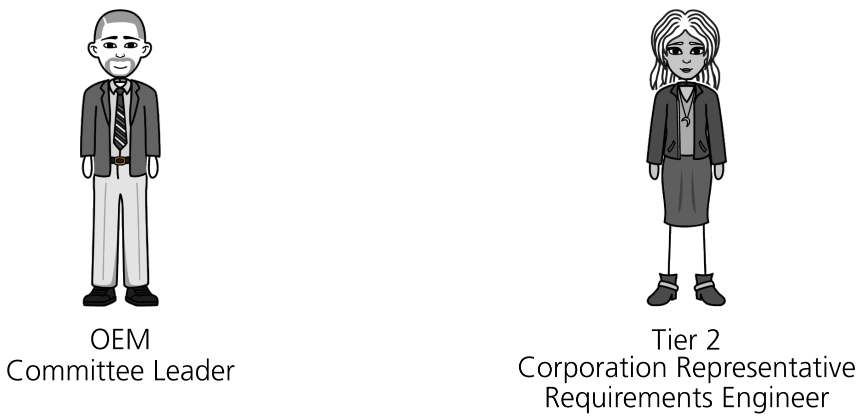
Examples

Two examples for a set of chosen roles are shown in the following: A corporation member of an Original Equipment Manufacturer (see Figure 3.1a) has an idea for a new innovation he likes to discuss. He founds a committee for discussing the new innovation and takes the role of the committee leader. He thus have two roles assigned: The role of an OEM representative and the role of the committee leader.

The committee leader invites a member of a Tier 2 supplier to join the committee see Figure 3.1b). She decides to represent her corporation and play an active role in the committee. She additionally brings her expertise as a roadmap manager. She thus have three roles assigned: The role of a Tier 2 representative, the role of the corporation representative for the corporation she works for and the role of the roadmap manager of her corporation.

Table 3.2.: **The involved roles executing the required activities of the recommended process for IMoG**

Roles	Description
Roadmap Manager	The roadmap manager monitors the innovation status, reports to top management on the feasibility of the innovation, surveys new technologies from other partners, and updates the roadmap. The roadmap manager investigates trends and innovations. During innovation modeling, the roadmap manager performs the initial tasks and writes the roadmap after consulting with the other domain experts, requirements engineers, and system architects.
Requirements Engineer	The requirements engineer creates initial top-level requirements for the innovation and captures them uniformly (formally or in natural language). The requirements engineer leverages the expertise of the domain experts and system architects to uniformly refine the requirements in the system models.
System Architect	The system architect has the role of an interdisciplinary expert who designs systems by using modeling techniques. The system architect has know-how in the area of software-hardware design. In innovation modeling, the system architect takes on the role of the innovation modeler and its decomposition into subsystems.
Domain Expert	The domain expert represents a specialist of a particular discipline covering subdomains of development. The domain expert supports the innovation modeling and evaluates its influences and dependencies of certain domain elements on other domain elements.



(a) **The committee leader.** (b) **The invited Tier 2 representative.**

Figure 3.1.: **Two role examples. Figures by StoryboardThat (©), `www.storyboardthat.com`, used by permission.**

3.2. Process parts: The Activities, Artifacts and Tools

This section presents the recommended activities, the target artifacts and the recommended tools of IMoG's process. The section starts with the overview in Section 3.2.1 of the activities, the produced artifacts and the involved tools. Based on this overview the process details are described from the side of the activities in Section 3.2.2.

In chapter 4, the IMoG methodology is introduced. The methodology shows *what* is captured in the roadmap model, in *which* way these elements relate and *how* details shall be processed. However, the process description encompasses the artifacts, which represent the results of IMoG's methodology. Because of this dependency, it is recommended to read the methodology chapter first before reading the artifact description. Similarly, it is recommended to read the description about the involved tools after the artifacts.

3.2.1. An abstract overview over the activities, artifacts and tools

Activities

IMoG recommends seven activities for modeling the innovation. Every of these activities is processed by people taking the recommended roles of IMoG's process, which (the roles) were presented in Section 3.1. The mapping of which activity is processed by which roles is presented in Figure 3.2. Note that the roles are now depicted as colored stick figures. Despise their graphical depiction their meaning remains the same as before. The activities are described in the following.

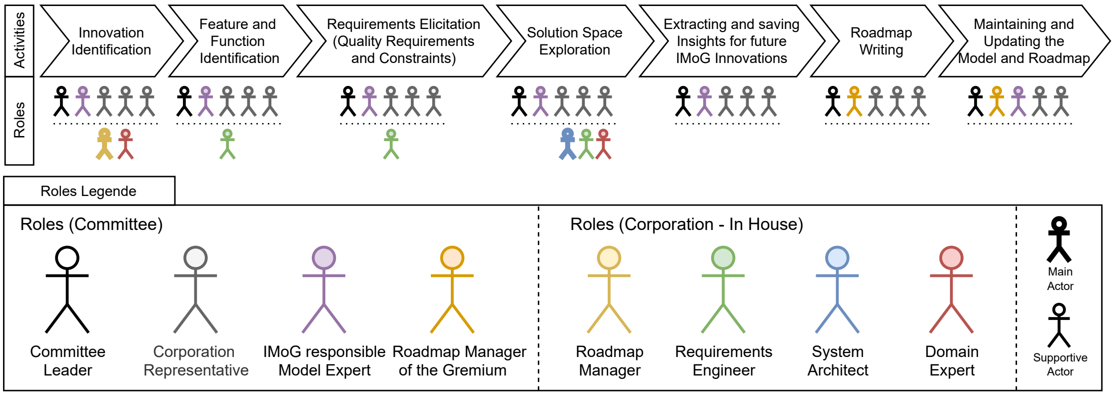


Figure 3.2.: **Activities (arrows) and roles of the working process. The space between the activities represents nothing special and is for the sake of the graphical representation only. The roles are described in Section 3.1.**

The first activity is called **Innovation Identification**. The innovation identification includes creative methods as well as market segment analysis to develop a new innovation idea and create an initial description of the innovation. The involved roles include the committee leader, the IMoG modeler and the corporation representatives. The committee leader sets up and coordinates the meetings. The IMoG modeler is responsible for creating the models and the corporation representatives are responsible for proposing their interests. The in-house roles include the roadmap manager and the domain experts that help the representatives to identify and describe their interests.

The second activity is called **Feature and Function Identification**. The purpose of this activity is to refine the problem understanding and create a feature hierarchy based on the description of the innovation. As a further refinement, the feature hierarchy may include user stories and use cases. The involved roles include the same committee members of the innovation identification activity: the committee leader, the IMoG modeler and the corporation representatives. Requirements engineers of the corporations support the creation of the feature hierarchy.

The third activity is called **Requirements Elicitation**, which adds quality requirements and constraints to the feature hierarchy and refines the problem space further. It is the last activity focusing on the problem space. The roles that are involved in this activity are the same as in feature and function identification activity.

The solution space of the innovation is examined after the problem is sufficiently understood. The corresponding activity is called **Solution Space Exploration**. It consists of modeling the possible solutions of the innovation with (sufficient) technical details. The involved committee roles include the committee leader, the IMoG modeler and the corporation representatives. The corporation internal leader of this activity is the system architect to examine and analyze the possible solutions. The system architect gets support from the requirements engineer and the domain expert, however, their help is of supportive nature.

After the solutions are examined, the committee extracts the insights gained by the generated model and saves them in their database for further innovations. This activity is called **Extraction and Saving of the Insights**. No in-house corporation roles are needed.

The **roadmap writing** is the next activity building upon the insights from the last activity. The committee members meets again to discuss the roadmap together. The modeling activities are finished and thus the IMoG modeler does not take part in this activity. The roadmap manager takes responsibility for the roadmap writing, structures the document, and assigns tasks. After this activity, the main roadmapping activies are done.

Based on this roadmap, reoccurring meetings are established to **maintain and update** the roadmap. The same roles are involved as in the writing of the roadmap.

It is not required to complete each of the seven activities before starting the next one (as usual). Instead, it is sufficient to draft each model of each activity and refine them when necessary, similarly to what was proposed with the twin peaks model [14].

Artifacts

(It is recommended to read Chapter 4 before this Section.)

The artifacts are also added to the process image in Figure 3.3. The idea description and the filled Strategy Perspective constitute the artifacts of the "Innovation Identification" activity. The details of the perspective are presented in Chapter 4. The artifacts of the "Feature and Function Identification" activity are the user stories, use cases and the filled Functional Perspective. The filled Quality Perspective constitutes the artifact of the "Requirements Elicitation (Quality Requirements and Constraints)" activity. The filled Structural Perspective constitutes the artifact of the "Solution Space Exploration" activity. The (Domain) Knowledge Perspective and the list of insights constitute the artifacts of the "Extracting and saving Insights for future IMoG Innovations" activity. Finally, the roadmap is the artifact of the "roadmap writing" activity, which is updated within the "maintaining and updating the model and the roadmap" activity. The artifacts are illustrated with the presentation of the perspectives in Chapter 4

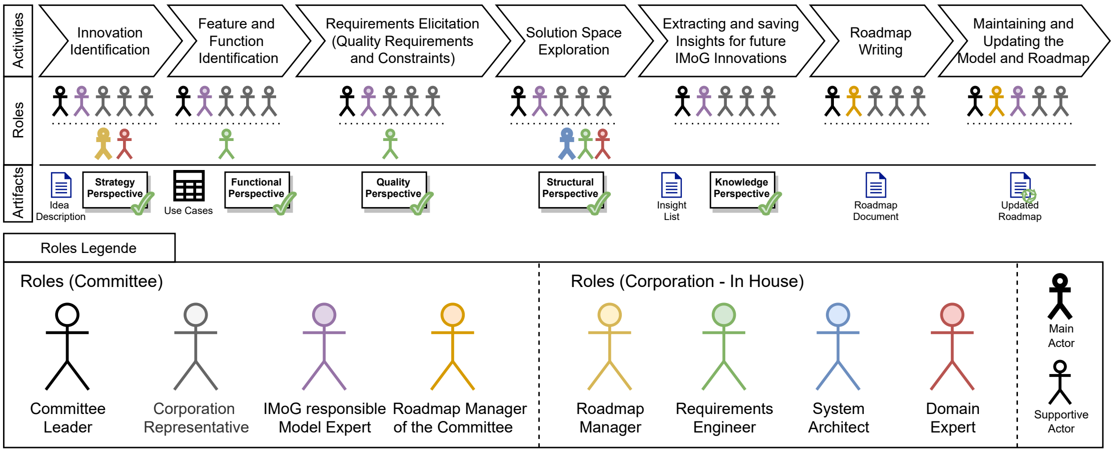


Figure 3.3.: **IMoG process with artifacts appended**

Tools

(It is recommended to read Chapter 4 before this Section.)

The proposed tools are added to the process image in Figure 3.4. Overall, we think that a dedicated tooling for IMoG is required and thus the recommended tooling for IMoG shown in the figure is such dedicated tooling. A tooling prototype for the Functional Perspective is already implemented and called "IMoG IRIS" prototype. Unfortunately, the resources for implementing were not enough to extend the dedicated prototype for the remaining activities. These left open implementations are marked in the figure with "To Be Done".

Next to a dedicated tooling, some activities are best supported by using extra tools: The "Innovation Identification" activity would be best supported by a creativity tool that the committee is well versed with. This, for example, may be a mind mapping tool, some whiteboards or something else. The "Feature and Function Identification" activity would be best supported with a dedicated tool to create and manage use cases and user stories. This could be a common text and table manipulation tool or a more sophisticated requirements engineering tool that supports user stories and use cases well. The "Extracting and saving Insights for future IMoG Innovations" activity would be best supported by a text editor to write the insights down. A text editor is helpful for the "Roadmap Writing" activity and the roadmap updating activity.

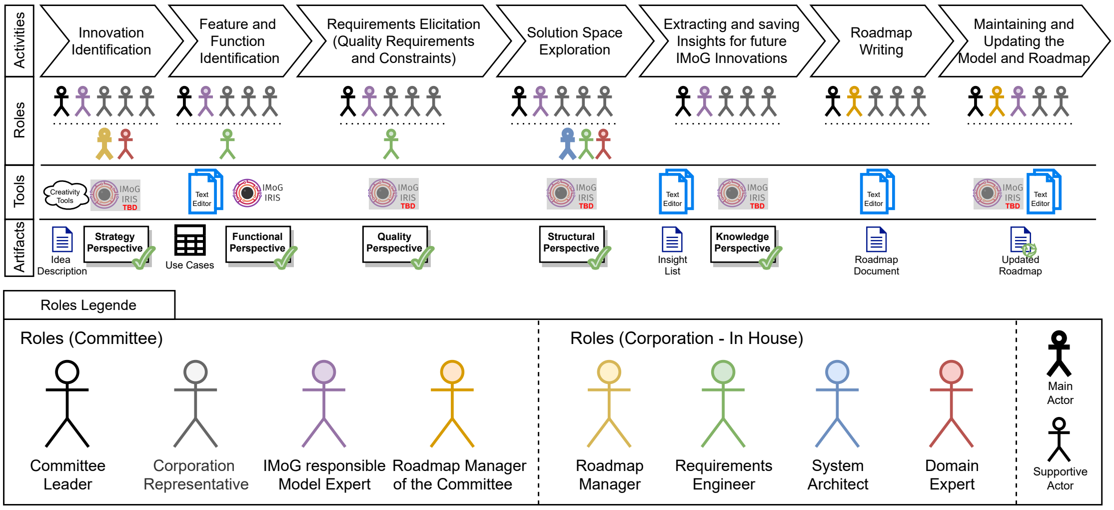
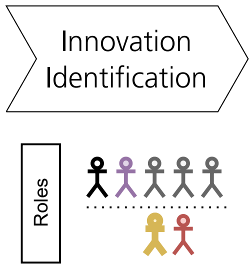


Figure 3.4.: **IMoG's process description appended with the proposed tooling (final representation).**

3.2.2. Detailed activities description

This section describes each activity in detail.

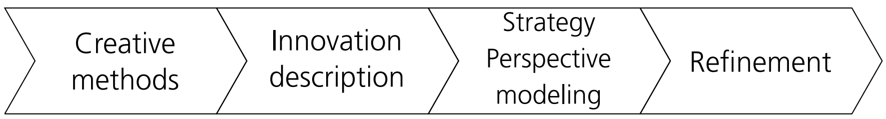
(Activity) **Innovation Identification**



(Roles) The involved roles include the committee leader to set up and manage the meetings, the IMoG modeler responsible for creating the models and the corporation representatives in the roles of the roadmap manager for proposing their interests in the innovations as well as some domain experts for supporting the roadmap managers.

(Short activity description) This activity uses creative methods as well as market segment analysis to develop a new innovation idea and create an initial description (see figure 3.5).

(Sub-activities)



(Detailed description) The committee leader invites the committee members with the above mentioned roles to a meeting to develop a new innovation idea. The committee decides on a creative method or decides on a market analysis technique and carries out the creative method. Which creative method or market analysis to choose is not defined nor restricted. Creative methods include for example "Brainstorming", "Mind maps", "Zwicky Boxes", "Walt Disney method", "Scenario projection", etc. Market analysis include for example "User needs projection", "User stories", "Time to market analysis" or "Business model analysis". The method that fits the committee members and their idea the best is the one to choose. Once the committee finished the creative method and closes the meeting, the committee leader writes down a description of the result of the creative method. Based on this description the IMoG responsible person translates this description into a draft of the Strategy Perspective (see Section 4.2.1). Now, the committee refines this model of the Strategy Perspective in a few more meetings or by assigning personal tasks. The refinement process can also include refinement and review processes in the corporations itself by ask internals (probably people with the roadmap manager role or domain expert role) to give their inputs. The input may include more information about the innovation, refined descriptions, goals or identify elements that shall be traced. This refinement process goes on until they are sufficiently satisfied with the result (Strategy Perspective).

(Artifacts) The innovation description (including the common vision and possibly some diagrams) and the filled Strategy Perspective (presenting the vision, the diagrams as well as the stakeholders interests, concerns and strategy and textual goals) constitute the artifacts of the “Innovation Identification” activity.

(Tools) Overall, we think that a dedicated tooling for IMoG is needed and thus the proposed tool for IMoG would be such dedicated tooling. We already implemented a tooling prototype for the Functional Perspective, called “IMoG IRIS” prototype. Unfortunately, we do not have enough resources to extend the dedicated prototype (IMoG IRIS) for the Strategy Perspective activities. Additionally, this activity is best supported by a creativity tool that the individual committee that they efficiently use already. This creativity tool may be any tool that supports the creative techniques and analysis (paper, mindmaps, documents, scratchboards, drawio, etc...).

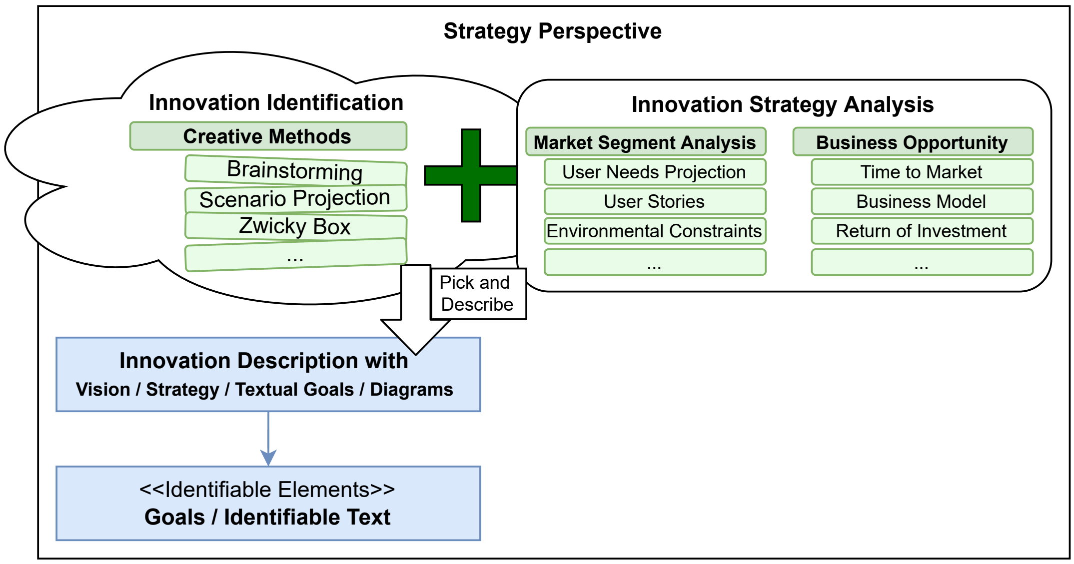
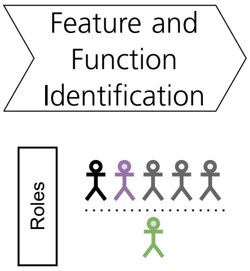


Figure 3.5.: **Possible methods for the Innovation Identification activity.**

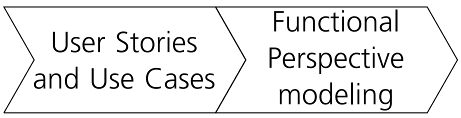
(Activity) **Feature and Function Identification**



(Roles) The involved roles include the committee leader, the IMoG modeler and the corporation representatives. In-house requirements engineers are also involved in this activity.

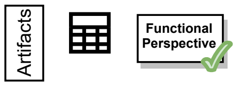
(Short activity description) The goal of this activity and the Functional Perspective is to refine the problem space and create a feature hierarchy including optional user stories and use cases based on the description of the innovation.

(Sub-activities)

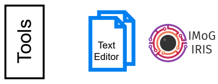


(Detailed description) After the identification of the innovation in the "Innovation Identification" activity, the next activity focuses on the identification of the features and functions needed to fulfill the innovation. The features and functions shall represent a refining of the problem space of the innovation. The committee leader invites the committee members with the above mentioned roles to a meeting. First they decide if they want to create user stories and use cases for understanding the general conditions of their innovation or if the general conditions of the innovation are sufficiently understood without user stories and use cases. If they decide to create user stories and use cases, they use the meeting to identify the user stories and use cases. The corporation representatives are responsible for giving and checking the input for the user stories and use cases. They may request their in-house requirements engineers for supporting this task. The IMoG modeler supports the corporation representatives by creating templates and giving advice for formulation. The committee leader moderates the meetings. The outcome of the meeting is a draft of these user stories and use cases. The members of the committee then distribute tasks to refine the user stories and use cases to a sufficient degree. They meet and refine again until they are sufficiently satisfied with the result.

(Detailed description continued) Then the committee leader invites the committee members for another meeting to create a draft of the feature model in the Functional Perspective. The IMoG modeler creates a draft of the Functional Perspective based on the inputs of the corporation representatives. The committee refines the model by giving input via in-house meetings of the corporations and by additional committee meetings. The in-house requirements engineers help the corporation representative and checks the validity and consistency of the Functional Perspective. This refinement process goes on until they are sufficiently happy with the result (Strategy Perspective).

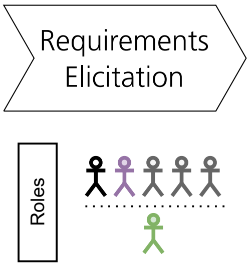


(Artifacts) The artifacts of the "Feature and Function Identification" activity are the features and functions in the form of the Functional Perspective and the optional user stories or use cases that are mapped on the features and functions. All features, functions, user stories, use cases as well as all other information are represented in the Functional Perspective.



(Tools) Overall, we think that a dedicated tooling for IMoG is needed and thus the proposed tool for IMoG would be such dedicated tooling. We implemented a tooling prototype for the Functional Perspective, called "IMoG IRIS" prototype. Additionally, this activity is best supported with a dedicated tool to create and manage the use cases and user stories. This could be a common text and table manipulation tool or a more sophisticated requirements engineering tool that supports well user stories and use cases.

(Activity) **Requirements Elicitation** (Quality Requirements and Constraints)



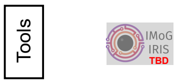
(Roles) The involved roles include the committee leader, the IMoG modeler and the corporation representatives. In-house requirements engineers are involved in this activity.

(Short activity description) The goal of this activity and the Quality Perspective is to refine the problem space by adding quality requirements and constraints to the feature hierarchy and finishing the refinement of the problem space.

(Sub-activities)

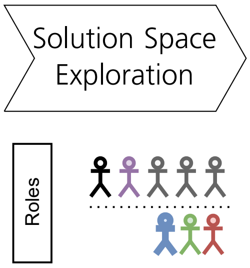
(Detailed description) The committee leader invites once again the committee members with the above mentioned roles to a meeting. Every quality requirement and constraint that came up during the identification of the features and functions is now placed into the Quality Perspective of IMoG and mapped on the features and functions of the Functional Perspective. (Note: Process Requirements are not relevant, because the innovation is not built in the committee!). Afterwards, a dedicated round of meetings moderated by the committee leader and focusing on the structured elicitation of missing quality requirements and constraints is started. These meetings follow the typical steps of requirement engineering (requirements elicitation, requirements analysis requirements documentation, requirements verification and validation [8]). The corporation representatives are responsible for eliciting the requirements and checking the consistency of the requirements. They may request their in-house requirements engineers for supporting this task. The IMoG modeler supports the corporation representatives by filling the requirements into the Quality Perspective. They stop the rounds of meetings once they are sufficiently satisfied with the result. After the meetings, the modeling of requirements and the problem space is done.

(Artifacts) The artifacts of the "Requirements Elicitation" activity are the added quality requirements and constraints in the Quality Perspective, which are mapped on the features and functions of the Functional Perspective. With the requirements elicited the modeling of the problem space is finished (or at least interpreted as a draft like in agile work / twin peaks [14]).



(Tools) Overall, we think that a dedicated tooling for IMoG is needed and thus the proposed tool for IMoG would be such dedicated tooling. We already implemented a tooling prototype for the Functional Perspective, called "IMoG IRIS" prototype. Unfortunately, we do not have enough resources to extend the dedicated prototype (IMoG IRIS) for the Quality Perspective activities. Thus standard requirements managing tools like IBM Rational DOORS or Jama are recommended.

(Activity) **Solution Space Exploration**



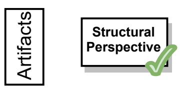
(Roles) The involved roles include the committee leader, the IMoG modeler and the corporation representatives. In-house requirements engineers are involved in this activity. The leader of this task is the system architect to examine and analyze the possible solutions. The system architect gets support from the requirements engineer and the domain expert, however, their help is of supportive nature.

(Short activity description) The goal of this activity and the goal of the Structural Perspective is to model the solution space of the innovation with (sufficient) technical details.

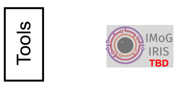
(Sub-activities)

(Detailed description) The modeling of the solution space is the next step. The committee leader invites the committee members with the above mentioned roles to explore and discuss the possible solutions. The exploration of the solutions may include the following steps (the order does not have to be strictly followed): Starting with the context level, a model describing the environment of the intended innovation solution and the innovation itself is designed. In this context model, the innovation can be understood and represented as a black box (meaning that the innovation is not decomposed or any of its parts further described). The next step may include the system decomposition focusing on how the innovation can be constructed. The innovation is represented in detail (white box). The general decomposition concepts of using logical components, "solution principles" (e.g. combustion or electric) and actual solutions (e.g. specific engines) as well as hardware and software mappings are part of the system decomposition step. This step may also include the mapping of the features and functions of the Functional Perspective on the components of the system decomposition to achieve traceability to the problem space. The third step may include an effect chain modeling to depict and analyze the connections between the innovation components (system decomposition parts) and its environment.

(Detailed description continued) The analysis allows to understand the dependencies between the innovation components and its environment and their impact on changes. The fourth step may include the structured elicitation of missing quality requirements and constraints for the solutions. This step uses the same steps already mentioned in the Quality Perspective (Adding solution requirements to QP, requirements elicitation, requirements analysis requirements documentation, requirements verification and validation [8]). The last step may include an alternatives exploration including the use of Key Performance Indicators (KPIs) to describe the possible alternatives of the system components and their advantages and limitations.
These steps can be divided over a series of meetings with internal discussions in the corporations, where the committee leader manages the formalities. The IMoG modeler and the corporation representative including their internal roles of the system architect, requirements engineer and domain expert, are responsible for exploring the solution space using the five steps. Additionally, the IMoG modeler is responsible for the creation of the Structural Perspective. The importance of each internal role varies between the steps: The system architect is most important during the context level modeling, the system decomposition and FP mapping, the effect chain analysis and the alternatives exploration. The system architect however plays a smaller role in the requirements elicitation for solutions step, where the requirements engineer has the responsibility. On the other side the requirements engineer plays a less important role in the other steps. The domain experts gives their expertise in all steps. However, their input is especially in the system decomposition, the effect chain modeling and in the alternatives exploration needed.
After the steps, the modeling of requirements and the solution space is done.

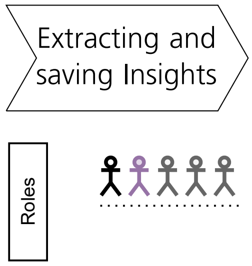


(Artifacts) The artifacts of the "Solution Space Exploration" activity are the model of the solutions covered in the Structural Perspective and its dependencies to the other perspectives. This includes the added quality requirements and constraints to the Quality Perspective and the mapping on the features and functions of the Functional Perspective. The solution space exploration may include a context model and the decomposition of the innovation including effects and alternatives.



(Tools) Overall, we think that a dedicated tooling for IMoG is needed and thus the proposed tool for IMoG would be such dedicated tooling. We already implemented a tooling prototype for the Functional Perspective, called "IMoG IRIS" prototype. Unfortunately, we do not have enough resources to extend the dedicated prototype (IMoG IRIS) for the Structural Perspective activities. Thus standard system modeling tools like any UML tool or SysML tool are recommended.

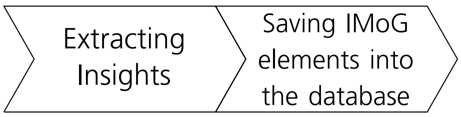
(Activity) **Extracting and saving insights** (for future IMoG innovations)



(Roles) The involved roles include the committee leader, the IMoG modeler and the corporation representatives. In-house roles are not needed in this activity.

(Short activity description) The goal of this activity and the (Domain) Knowledge Perspective is to extract the insights of the innovation to use them as a basis for the roadmap and to save the insights of the innovation for future IMoG innovations.

(Sub-activities)

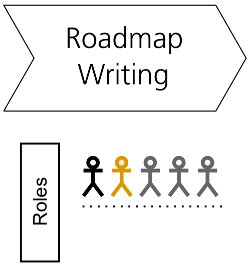


(Detailed description) The modeling of the problem space and solution space is finished. The committee leader invites the committee members with the above mentioned roles to extract the insights of the model to use them as a basis for the roadmap. The insight extraction is done by the committee members. This includes the committee leader, the IMoG modeler and the corporation representatives. The outcome of this first activity is the list of insights written down into a document. Afterwards the IMoG modeler exports the IMoG elements into a publicly available database. For this activity, the IMoG modeler asks the committee members which elements to save and which not. The IMoG modeler suggests dependencies to draw between IMoG elements and already available information and discusses these suggestions with the committee members. The committee members may also suggest dependencies. The outcome of this second activity is the publicly available database enhanced by elements of the regarded innovation.

(Artifacts) The artifacts of the “Extracting and saving insight” activity are a list of insights as a basis for the roadmap activities and the publicly available database enhanced by elements of the regarded innovation.

(Tools) Overall, we think that a dedicated tooling for IMoG including a publicly available cloud service containing the IMoG model is needed and thus the proposed tool for IMoG would be such dedicated tooling. We already implemented a tooling prototype for the Functional Perspective, called “IMoG IRIS” prototype. Unfortunately, we do not have enough resources to extend the dedicated prototype (IMoG IRIS) for the Domain Knowledge Perspective activities. Thus standard text editors and databases are recommended.

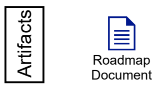
(Activity) **Roadmap Writing**



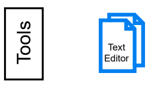
(Roles) The involved roles include the committee leader, the roadmap manager of the committee and the corporation representatives. The roadmap manager takes responsibility for the roadmap writing, structures the document, and assigns tasks. The corporation representatives are responsible for giving sufficient help in the roadmap creation, reviewing and refinement. In-house roles are not needed in this activity.

(Short activity description) The goal of this activity is to use the extracted insights of the innovation to write the roadmap.

(Detailed description) The modeling activities are finished and the insights of the innovation are extracted. The committee leader invites the roadmap manager of the committee and the committee members to write the roadmap based on the insights of the innovation. The roadmap manager of the committee creates a first draft of the document structure with sufficient support of the committee members and then refines the roadmap together with the committee members. This refinement process goes on until they are sufficiently satisfied with the roadmap.

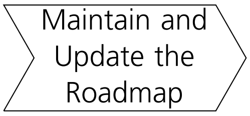


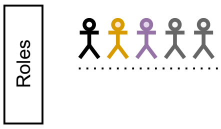
(Artifacts) The artifact of the “Roadmap Writing” activity is the roadmap.



(Tools) Standard text editors (e.g., LaTeX, word, etc.) are recommended.

(Activity) **Maintain and update the roadmap**

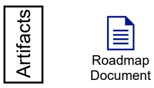




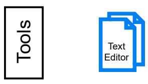
(Roles) The involved roles include the committee leader, committee leader, the IMoG modeler, the roadmap manager of the committee and the corporation representatives. In-house roles are not needed in this activity.

(Short activity description) Reoccurring meetings are established to maintain and update the roadmap.

(Detailed description) The roadmap is written! Now the committee meets once in a defined time frame to maintain and update the IMoG model and roadmap. The committee leader manages the formalities, the IMoG modeler is responsible for updating of the model and the roadmap manager of the committee is responsible for updating the roadmap. The corporation representatives are the most important members here as they decide in which direction the roadmap should be point. The outcome of this activity is the updated model and roadmap.



(Artifacts) The artifacts of the "Maintain and Update the Roadmap" activity are an updated model and an updated roadmap.



(Tools) All of the tools mentioned in the other activities are required to maintain the model and the roadmap.

3.3. Example – Mobility with an e-scooter

Let's illustrate the process with an example storyboard (see Figures 3.6 and 3.7). A manager (see the Figure on the right) from a known car manufacturer wants to dive into future mobility aspects and explore new areas for potential investments. He likes the aspect of e-scooters as part of future mobility services and decides to think through this innovation together with the automotive value chain. He starts a new committee with himself as the committee leader and publicly invites partners of the automotive value chain to join the committee. Several members of the automotive value chain join the innovation exploration. Some of them will also decline. He also creates a public invitation to increase the number of partners and thus the relevance of the committee. Some additional OEMs, Tier1 and Tier2 join the consortium. Before starting the committee he requests the committee corporations to assign the internal roles. Each OEM, Tier1 and Tier2 assign the roles of the Roadmap Manager, Requirement Engineer, System Architect and Domain Expert. Additionally, the committee leader invites suitable people to take over the role of the IMoG modeler and the role of the roadmap manager. Then the committee is ready to explore the innovation by following IMoG's process.

OEM
Committee Leader

The committee starts with the first activity – the Innovation Identification (see Figures 3.8 and 3.9). The committee leader invites the committee to a meeting to identify the innovation. The committee meets and chooses a fitting creativity method to identify the innovation they want to explore. In this case, they agree on using the creativity method https://en.wikipedia.org/wiki/Morphological_analysis_(problem-solving). The committee carries out the creativity method until they are satisfied with their result. Based on the outcome, the committee leader writes an innovation description. Afterwards, the IMoG modeler takes the creativity method outcome and the innovation description to create a draft of the Strategy Perspective. Both results are discussed and refined in the committee and internally until they are satisfied. Then the Strategy Perspective and the Innovation Identification activity is finished.

The second activity is the Feature and Function Identification (see Figures 3.10 and 3.11). The committee leader invites the committee to a meeting to identify the features and functions. The committee decides if they want to create user stories or use cases before identifying the features and functions. In this case they agree to create them. The committee starts to define user stories and use cases. The user stories and use cases are refined internally in each company with their requirements engineer and consolidated in the com-

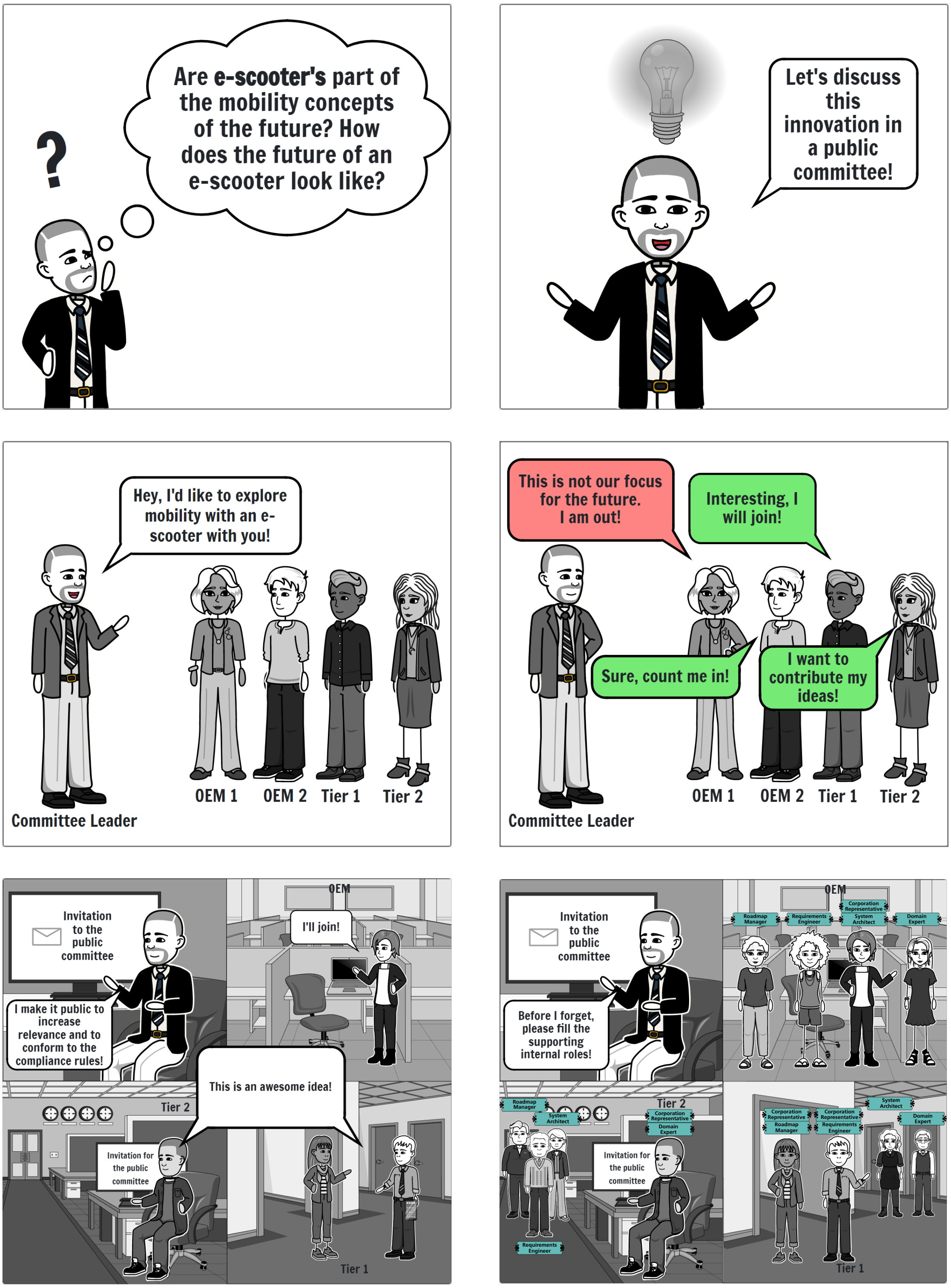


Figure 3.6.: **Roles storyboard.**

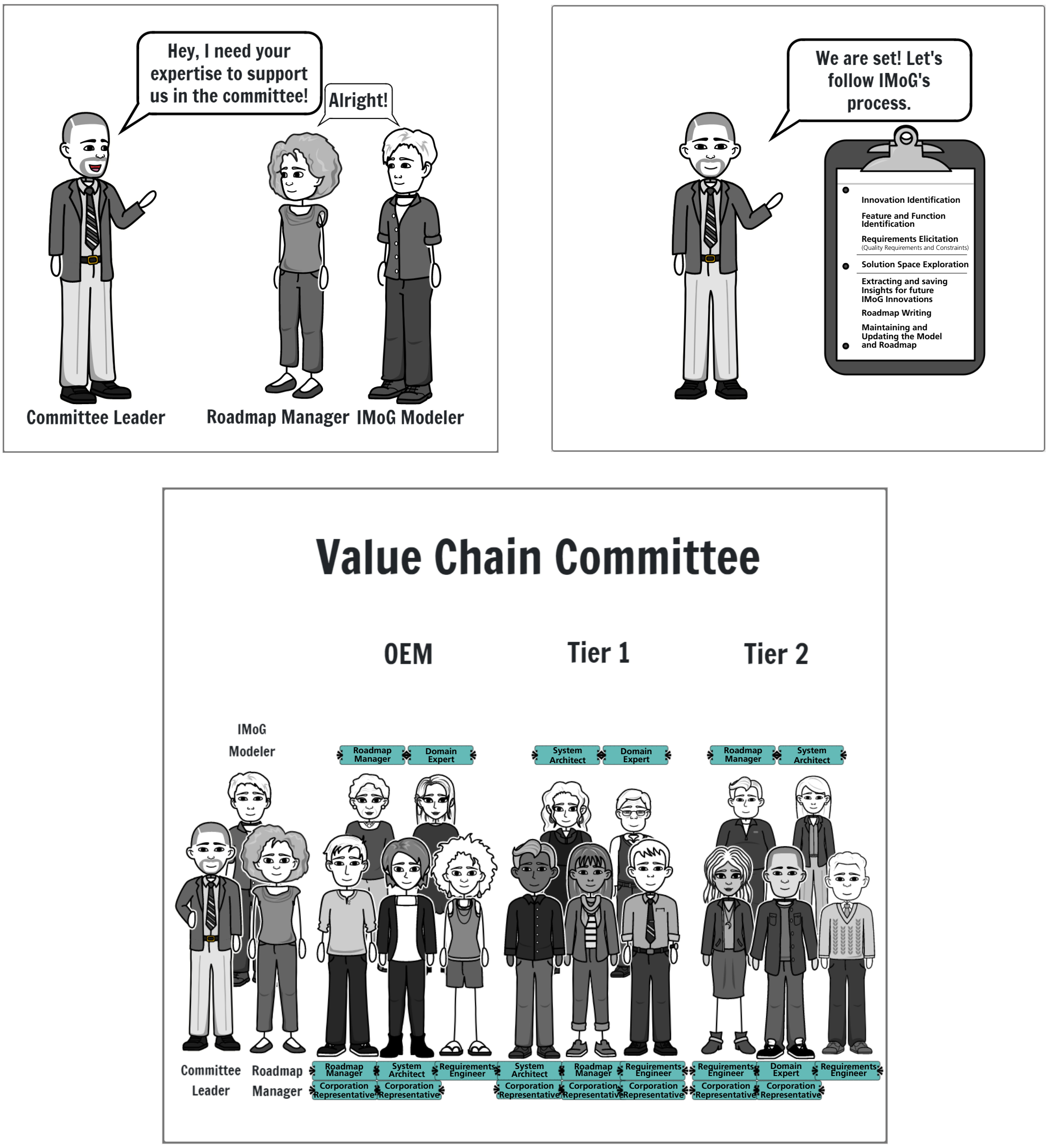


Figure 3.7.: **Roles storyboard (part 2).**

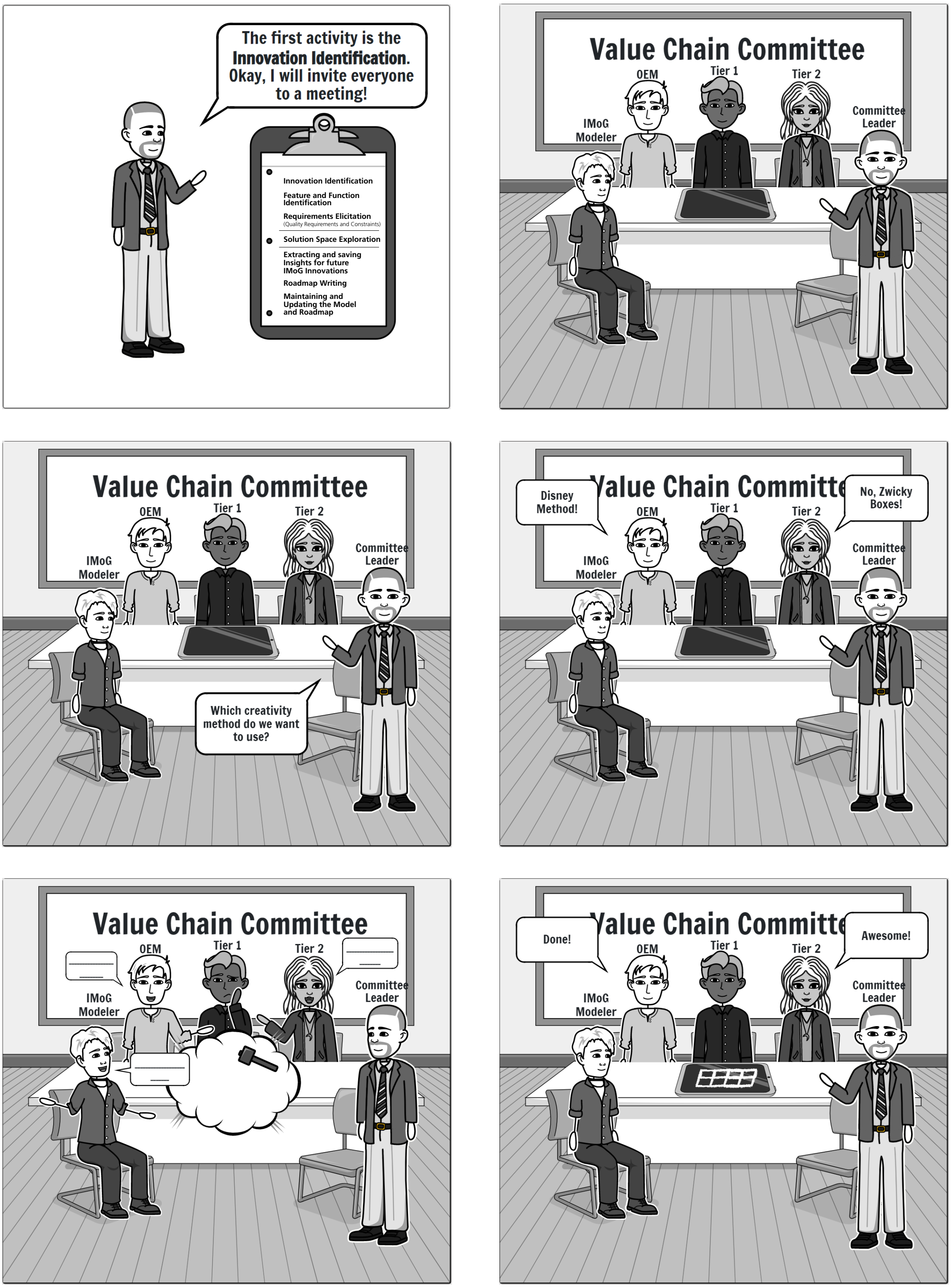


Figure 3.8.: **The first activity: Innovation Identification**

Figure 3.9.: **The first activity: Innovation Identification (part 2)**

mittee. Afterwards or before the elevation of the user stories and use cases, the features and function are defined and put into relation in a feature model. Similarly to the user stories and use cases, the feature model is refined internally in each company with their requirements engineer and consolidated in the committee. Then, the Feature and Function Identification activity is finished.

The next activity is the Requirements Elicitation. The committee leader invites the committee to a meeting. Then the committee meets to add the requirements to the features and functions (see Figures 3.12 and 3.13). They write down the requirements and constraints that were raised in the last two activities and now structurally elicit missing quality requirements and constraints for the features and functions. Once they are done, the requirements and constraints are refined and completed internally with the requirements engineer. Afterwards, the Requirements Elicitation activity and the modeling of the problem space is done.

The next activity is the Solution Space Modeling and Analysis (see Figures 3.14 and 3.15). The committee leader invites the committee to another meeting. The committee explores the solution space of the innovation by modeling the innovation and its environment. The exploration includes several steps like the context level modeling, the system decomposition and feature mapping, the effect chain and impact analysis, the requirements elicitation for solutions and the alternative and key performance indicator exploration. Once the exploration is drafted, the refinement of the innovation model takes place internally within the companies. In this activity, the system architect takes the main lead and gets support by the requirements engineer and the domain expert. When this activity is finished, the solution space is also finished.

The next activity is the extraction of insights for future IMoG innovations (see Figures 3.16 and 3.17). The committee meets again to discuss. With the problem space and solution space modeled, the key elements of the innovation are identified and listed to be used in the innovation roadmap and in future IMoG models. Once the elements are identified, the IMoG modeler puts them into the public database to be reused. Now it is time to write the roadmap!

The next activity is the roadmap writing (see Figure 3.18). The roadmap manager of the committee creates a draft of the roadmap. The committee discusses the draft and refines it until the roadmap is finished. With this activity finished the general direction is known to the value chain, which can now start their own development processes outside the committee to make the innovation come true. The main committee activities are also ended with this activity.

The last activity left open is the maintenance and update of the model and the roadmap

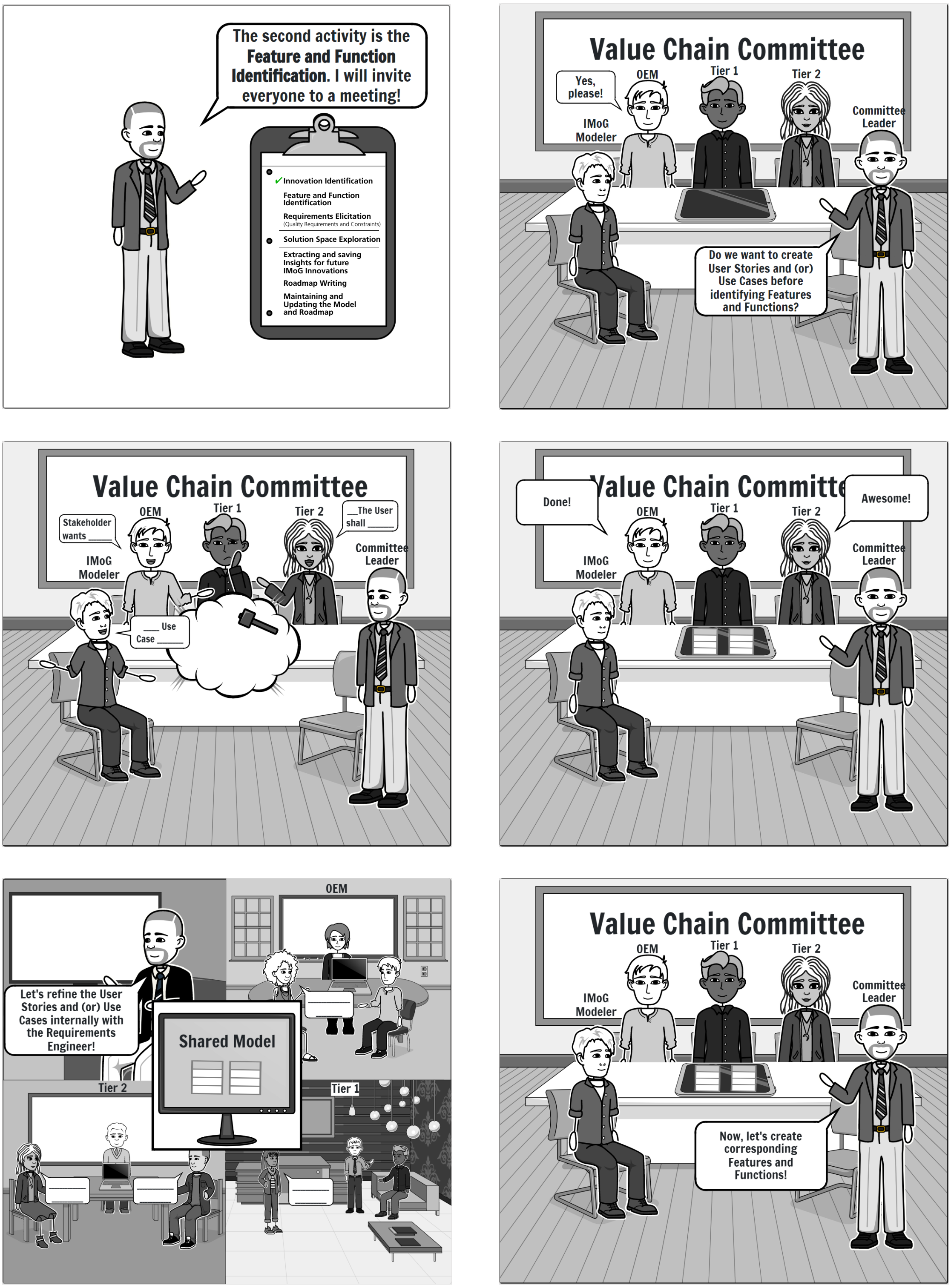


Figure 3.10.: **The second activity: Feature and Function Identification**

DLR – *Innovation Modeling Grid*

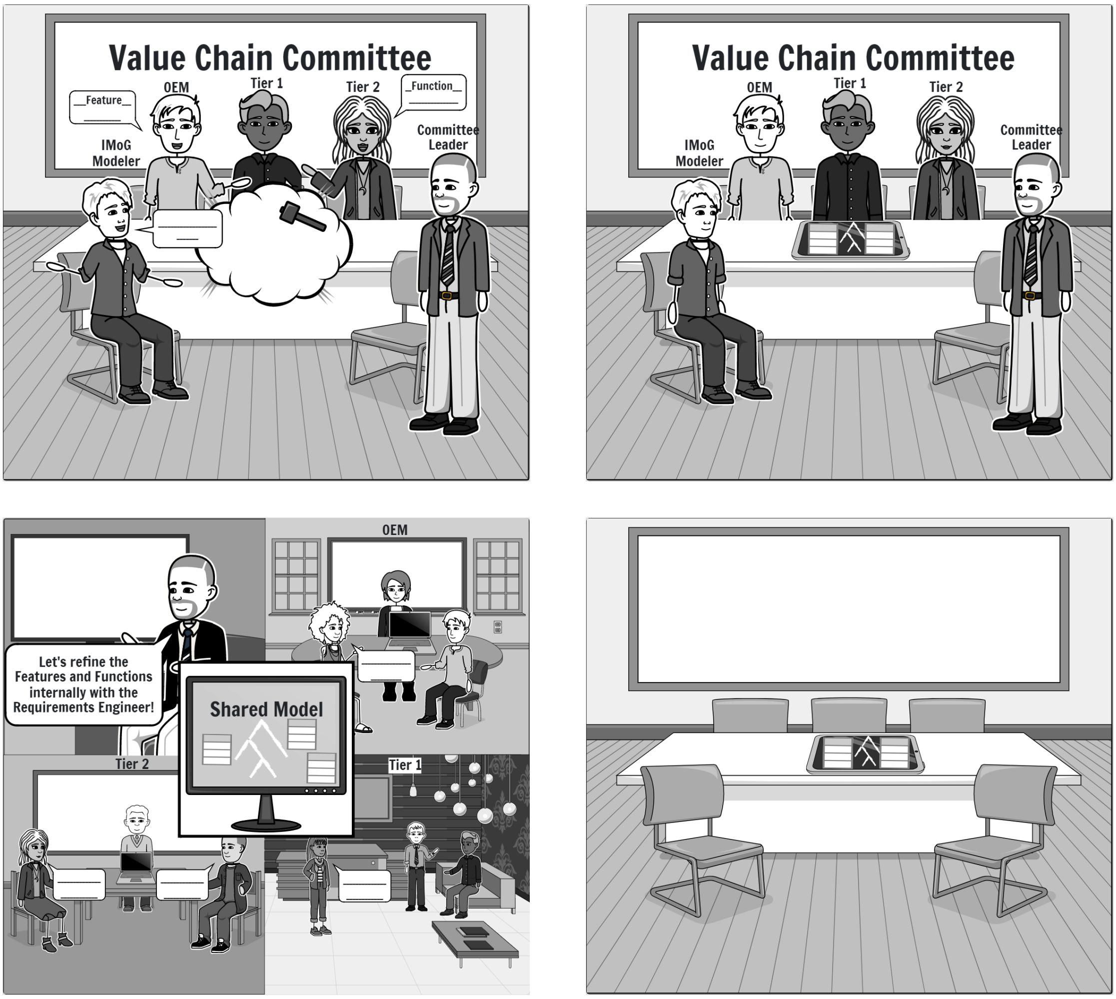


Figure 3.11.: **The second activity: Feature and Function Identification (part 2)**

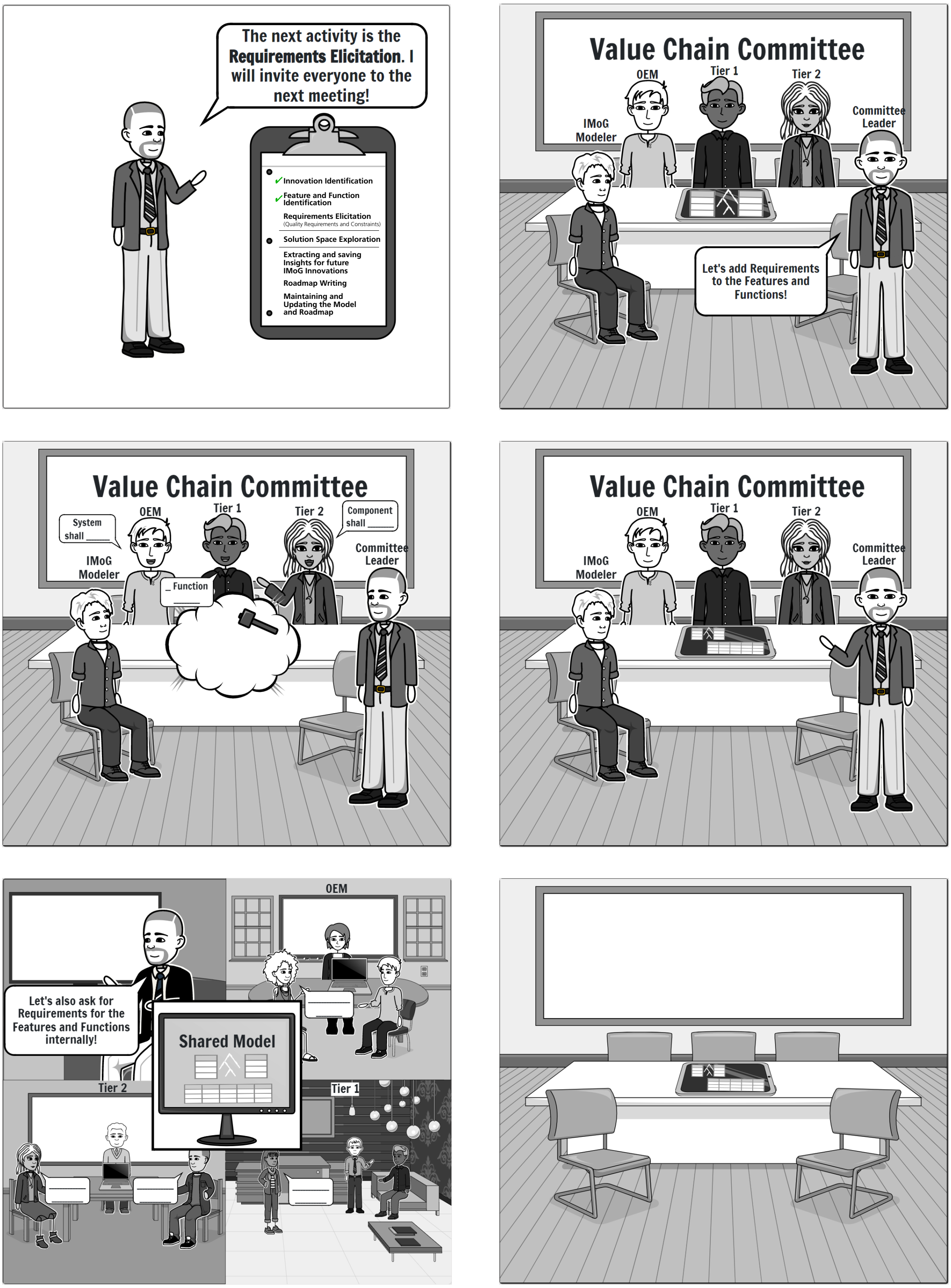


Figure 3.12.: **The third activity: Requirements Elicitation**

DLR – Innovation Modeling Grid

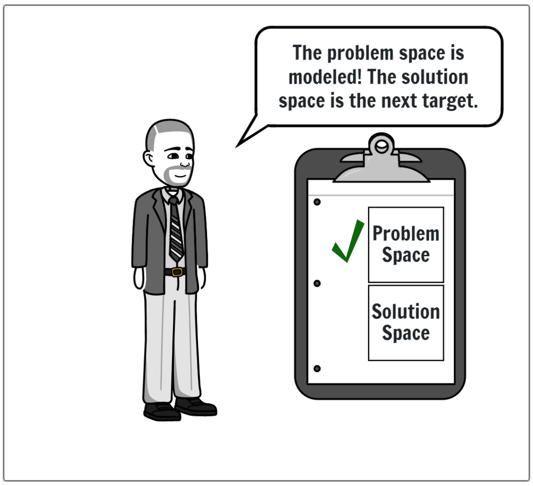


Figure 3.13.: **The third activity: Requirements Elicitation (part 2)**

(see Figures 3.19). In predefined intervals (like every two years), the committee takes place and realign the model and the roadmap with their gained knowledge over the time frame. With this activity they can also take action for important changes that the whole value chain needs to know. The model is also refined internally with all important roles. When this activity is done, the committee meets then again at the next predefined date for the maintenance and update.

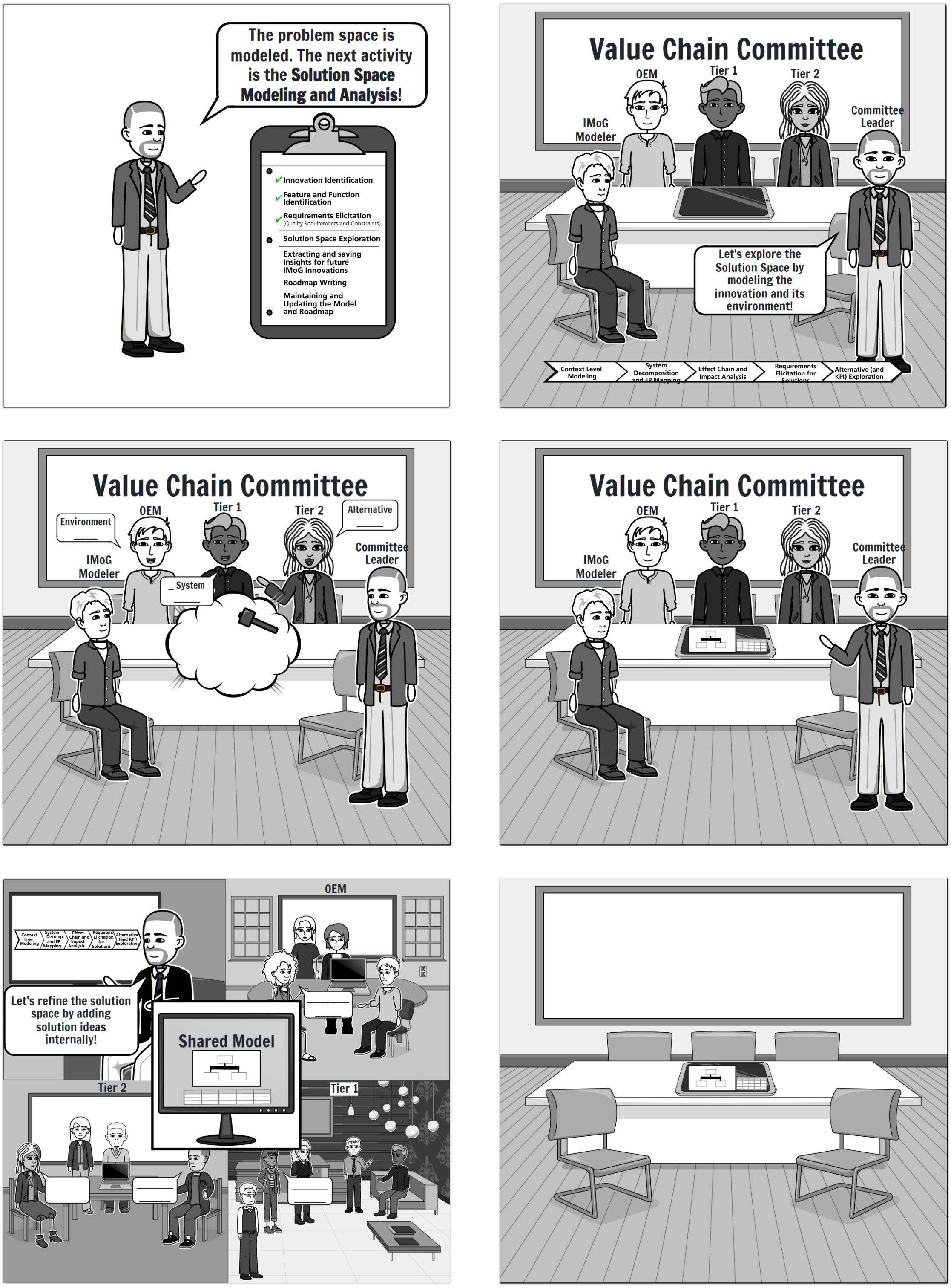


Figure 3.14.: **The fourth activity: Solution Space Modeling and Analysis**

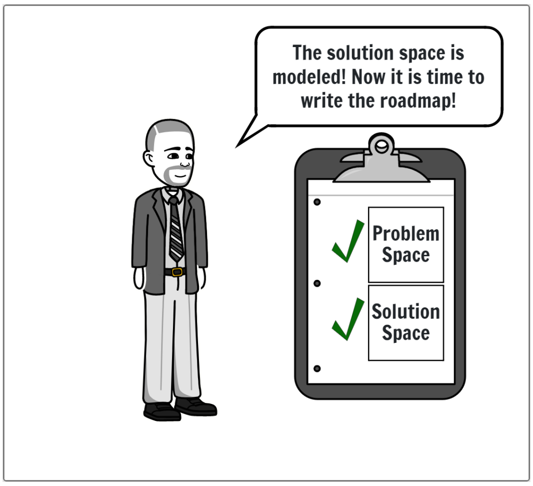


Figure 3.15.: **The fourth activity: Solution Space Modeling and Analysis (part 2)**

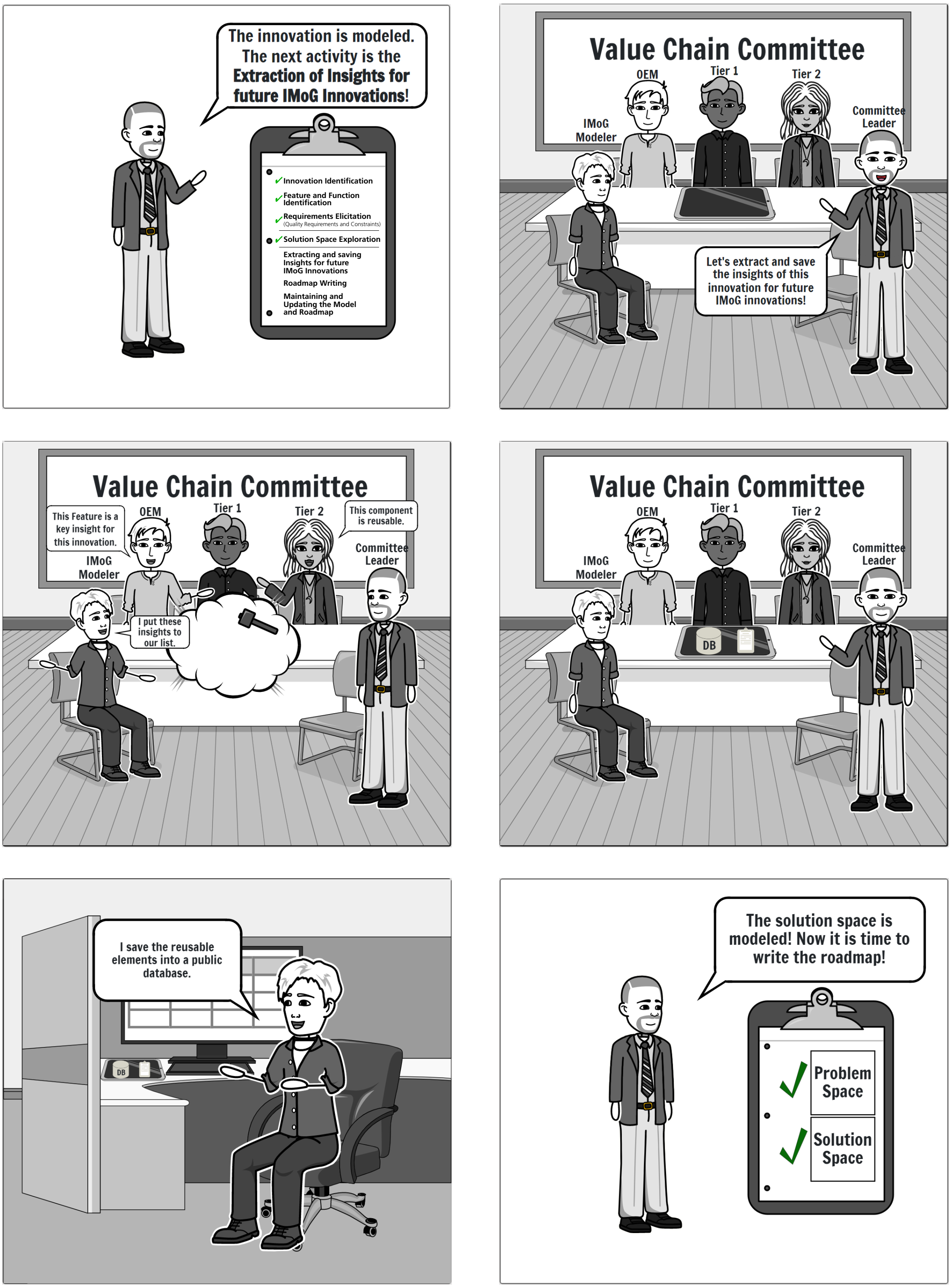


Figure 3.16.: **The fifth activity: Extraction of Insights for future IMoG Innovations**

DLR – Innovation Modeling Grid

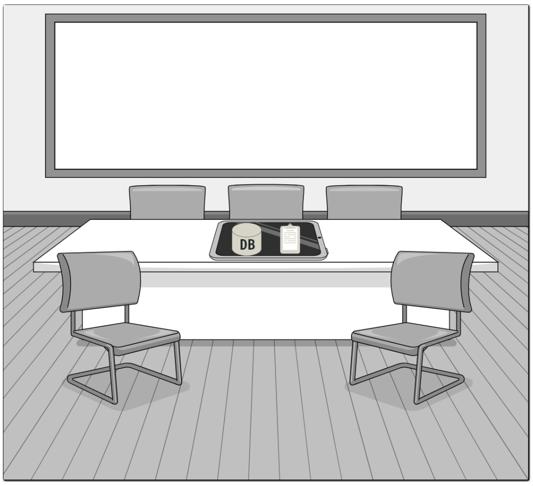


Figure 3.17.: **The fifth activity: Extraction of Insights for future IMoG Innovations (part 2)**

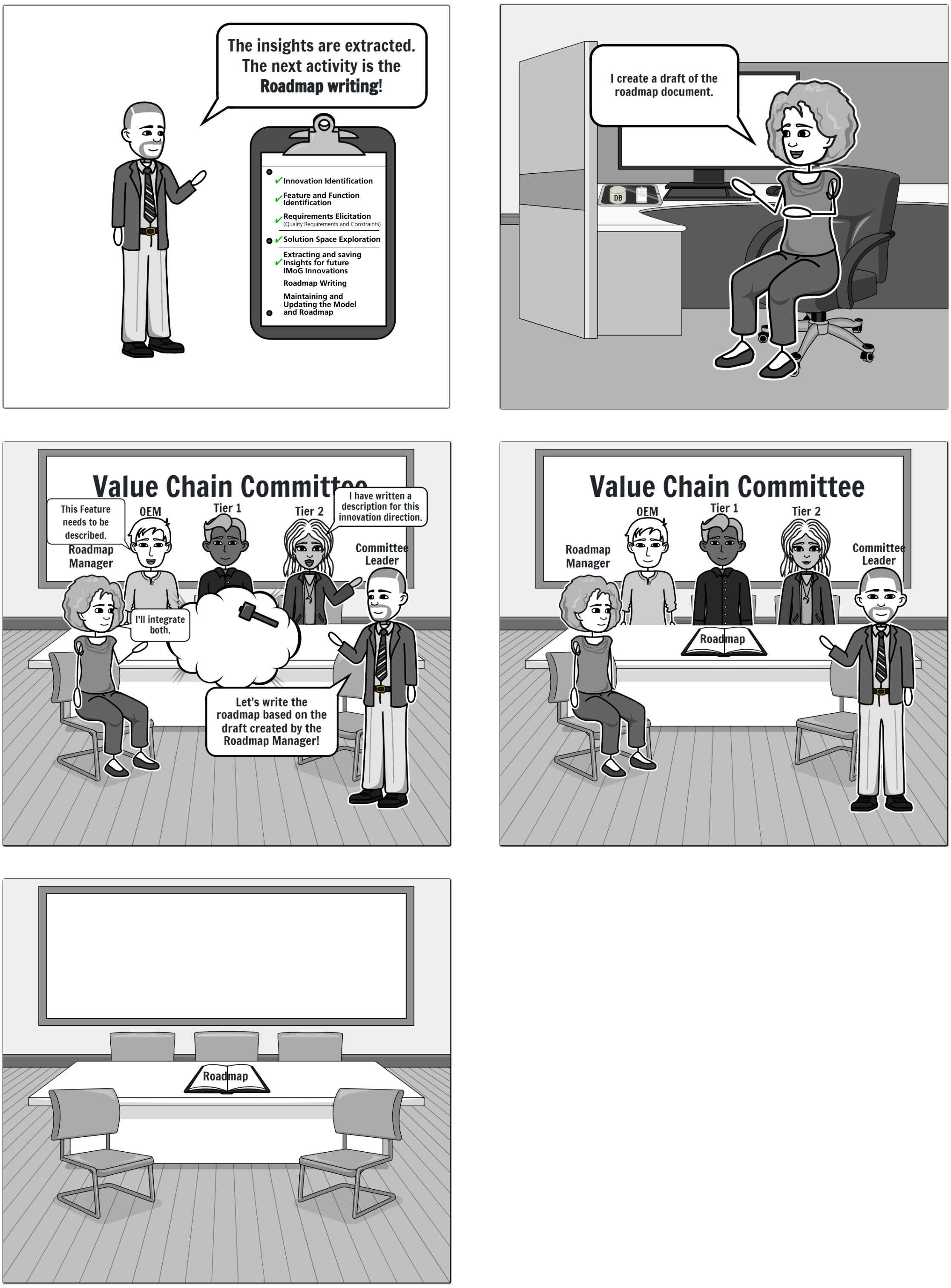


Figure 3.18.: **The sixth activity: Roadmap Writing**

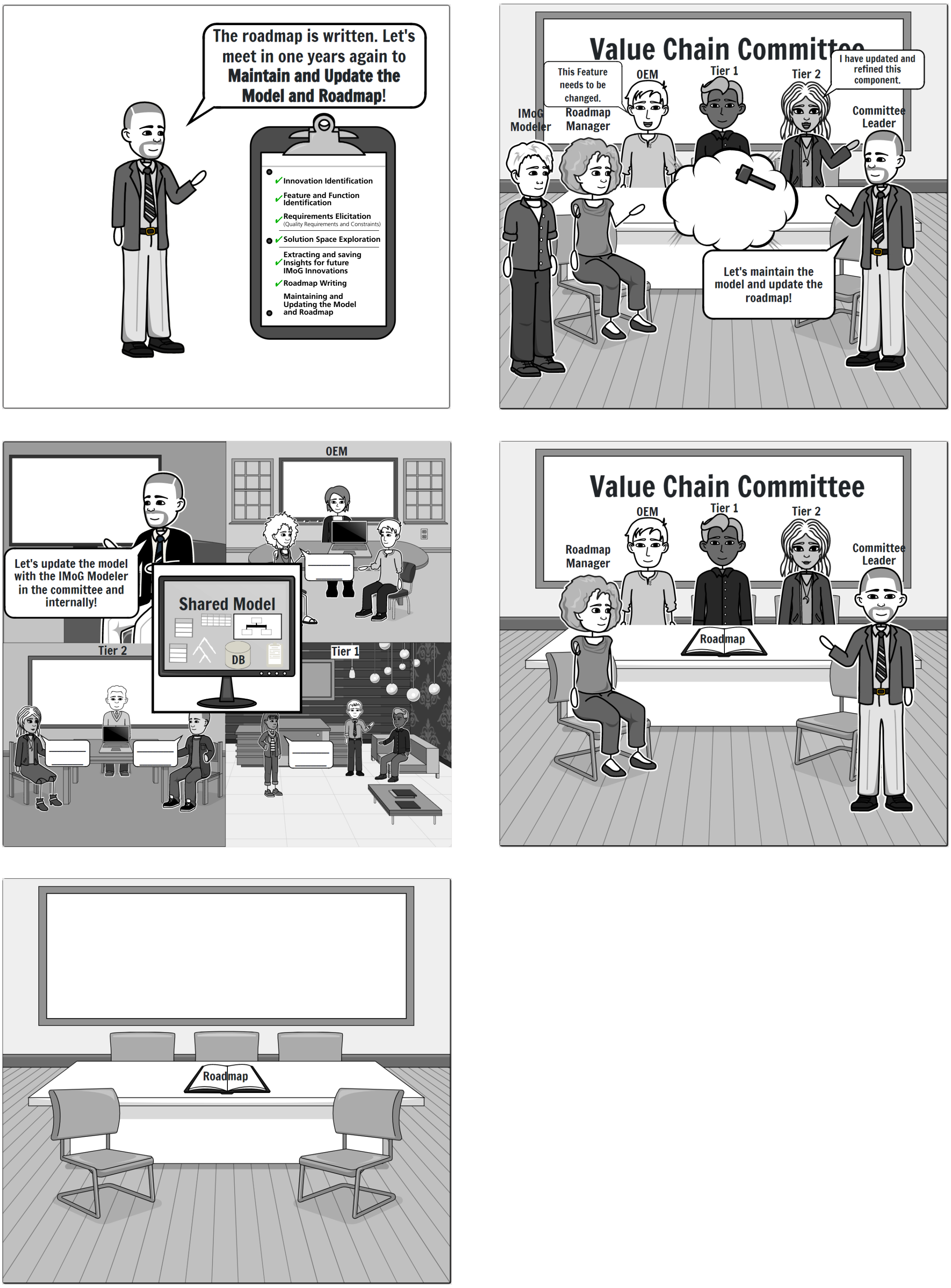


Figure 3.19.: **The seventh activity: Maintenance and Update of the Model and the Roadmap** *DLR – Innovation Modeling Grid*

4. Innovation Modeling Grid

We developed a methodology called "Innovation Modeling Grid" (IMoG) to accelerate the innovation development process along the automotive value chain. The methodology IMoG provides a structure and defines elements to model the problem and the solution space of innovations. IMoG defines a structure to reduce the time spend on the "What and how to model?" question and to help the modeler to focus on their innovation instead. Furthermore, a process and a dedicated tooling supporting the methodology is required to handle the methodology this public roadmapping context. A dedicated tooling for IMoG is currently in progress, but out of the scope of this document.

Section 4.1 presents the design principles of the methodology IMoG. IMoG itself is described in Section 4.2. An FAQ is given in Section 4.3. IMoG's process is described in Chapter 3.

4.1. Design Principles of IMoG

We developed IMoG under the context that an automotive value chain committee creates and maintains a public microelectronic roadmap. This context shaped IMoG and we raised the following design principles:

- **Abstract Innovations**: The focus of IMoG lies on describing abstract innovations that are shared in a public committee. These innovations are represented by a mix of informal and formal elements to remain beneficial to all participants. IMoG models are expected to include fewer details than development and engineering models. Therefore, complex modeling concepts are left out. This includes, for example, the concept of "Ports" to model communication interfaces of solutions and check their consistency. However, this does not mean that any kind of detail is too much. It is expected that IMoG contains sufficient details of the crucial parts of the innovation where the highest uncertainty and risk lies. Instead of ports, it is recommended to describe one communication channel with sufficient details.

- **Problem space vs solution space**: IMoG divides the modeling into a problem description and a solution description [5, 15]: The problem space should mainly contain information about the problem with as little information as necessary about the possible solutions. The solution space covers on the other side the possible solutions. Furthermore, a map between the problem space elements and the solution space elements is necessary for basic tracing. Natural language constraints, quality requirements and general conditions complete this tracing by giving the option to add further information. In the context of a road mapping committee, this problem-solution distinction is suitable, because it eliminates the frequently asked question whether a particular "Function", "Block", or "Requirement" describes in the IMoG model the target state or the actually designed state and thus helps reducing the cognitive load.

- **Support of Decomposition / Refinement / Variability**: IMoG distinguishes between three core concepts: Decomposition, Refinement and Variability. Decomposition describes a partitioning of an element into its parts. Refinement describes a more fine grained specification of an element. Variability describes possible alternatives of an element. Variability tends to create the wish for measurement and assessment, thus suitable comparison parameters should be defined. These concepts are less important for describing the problem space. However, they are invaluable to understand and apply for the solution space.

- **Other modeling dimensions**: IMoG applies other concepts to manage innovation modeling as well. The concepts of abstraction levels and perspectives help with the separation of concerns by focusing on certain aspects of the innovation. Furthermore, abstraction levels and types help with the support of filtering mechanisms to hide temporally unneeded details. The concept of availability describes when certain elements are available and ready-to-use, which plays an integral part in road mapping.

4.2. Innovation Modeling Grid Methodology

The Innovation Modeling Grid (IMoG) is depicted as a matrix in Figure 4.1. Each row represents an abstraction level, which can be understood as separating and designing the details of the innovation at different detail levels. IMoG proposes three abstraction levels:

- The **Context Level** describes the innovation as a whole system embedded into its

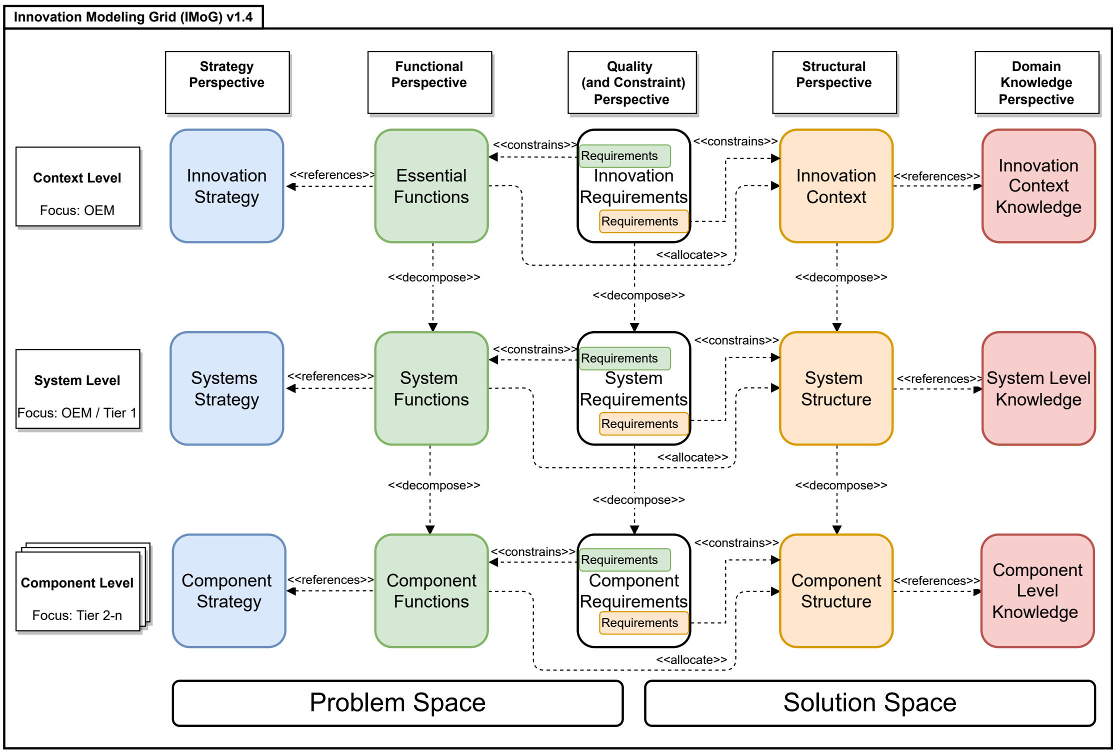


Figure 4.1.: **IMoG version 1.4. It contains three abstraction levels (rows) and five perspectives (columns). Each perspective and abstraction level is interconnected with its neighbor cell.**

environment. In the automotive domain, this level is particularly interesting for the OEM(s) in the committee.

- The **System Level** represents the innovation systems and its parts and is primarily relevant for Tier 1 suppliers in the classic automotive environment.

- The **Component Level** consists of the components of the system.

The modeler can also add more abstraction levels if needed. However, the three abstraction levels should be a sufficient starting point for the most innovations.

IMoG follows a classical approach to distinguish between the problem space and the solution space, and proposes to analyze the spaces through so-called perspectives. A perspective describes an aspect of the innovation, for example the strategy, the features or the structure of the innovation. In IMoG, each perspective is represented as a column. The Strategy Perspective, the Functional Perspective, and partly the Quality Perspective relate to the problem space and focus on describing aspects of the problem without many technical details. On the other hand, the Structural Perspective, the Domain Knowledge Perspective, and the latter part of the Quality Perspective relate to the solution space and describe potential technical solutions corresponding to the problem in an abstract manner. In the context of innovations this solution space are kept abstract as the knowledge about the future is only vague.

The five perspectives and the three levels of abstraction are arranged into a grid, where each cell in the grid represents a model of an aspect of the innovation on a specific abstraction level. A grid cell is called a *view*. Note that not all grid cells need to be filled: When a modeler does not see a purpose for one view, then there is no issue omitting this view. This may happen if a breakdown of a model is not further required. The IMoG meta model recommends for each perspective a set of model elements. The corresponding details are out of the scope of this article. Each perspective is presented in the following. Afterwards, the interconnection (and thus the arrows in Figure 4.1) between the perspectives are described.

4.2.1. Strategy Perspective

The identification of an innovation usually starts with creativity techniques, sketches and discussions. These discussions are the starting point of the Strategy Perspective. The Roadmap Manager of the committee (see Section 3.1) takes the result and writes an innovation description as well as the strategy description behind the innovation. The descrip-

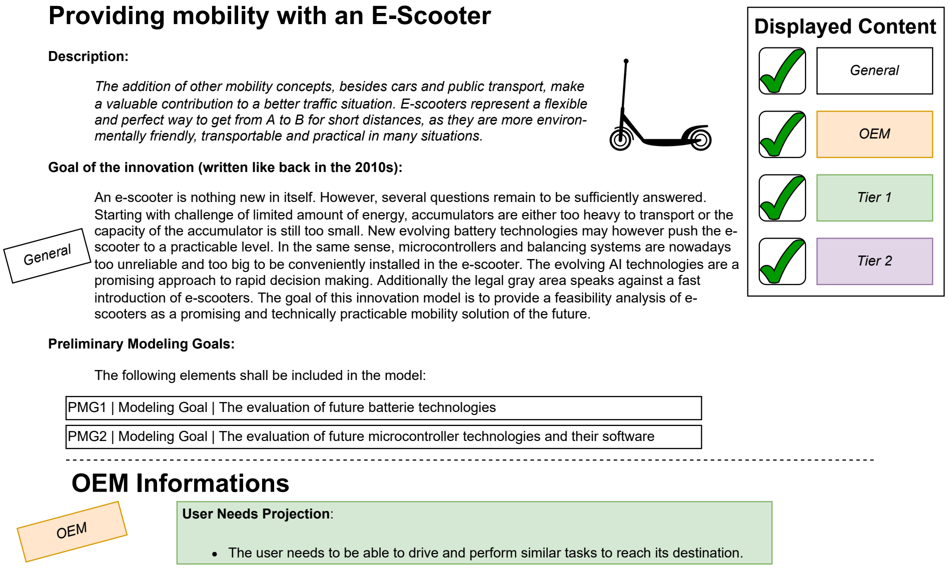


Figure 4.2.: **Strategy Perspective: part of the innovation description of the e-scooter.**

tion targets the innovation strategy, which may contain a vision, rationales, images, goals and diagrams. Those descriptions can contain identifiable elements to enable referencing and tracing. Additionally, the description contains companies' intends and their stakes in the innovation. The description and identifiable elements encompass enough information to start the modeling activities on the other Perspectives. The filled Strategy Perspective constitutes a part of the artifacts of the "Innovation Identification" activity.

In the following, we illustrate the process steps with the innovation "Providing mobility with an e-scooter" (see Figure 4.2). The Strategy Perspective of the e-scooter innovation includes a description with a vision and what the innovation is about, the goals written as text as well as goals listed as elements for cross referencing, information from the car manufacturer (OEM) regarding their estimated customer needs, their concern and possibly some additional bubble diagram for a better explanation of their interest and information from the other suppliers (Tier 1 and Tier 2) including their interest, diagrams, etc.

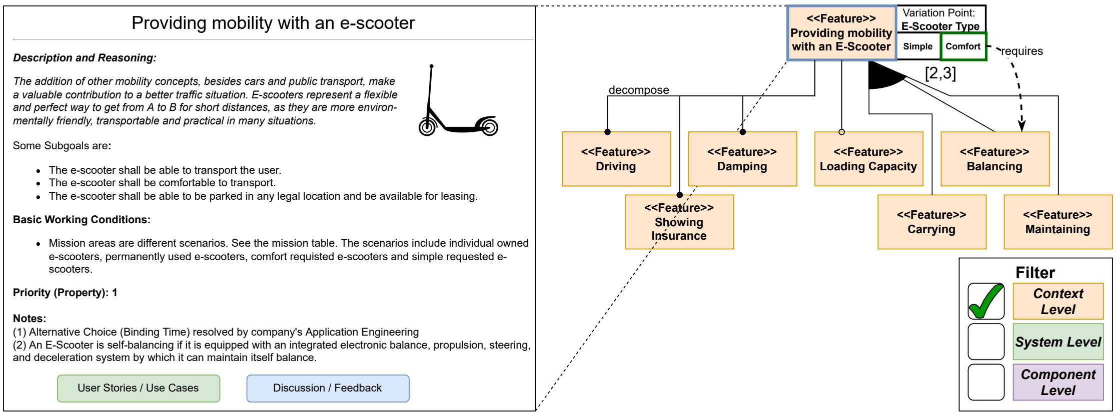


Figure 4.3.: **Functional Perspective: a part of the feature model of the e-scooter.**

4.2.2. Functional Perspective

The Functional Perspective describes the required features (end-user visible characteristics) and functions (traceable tasks or actions that a system shall perform) of the innovation. The features and functions of the Functional Perspective represent a derivative of the well-known feature models from [9]. The Functional Perspective's input is the strategy description. Optionally, User Stories or Use Cases can be created if the committee determines the need for more information on each feature and function.

Considering the e-scooter example, the Functional Perspective model may look as it is depicted in Figure 4.3. It starts with "Providing mobility with an e-scooter" as its root feature, decomposed into several other features using well-known relations. The mandatory relations are depicted using an arrow with a black circle at its end, and optional relations as an arrow with a white circle at its end. Finally, an or-relation with cardinality is depicted as a black arc with several arrows going out of it with its cardinality interval) and a constraint relation ("requires"). For more details about these described relations, see [9, 5]. The Variation Point representation represents a labeled alternative relation which graphical depiction is IMoG specific. The e-scooter feature description can be viewed on the left. It includes a detailed textual description, aligned goals, basic working conditions and other properties, like notes, priorities or links to user stories and use cases. The functions of the model are left out of the image. One specialty of the Functional Perspective is the detailed description of each feature and function, which helps to understand what they actually represent.

ID	Priority	Name	Text	Parent Req.	Labels / Sources	Target	Abstraction Layer	Version
1	1	Safety	The e-scooter shall have 2 brakes.	-	Safety Concept	Braking	Context Level	2
2	1	Braking	The braking power shall be greater than ..	-	User Needs	Braking	Context Level	1
3	2	Damping	The cushed power shall be >= ..	-	User Needs	Damping	Context Level	1
4	2	Weight	The weight shall be < ..	-	User Needs	Carrying	Context Level	1
5.1	3	Carry-Bar	The ergonomy of the handbar shall be better than ..	4 (Dcmp)	User Needs	**Carrying.Simple**	System Level	2
5.2	3	Carry-Weight	The weight shall be < ..	4 (Dcmp)	User Needs	Carrying	System Level	2
5.3	3	Carry-Platform	The platform shall be be smaller than ..	4 (Dcmp)	User Needs	Carrying	System Level	1
6	1	Stability	The stability of each subcomponent of the e-scooter shall be > ..	-	Safety Concept	Balancing	System Level	1
7	4	Modularity	Each subsystem of the e-scooter shall be build and built-in modular.	-	Safety Concept	Maintaining	System Level	1
8	2	Badge	The stability of the badge shall be > ..	-	User Needs	Show. Insurance	System Level	3
9.1	1	Load Capacity-Stability	The stability of each subcomponent of the e-scooter shall be > ..	6 (Refine)	Safety Concept	Loading Capacity	System Level	1
9.2	1	Load Capacity-Engine	The engine power shall be > ..	0 (Dcmp)	Engine	Engine	System Level	1
9.3	1	Load Capacity-Brake	The braking power shall be greater than ..	2 (Dcmp)	Safety Concept	Loading Capacity	System Level	3

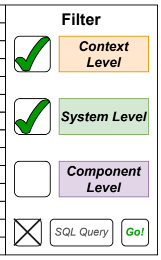


Figure 4.4.: **Quality Perspective: a typical table of requirements with many attributes, which reference features or functions of the Functional Perspective or solution blocks of the Structural Perspective. The details – like the meaning of the attributes – can be chosen depending on each innovation and are not further elaborated here. As depicted on the right side of the image, filter functionality is of special importance for the Quality Perspective.**

4.2.3. Quality Perspective

Based on the strategy description and the features and functions, the Quality Perspective captures further quality requirements and constraints of each feature and function. Requirement diagrams and requirement tables are suitable representations of the Quality Perspective. The strategy description, the features and functions and the requirements and constraints build together the problem space. It shall noted, that the Quality Perspective also contains the quality requirements and constraints of the solution space, which are referenced on the solutions on the Structural Perspective.

The e-scooter innovation's Quality Perspective is depicted in Figure 4.4. It contains the quality requirements for the problem space and for the solution space.

4.2.4. Structural Perspective

The Structural Perspective targets the modeling of the solution space. It is worth mentioning, that the word "Structural" does not mean here the relations of solution blocks to each other alone, but also includes properties and values of these solution blocks. The context level of the Structural Perspective contains the environment, the relation and the effects between the environment and the innovation. A simple environment description may for example contain the street, the driver and the e-scooter (see Figure 4.5). It contains the innovation (e-scooter) with the driver and roadway blocks (blue rectangles with a name

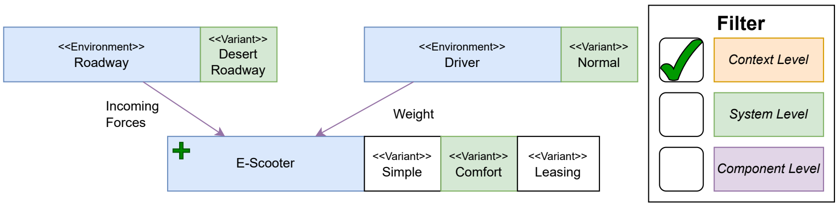


Figure 4.5.: **Structural Perspective - Context Level: A simple context model for the e-scooter.**

and optionally a stereotype over the name). Each block has variants attached, that specify similar forms of solutions. The variants are depicted (as green and white boxes) next to the solution blocks with the stereotype «Variant». Each of these blocks own properties, which further refine the block. The properties are crucial information for analyzing and evaluating the solution space. Furthermore, relations like 'Incoming Forces' and 'weight' are modeled as unidirectional (purple) arrows, where purple represents the color for relations stereotyped as «effect». The solution blocks of the different abstraction levels are left out of the model.

The system level contains the decomposition of the innovation into components, while also including the software and the hardware elements. Software and hardware elements as well as architectures and mappings between them are included in the system level. The component level encompasses the system atoms which are decomposed from the system blocks. These may include sensor descriptions with parameters, functions, properties or abstract technologies. The atoms may include sensor descriptions with parameters, functions, properties or abstract technologies. When creating a solution space any form of constraints and parameters of chosen technologies are particularly of interest. Furthermore, requirements can be added to any solution block on any abstraction level. These requirements are placed on the Quality Perspective and referenced on the corresponding solution blocks on the Structural Perspective. An example of a system model can be viewed in Figure 4.6: it decomposes the e-scooter block known from Figure 4.5 into several parts of the e-scooter. The model elements are designed specifically for the microelectronic context.

4.2.5. Domain Knowledge Perspective

The insights from the innovation and the reusable element are collected and stored on the Domain Knowledge Perspective. The elements of the Domain Knowledge Perspective enable references to the finished innovation model in future innovation models (see

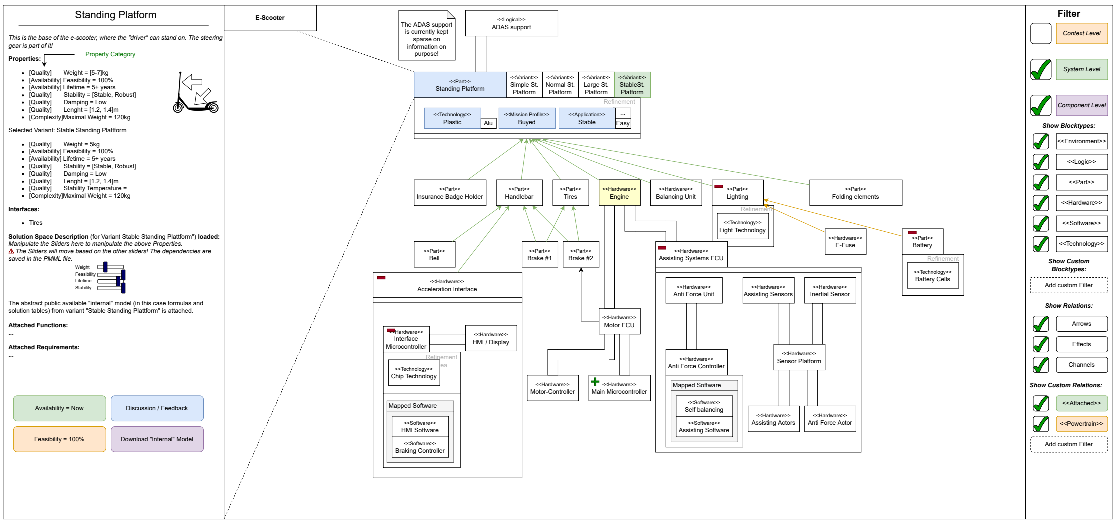


Figure 4.6.: **Structural Perspective - System Level: The decomposition of the e-scooter into its system parts. It contains many blocks, variants, relations and channels (for modeling communication). This figure shall only give a glance at what may be included in the Structural Perspective.**

ID	Name	Type	Future Availability	Property 1	Property 2	Property 3	Property 4	Property 5	Property 6	Property 7
1	Disk Brake	Brake	Now	Weigth = 1kg	Lifetime = 5y	Max. Avg. Force = 40N	Stability Temperature = [-10,55]°	Max UV Radiation = 9	Max Gear Wear = Medium	Max Chem Re
2	Brushless Brake	Brake	Now	Weigth = 50g	Lifetime = 5y	Max. Avg. Force = 40N	Stability Temperature = [-10,55]°	Max UV Radiation = 9	Max Gear Wear = Low	Max Chem Re
3	Block Brake	Brake	Now	Weigth = 500g	Lifetime = 5y	Max. Avg. Force = 50N	Stability Temperature = [-10,55]°	Max UV Radiation = 9	Max Gear Wear = High	Max Chem Re
4	Strong Light Brake	Brake	2025	Weigth = 100g	Lifetime = 5y	Max. Avg. Force = 70N	Stability Temperature = [-10,55]°	Max UV Radiation = 5	Max Gear Wear = Medium	Max Chem Re
1	Small Battery	Battery	Now	Weigth = 1kg	Lifetime = 5y	Max. Avg. Force = 30N	Stability Temperature = [-10,55]°	Max UV Radiation = 9	Max Gear Wear = Medium	Max Chem Re
2	Medium Battery	Battery	Now	Weigth = 2kg	Lifetime = 5y	Max. Avg. Force = 30N	Stability Temperature = [-10,55]°	Max UV Radiation = 9	Max Gear Wear = Medium	Max Chem Re
3	Large Battery	Battery	Now	Weigth = 4kg	Lifetime = 5y	Max. Avg. Force = 30N	Stability Temperature = [-10,55]°	Max UV Radiation = 9	Max Gear Wear = Medium	Max Chem Re
4	Super Cells	Battery	2030	Weigth = 3kg	Lifetime = 5y	Max. Avg. Force = 30N	Stability Temperature = [-10,55]°	Max UV Radiation = 9	Max Gear Wear = Medium	Max Chem Re

Figure 4.7.: **Domain Knowledge Perspective: A glance at the database view of the Domain Knowledge Perspective.**

Figure 4.7). Furthermore, the Domain Knowledge Perspective may contain a component database in a knowledge representation. The database may, for example, contain sensor characteristics and constraints from road traffic regulations, with each element owning an id, a name, a type, an estimated year of availability and several properties depending on the context of innovation.

In essence, this perspective is used to refine the model with existing knowledge and constraints. Afterwards, the gained insights can be used to write the roadmap!

4.2.6. Connecting Perspectives

All perspectives were presented in detail. However, their interconnection needs to be mentioned. These interconnections are already visible in Figure 4.1 and shall be described here briefly. The elements of the Strategy Perspective can be referred by the features and functions, building the interconnection between the Strategy Perspective and Functional Perspective (represented by the «references» relation in Figure 4.1). The constraints are part of the Quality Perspective and own a target reference to the corresponding features and functions. The same holds for the requirements mapped on the Structural Perspective's solution blocks. Thus the Quality Perspective has traces to both Functional Perspective and Structural Perspective (represented by the «constrains» relation in Figure 4.1). Each feature and function should be mapped on one or several solution blocks (represented by the «allocate» relation in Figure 4.1). This allocation is crucial, because it represents the interconnection of the problem space with the solution space. Finally, there is the reference between the solution blocks of the Structural Perspective and the Domain Knowledge Perspective (represented by the «references» relation in Figure 4.1). Thus all perspectives are interconnected to each other. Worth to note is, that the IMoG modeler has to take care of not introducing inconsistencies (e.g., a requirement that is mapped on a feature or function, which is then allocated on a solution block that owns a contradicting requirement).

4.2.7. Reviewing IMoG: Pros and Cons

IMoG's definition comes with strengths, some limitations and some recommendations from the authors. These strengths, limitations are presented in the following including some recommendations and the experience from the authors.

One strength of IMoG is that it is well defined and owns a concise meta model for innovations. This is illustrated by the following points. First, the distinction between problem and solution space reduces the thinking overhead when exploring innovations. One can first focus on the problem and its needs before diving into solutions. This distinction was evaluated in IMoG's definition phase as very helpful. Second, the recommended elements of IMoG are on an appropriate level of abstraction for modeling innovations. Elements that are required in engineering phases are left out. This is especially true for innovations discussed in committees. Third, IMoG's perspectives and abstraction levels represent a good choice. These perspectives and abstraction levels are not too many or too detailed, however, they do capture the important aspects of innovations, like strategies, features and functions, requirements and constraints as well as solutions and properties. And fourth, the handling of availability and variability is supported as well, which is crucial for modeling future innovation while coping with the design space.

Another strength is IMoG's flexibility. It is possible to start an innovation considering market pulls as well as with considering technology pushs. A market pull is understood as an innovation that is driven by a demand of the market. A market pull in IMoG starts with the identification of the innovation from the Strategy Perspective and then slowly moves over the functions and requirements to the solutions. A technology push is understood as an innovation that is driven by the development of a new technology. A technology push in IMoG starts with modeling the technology on the solution level and then slowly explores the possible demands on the Quality Perspective, Functional Perspective and Strategy Perspective. Another sign of IMoG's flexibility lies in its domain agnostic-ness. IMoG targets microelectronic innovations discussed in a committee, however many elements of IMoG are abstract enough to be used in any context. Elements like software and hardware are more system related, however, still very abstract. Thus, IMoG can be applied to similar (enough) problems.

Furthermore, IMoG is easy to apply with an IMoG expert. By guiding through the exploration process substantial time can be saved as the people creating the idea do not have to bother with the modeling elements and modeling decisions. This was also validated in the evaluation studies conducted by the authors. IMoG's validity, usefulness and adequacy were all positively evaluated.

IMoG also has some limitations. Its high abstraction is the cost of flexibility. Without having an IMoG expert, it is challenging to find a suitable path through the grid for a specific innovation, because multiple paths may seem valuable. Furthermore, detailed behavioral models are out of scope of IMoG. This can be considered a strength as state machines and alike are often too much detail for innovations. And if the detailed behavioral models are really required, they may be added as an attachment. On the other side, the high level of abstraction is definitely a challenge when transforming the IMoG model into a product level model. A transformation approach is required here that takes the innovation's context into account (e.g. see Broy et al. in [3]). Overall, IMoG is difficult to apply without guidance from an IMoG expert.

Another limitation of IMoG is the scalability known from other modeling languages. Its graphical nature does not scale very well in large diagrams, however, innovation modeling tend to have a small amount of elements. Therefore, scalability was not yet identified as a big problem.

Intellectual Property protection is of high importance in committees. This limitation is not tackled by IMoG, however, it does not restrict the use of further approaches tackling this issue while using IMoG.

From the experience of the authors, the following three recommendations support the application of IMoG: First, it is recommended to interpret the abstraction levels as filtering mechanisms. This thinking helps to apply abstraction levels only when they provide a clear advantage and not just "… because IMoG says so". Furthermore, it is recommended to search for an IMoG expert before starting the innovation modeling in a committee. Without one, the whole modeling phase may become quite challenging and inefficient. This may include improvised, ineffective meetings with inconsistent diagram exchanges. Finally, it is recommended to make use of the Glossary and FAQ, that was created for IMoG as well as for every perspective.

4.3. FAQ

The Roadmap Manager, the Requirements Manager, the System Engineer and the Domain Expert are the Stakeholders mentioned in IMoG. However Requirements Engineering and Systems Modeling often contain much more Stakeholders, Why are there only these few Stakeholders defined?

Requirements Engineering and Systems Modeling often present the Stakeholders which do have a stake in the product like the customer, the maintainer, the investor, etc. However, in the context of roadmapping innovations in a committee modeling those Stakeholders is often not needed. This stems from the abstract level of the roadmapping activities and the avoidance of details to not unnecessary constrain the solution space and protect Intellectual Property. The here mentioned Stakeholders are the roles that build the innovation model in the committee. To not assume or restrict the internal structures and to not confuse the participating corporations, the Stakeholder for each Perspectives are intentionally kept abstract. The Stakeholder descriptions exists for hinting on the participation roles in IMoG and are expected to be filled by people that have a different role description in their company. One person may even fulfil more than one Stakeholder role in IMoG.

Capturing and modeling failures and paths that did not succeed can help enormously to avoid repeating the same error. Is there a way to model failures in IMoG?

There is currently no support for modeling failures. One may include a custom stereotype and filter mechanism to support failures, however this would introduce some difficulties in tracing and coverage analysis as well as generating integration problems and cause model bloat. Additionally failures may not only include solutions but Features and Functions as well. For example, the Flying and Diving functionality combined may be unfeasible and represent a failure in the model. The removal or explicitly marking of one of those functionalities as a failure may incorporate many blocks and relations and cause the model to be broken. Thus, there is no support for failure modeling. To model failures one may use mechanisms of version control, solely existing failure models and textual descriptions to cover this need. Additionally one may use the Knowledge Perspective to add knowledge about failures and add descriptions of failed blocks into the database. The latter options are considered sufficient.

A typical way to move from the Problem Space to the Solution Space is to model first functions, then describe the technology (or concepts) and lastly describe the solutions via elements like components and parts. The technology and concepts are considered constraints. How does this move look like in IMoG?

In IMoG this move can be represented in a similar fashion: First the functionality is described on the Functional Perspective. Then the technology and solutions are both described on the Structural Perspective: The technology can be represented by using so called Templates of Groups of Components with its connections. If the technology description contain some constraints, these can be described as Requirements on the Quality Perspective and linked on the templates and inner components. The solutions can be represented as the components (or Blocks) of the templates. Thus, an element mapping may look as follows: Function ➔ Technology Template ➔ Block (Part).

Does a Problem-Solution Separation exist in IMoG?

Yes, the Strategy Perspective, Functional Perspective and half of the Quality Perspective belong to the Problem Space. The Structural Perspective, the Knowledge Perspective and the other part of the Quality Perspective belong to the Solution Space.

How to properly handle changes in the IMoG model?

Analyses like Change Impact Analyses, Version Control with Diff Tools that show the differences on a model level (similar to Git, but on a different abstraction level) and other model evolution analysis shall be provided. The implementation details are yet to be defined. A prototypical, IMoG independent implementation can be viewed in the Tool Iris from the University of Ulm.

Where is the line to distinguish between abstraction layers?

There is no clear recommendation that says when to use which abstraction layber. Instead, it is recommended to interpret the abstraction levels as filtering mechanisms. This thinking helps to apply abstraction levels only when they provide a clear advantage and not just because IMoG says so.

Does IMoG support Change Management?

IMoGs definition does not address this topic, because Change Management is viewed as an orthogonal topic. The model management and tooling should support Change Management via Version Control Systems and suitable processes.

Part II.

Details on the IMoG Methodology

5. Strategy Perspective

The Strategy Perspective is the first perspective in IMoG applications and targets the capturing of the chosen innovation (see Figure 5.1). The purpose of the Strategy Perspective is the capturing of the innovation idea as well as the capturing of the interests and strategies of the stakeholders. Regarding, the IMoG process, the Strategy Perspective is the generated artifact of the first activity: "Innovation Identification". The identification itself is not part of the Strategy Perspective. It is recommended to use creativity techniques to identify the innovation and use afterwards the Strategy Perspective to describe the innovation extensively. The choice of creativity technique should be chosen based on the preferences and experience of the involved stakeholders in the committee. One key principle of the

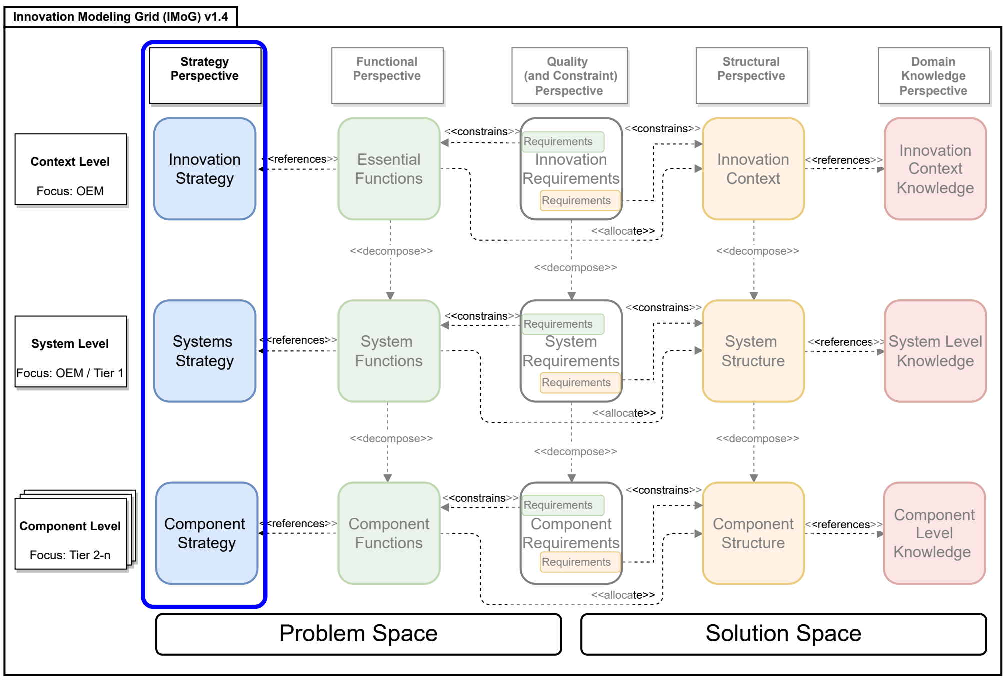


Figure 5.1.: **Location of the Strategy Perspective in IMoG**

Strategy Perspective is to not bother the committee with modeling restriction and give as much freedom as possible to capture the early innovation. Thus, the only guidelines of the Strategy Perspective care of labeling and referencing, to allow a tracing of information. Overall, the Strategy Perspective can be seen like the presentation of the innovation to externals.

The innovation identification activity and the capturing of the innovation can be imagined as followed (see Figure 5.2): The identification of an innovation starts with many discussions, sketches and non formal descriptions. For such activities, creativity techniques like brainstorming, scenario projection and zwicky boxes are suited. Based on these ideas, marketing analysts take the idea and perform market segment analysis and analyze business opportunities. This may include looking on user needs, environmental constraints, business models, time to market predictions and more. The outcome of these creativity results and analyses is the starting point of the Strategy Perspective. The committee leader takes the outcome and writes a draft of the innovation description. The description is then refined with the committee. The description may contain a vision, an explanation about the overall strategy, goals and diagrams. Identifiable elements can also be added to the content of the Strategy Perspective to allow referencing and tracing of goals, text phrases etc. The description and identifiable elements encompasses enough information to start the real modeling activities on the other perspectives.

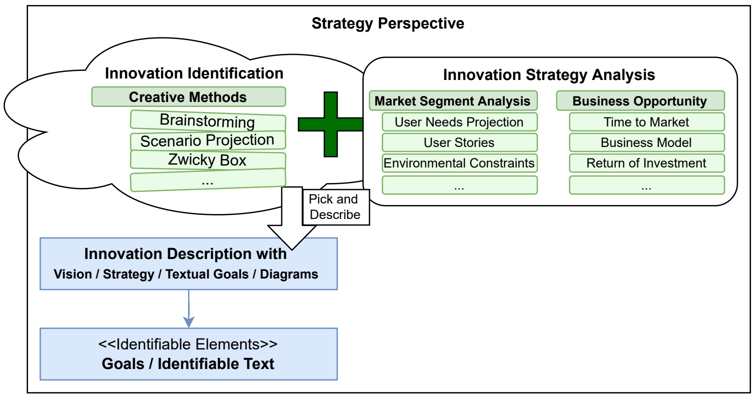


Figure 5.2.: **Activities considered for the Strategy Perspective**

The chapter is structured as followed: In Section 5.1 the meta model and its model elements are presented. In Section 5.2 an example of the Strategy Perspective is given. The strengths and limitations of the Strategy Perspective are discussed in Section 5.3. A FAQ finalizes the description in Section 5.4.

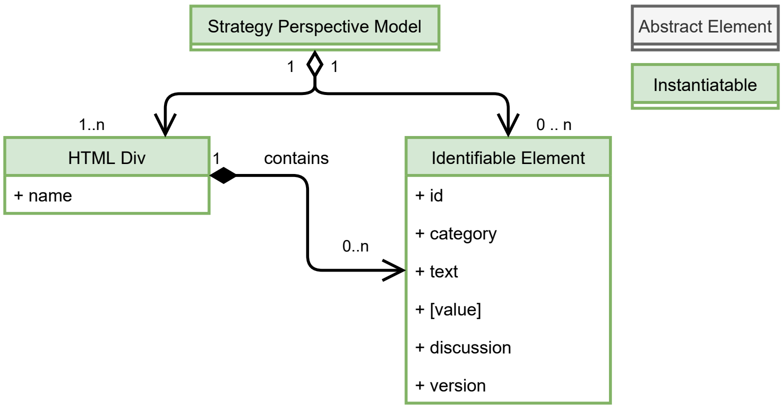


Figure 5.3.: **The model elements of the Strategy Perspective.**

5.1. Model elements

The meta model of the Strategy Perspective is kept simple. When discussing strategy it is uncommon to just start modeling activities. Instead the committee is mostly interested on the description of their interests. This meta model tries to encompass this view by only introducing descriptions (in form of HTML divs) and traceable (identifiable) elements in the Strategy Perspective model. The descriptions can be labeled to allow filtering them out. There are no relations defined for connecting different information on the Strategy Perspective. However, the identifiable elements are defined for the purpose of perspective cross-referencing (including references to functions, requirements and structure).

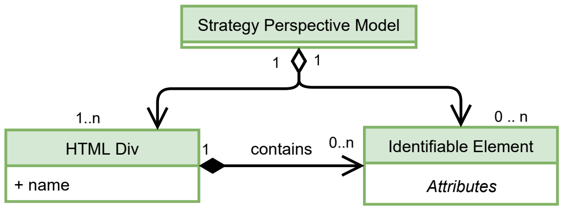
Meta Model Element:

Description:

Strategy Perspective Model
The Strategy Perspective Model is the underlying content of the Strategy Perspective of an innovation. It contains all HTML Divs and identifiable elements.

Example: A full Strategy Perspective Model example is shown in Section 5.2.

Meta Model Element:

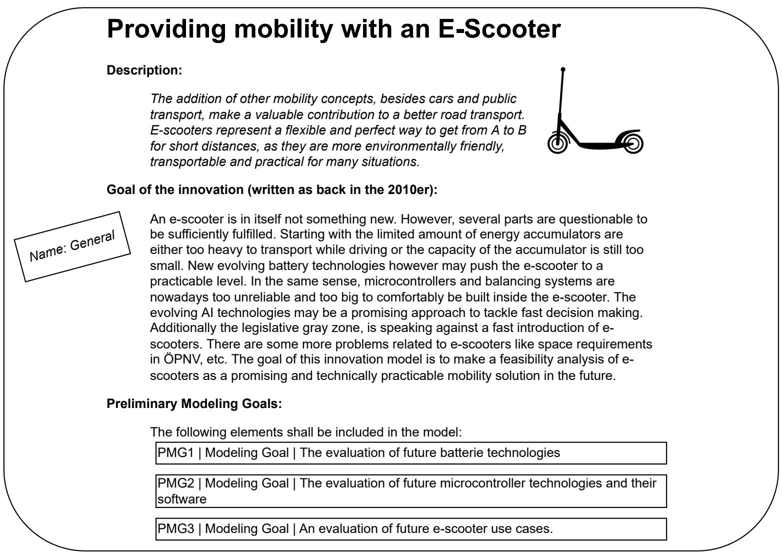


Description:

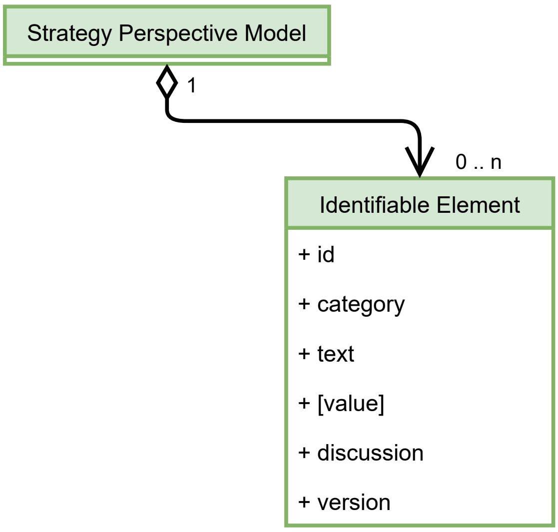
HTML div

The HTML Div is the container for all descriptions, textual goals, diagrams, etc. Additionally, they contain the Identifiable Elements embedded in their descriptions. The HTML Divs can be named to allow filtering them out by the tooling.

Example: The round rectangle represents an example HTML div with a name, descriptions, an image and identifiable elements.



Meta Model Element:



Description:

Identifiable element

The identifiable element is designed for tracing over perspectives. It defines the following attributes:

- The *id* represents the identifier. It can be a number, a string or any other value the tooling allows.
- The *category* attribute allows to customly group identifiable elements by strings. Example categories (maybe proposed by the tooling) are:
  - Modeling Goals
  - Sub Goals
  - Marketing Strategies (e.g. "Production + Sales OEM")
  - Parameters + Characteristics (2 Brakes)
  - Chosen E-Scooter values (Speed > 100km/h)
  - Tier 1 specialized part (Microcontroller Z)
  - Mindmap element
  - Technology demand
- The *text* represents the content.
- The optional *value* attribute allows to enhance the element with a value. The value can be used for checking consistency between the identifiable element of the Strategy Perspective and any other element that includes a value or property. Note, that on early strategy considerations values are seldom available.
- The *discussion* and *version* fields enhance the Block description by allowing discussions and version control.

Example: The following three examples represent Identifiable Elements with an id, a category and a text. The optional value is not set and there exists no discussion and version number. (The empty non optional attributes are not shown here).

PMG1 | Modeling Goal | The evaluation of future batterie technologies

PMG2 | Modeling Goal | The evaluation of future microcontroller technologies and their software

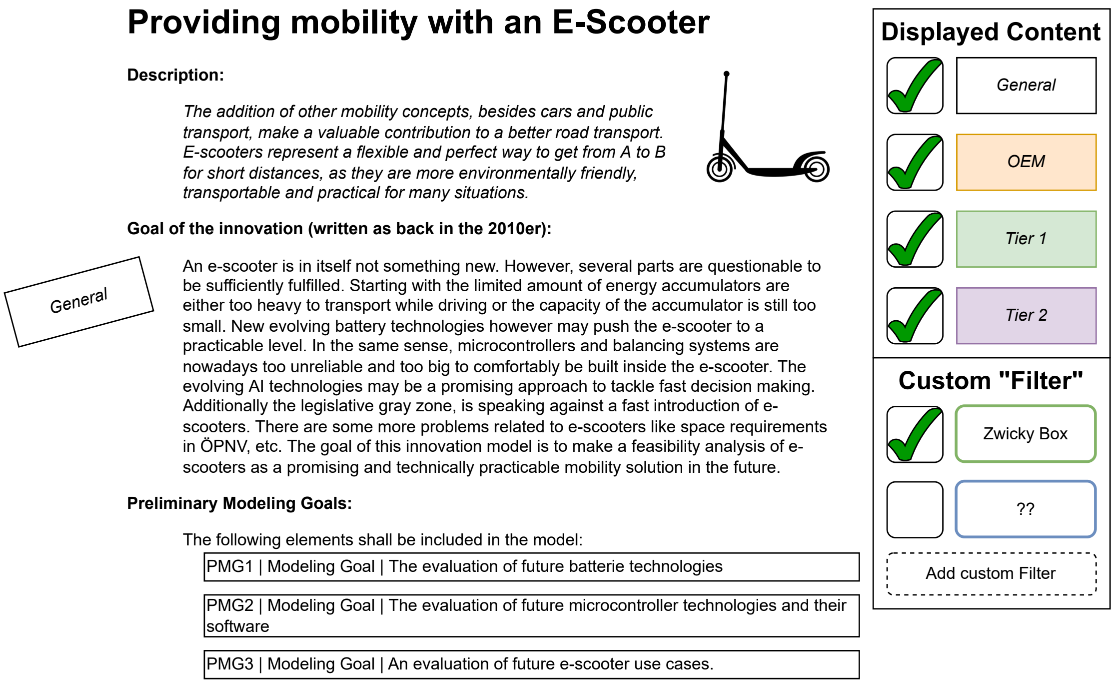
PMG3 | Modeling Goal | An evaluation of future e-scooter use cases.

5.2. E-Scooter example

The example of the Strategy Perspective describes the innovation of "Providing mobility with an e-scooter".

The example is divided into two views: The Strategy Description View describes the innovation textually and graphically from the OEM view, Tier 1 view, Tier 2 view and from a general view. It contains descriptions, goals, business models, aspects, important notes and diagrams from creativity methods. Additionally, some text phrases and goals are made identifiable to allow a mapping and tracing on the other perspectives. The Strategy List View shows the identifiable elements from the Strategy Perspective, that were used in the Strategy Description View. The descriptions are left out. This view is used to focus and remark the identifiable elements.

Strategy Description View



OEM Informations

OEM

OEM Market Segment Analysis:

User Needs Projection:

- The user needs to be able to drive and perform similar tasks to reach its destination.
- The user want to carry stuff alongside.
- The distance reached shall be acceptable high (e.g. 20km)
- The e-scooter shall help optionally on balancing issues.
- (The exact User Needs shall be defined by the requirements engineer)

User Stories: The user want to get from point A to B (short trip) with a comfortable e-scooter. (The exact User Stories shall be defined by the requirements engineer)

Environmental Constraints: From the legal perspective (2010) e-scooters are still in a gray zone. There needs to be legislation done to let them drive on the streets.

Business Opportunities:

Time to Market: Battery technology are trouble some. It is expected though that once the legislation is done, all issues are solved.

Return on Investment: Market pressure is fiercly expected, because the electro mobility technology is well known. However there shall be a good profit as a pioneer.

OEM Business Model:

There are several business models possible. Three of them are as follows:

The easiest business model contains the **production and sales** of the e-scooter itself. This works very well while the market is relatively new, the product is technically superior or the brand itself is an important factor.

Another option contains the **leasing** of the e-scooters. The production superiority plays in the leasing model a less important role. However, higher costs are expected in the product life cycle due to service prividance. A key driver for this model is a good customer retention.

Another option contains the collaboration with the government to provide e-scooter as **part of the ÖPNV**. This option is especially interesting when the local or country wide ÖPNV is free of charge.

Additional OEM Important Aspects:

The OEMs agreed on certain subgoals that they want to be fulfilled by the later innovation modeling. These subgoals shall be tracked in the model, thus they got IDs. However, they are not meant as full fleshed requirements yet. The requirements engineer shall take them as input and create detailed requirements. The subgoals are as follows:

SG 1| Sub Goal | The e-scooter shall be able to transport the user.

SG 2| Sub Goal | The e-scooter shall be comfortable to transport.

SG 3| Sub Goal | The e-scooter shall be able to be parked in any legal location.

SG 4| Sub Goal | The e-scooter shall be available for leasing.

OEM Notes:

Additionally, the OEMs made notes about words they used and what basic conditions they mean. These are not appended with IDs, however they shall be somehow considered

1. **Basic Working Conditions:** Mission areas are different scenarios. The scenarios include individual owned e-scooters, permanently used e-scooters, comfort requisted e-scooters and simple requested e-scooters.
2. An E-Scooter is self-balancing if it is equipped with an integrated electronic balance, propulsion, steering, and deceleration system by which it can maintain itself balance.

Creative idea drawings:

The following diagrams shall provide additional informations and support modeling. The creativity method "Zwicky Box" (also known as "Morphologischer Kasten") is used.

The first step in the creativity method "Zwicky Box"contains the identification of the problem. Here the problem is formulated as the question: "*How could the future mobility look like?*"

The next step contains the collection of parameters, which represent the most important indicators of the mobility concepts. Here, *"Speed", "Travel Distance", "Parking Overhead", "Luggage Space" and "Drive Train"* are chosen.

The third step includes the identification of caracteristics of each parameter. Here several speed values, travel distances, etc are chosen.

This results into the template table below.
The fourth and final step is the creative and exciting one. Based on the template combine the different characteristics into a new mobility concepts. After combining, you may have some thrilling new mobility solutions. Here of couse - next to several other combinations - the e-scooter mobility solution was identified and taken for further consideration.

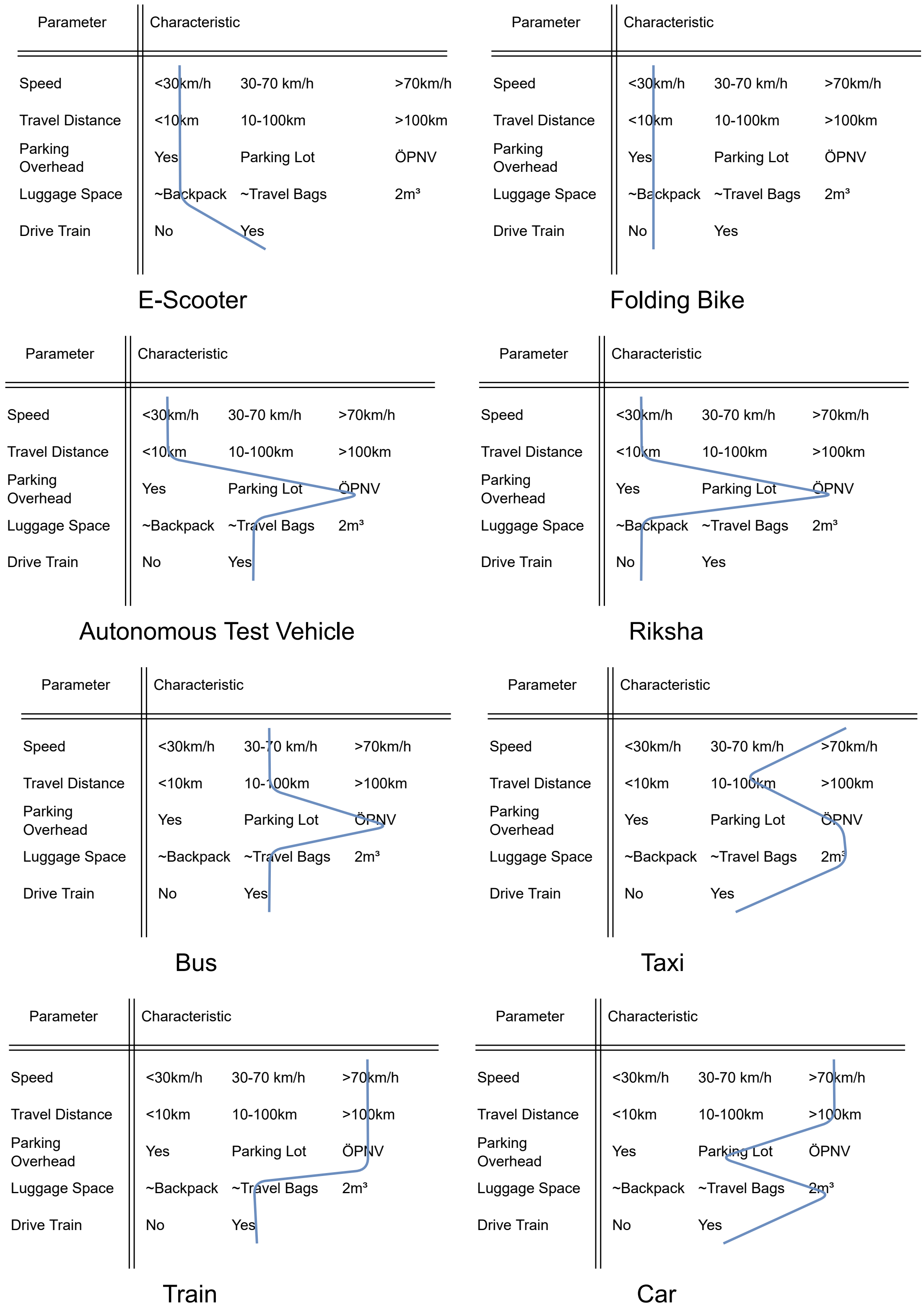
The diagram adds valuable additional information from the OEM perspective for the modeling of the innovation! One information was created as an Identifiable Element for further references:

EP 1| E-Scooter Property | The e-scooter shall be able to reach 20km/h. | Speed > 20km/h

Problem: How could the future mobility look like?

Parameter	Characteristic		
Speed	<30km/h	30-70 km/h	>70km/h
Travel Distance	<10km	10-100km	>100km
Parking Overhead	Yes	Parking Lot	ÖPNV
Luggage Space	~Backpack	~Travel Bags	$2m^3$
Drive Train	No	Yes	

Template


Parameter
Characteristic
Speed
<30km/h
30-70 km/h
>70km/h
Travel Distance
<10km
10-100km
>100km
Parking Overhead
Yes
Parking Lot
ÖPNV
Luggage Space
~Backpack
~Travel Bags
2m³
Drive Train
No
Yes
E-Scooter
Folding Bike
Autonomous Test Vehicle
Riksha
Bus
Taxi
Train
Car

Tier 1 Informations

Tier 1 Business Model:

Tier1

The e-scooter innovation provides several Tier 1 interests:

First and most important, providing e-scooter parts makes the Tier 1 take part of the **e-scooter branch**. In playing part in a new market branch equates to more sales.

Second, **optimizing parts** of the e-scooter as specialized products means building up market advantages, which in turn leads to a higher demand.
Some specialized parts could be:

- fitting light setup
- better drive train, engines, batteries
- fitting electrical brakes
- human interaction displays
- algorithm optimized solutions

Especially the algorithm optimized solutions like balancing units enable Tier 1's to explore new concepts of microelectronics for domain specialized solutions.

Tier 1 Market Segment Analysis:

User Needs Projection:

- The OEM needs some balancing unit,
- The OEM needs some batteries, that have a huge capacity.
- (The remaining User Needs shall be defined by the requirements engineer)

User Stories: The OEM wants to get some e-scooter parts from us.

Environmental Constraints: None worth mentioning

Business Opportunities:

Time to Market: Immediately once the technology is available.

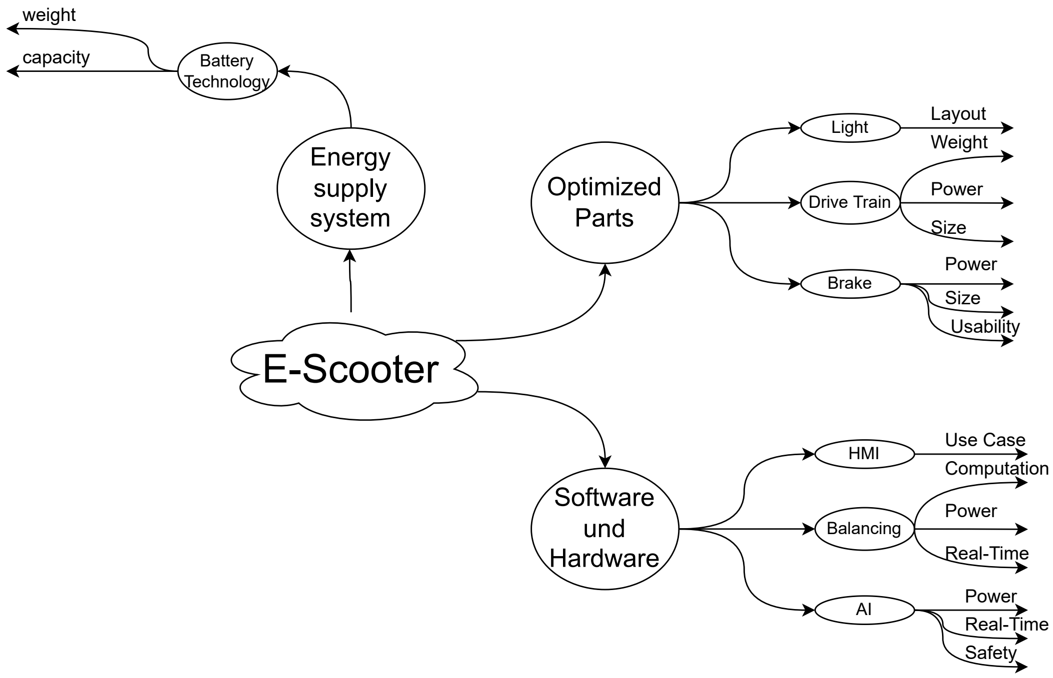
Return on Investment: Market pressure is highly expected. However this presents only a by product for the Tier 1.

Tier 1 Creative idea drawing:

The Tier 1 chooses to draw a mindmap (see the figure below):

The mindmap focuses on the parts of the e-scooter where the Tier 1 see a benefit in providing full solutions (optimized parts), the combination of some algorithm solutions for e-scooter optimized microcontrollers and the energy supply system. Each of the parts have as leaves important properties assigned, which the Tier 1 identifies as the most interesting.

The mindmap adds general value to the information of the OEM in form of the most important factors of each part.


weight
capacity
Battery Technology
Energy supply system
E-Scooter
Optimized Parts
Light
Layout
Weight
Drive Train
Power
Size
Brake
Power
Size
Usability
Software und Hardware
HMI
Use Case
Computation
Balancing
Power
Real-Time
AI
Power
Real-Time
Safety

Tier 2 Informations

Tier 2 Business Model:

The e-scooter innovation provides just a few Tier 2 interests:

First and most important, providing e-scooter technology makes the Tier 2 take part of the **e-scooter branch**. In playing part in a new market branch equates to more sales.

Tier2

Second, **the identification of new mobility solutions and their enablers** allow the Tier 2 to forecast the new technologies needed for developing those solutions, which in turn builds market advantages.
Some technology demands could be:

- new energy supply system concepts
- better sensors
- optimized small micro controller architectures
- human interaction display technology

Especially the energy supply technology is a promising demand for the Tier 2's.
That considered, there is no disruptive behavior recognizable for the Tier 2's.

Tier 2 Market Segment Analysis:

User Needs Projection:

- The OEM and Tier 1 needs batteries, that have a huge capacity.
- (The remaining User Needs shall be defined by the requirements engineer)

User Stories: The OEM and Tier 1 wants to get some better battiry technology from us.

Environmental Constraints: None worth mentioning

Business Opportunities:

Time to Market: Immediately once the technology is available.

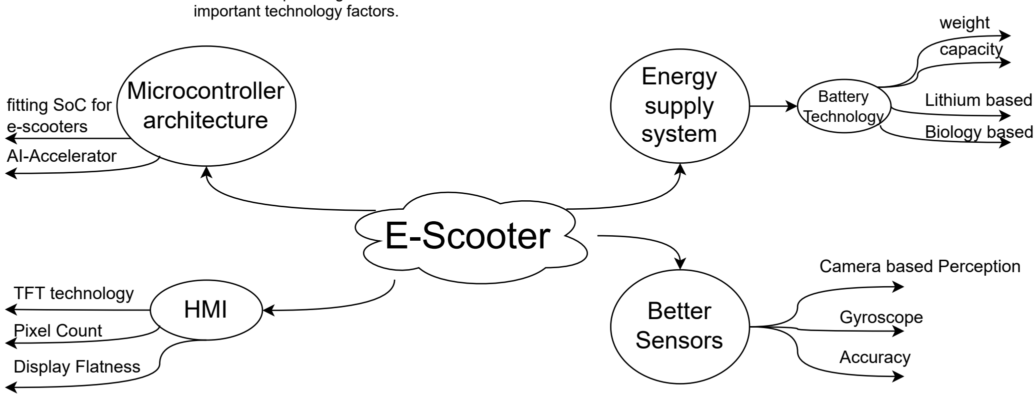
Return on Investment: Market pressure is expected once other Tier 2 got similar well technologies. However there shall be a good profit as a pioneer.

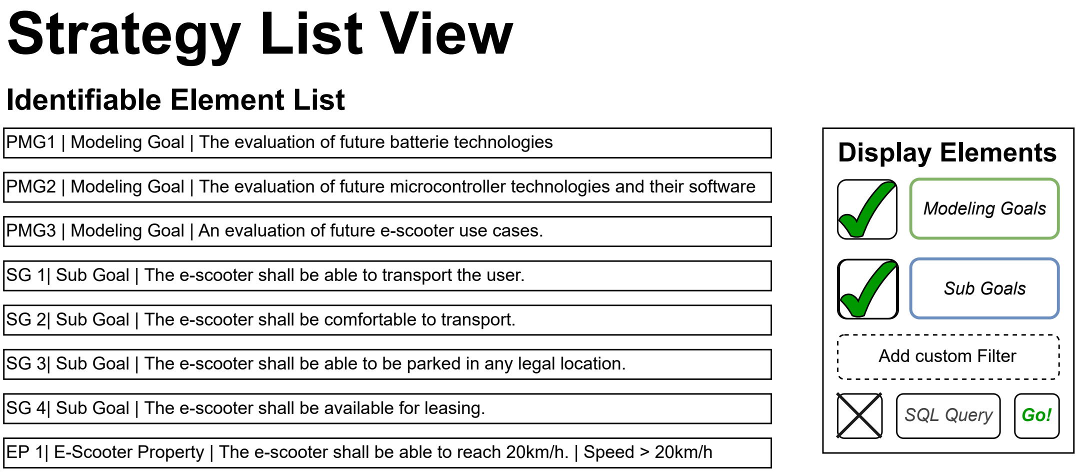
Tier 2 Creative idea drawing:

The Tier 2 chooses to draw a mindmap (see the figure below):

The mindmap focuses on the 4 identified technological interests of the e-scooter, namely the energy supply system, the sensors, the microcintroller architecture and the human interface.
Each of the parts have as leaves important properties and technologies assigned, which the Tier 2 identifies as the most interesting.

The mindmap adds general value to the information of the OEM and Tier 1 in form of the important technology factors.





5.3. Strategy Perspective: Strengths and Limitations

The Strategy Perspective is kept abstract on purpose. It contains (mostly) non formal descriptions making it easy to kick start the Strategy Perspective by directly starting with the creative methods results. Additionally, the Strategy Perspective does not restrict IMoG to use any specific creativity methodology. While the burden to choose a creativity methodology for an innovation is shifted to the user, the interchangeability of taking a proven methodology for the stakeholders instead of a predefined methodology is considered an advantage. These decisions to focus on abstract and interchangeable modeling make the Strategy Perspective simple to model and visualize. On the other hand, the abstract and informal model of the Strategy Perspective builds the basis for basic analysis other than tracing. This is not per se a con but rather a limitation that was traded in for flexibility through informality.

5.4. Strategy Perspective FAQ

The FAQ splits up into the categories:

1. Questions and answers about the general Strategy Perspective model elements

2. Questions and answers about the relation of the Strategy Perspective to other perspectives

Strategy model elements:

What Identifiers and Categories shall be chosen for the Identifiable Elements?

Both Identifier and Categories shall be defined as best fiiting to the Innovation. The Identifier and Category specification does not constraint the definition by more than being a number or string type.

Example: The Innovation "Providing mobility with an E-Scooter" may have the following Identifiable elements. The first element is of the "Sub Goal" category and have as an Identifier the initiales "Sg" with a number attached. Similar to the first elment is the second element build. Here the category is more innovation specific ("E-Scooter Property") and has as an Identifier "EP 1".

SG 1| Sub Goal | The e-scooter shall be able to transport the user.

EP 1| E-Scooter Property | The e-scooter shall be able to fly.

When to use values for the Identifiable elements?

Values shall be used when the strategy information can be represented as such and represents a basic condition that shall be automatically evaluated against later defined model elements.

Example: EP 1 represents a basic condition that shall be fulfilled by the later e-scooter models. Thus it has a value attached.
EP 2 represents only a guidance for the e-scooter distance reached. Thus it does not contain a value.

EP 1| E-Scooter Property | The e-scooter shall be able to reach 20km/h. | Speed > 20km/h

EP 2| E-Scooter Property | The e-scooter distance reached shall be around 30km. In general, the more the better. However for small e-scooter models, the distance may fall below 30km.

Relations to the other perspectives:

How does the Strategy Perspective relate to other Perspectives?

The Strategy Perspecive represents the first thoughts on the innovation and breaks them down into an innovation description and a set of Identifable Elements. The Strategy Perspective builds the basis of understanding. The modeling activities in the other Perspectives build on this basis. The Identifiable Elements are designed to transport valuable information to the modeling elements of the other Perspectives by referencing and tracing the Identifiable Elements onto the other Perspective Elements.

Example: SG 1 describes a function in an abstract way. This function shall be considered in the modeling activities. Thus SG 1 was created as an Identifiable Element for tracing and referencing. The Feature Tree Modeler creates the Feature "Driving" and references the Identifiable Element onto it to show that the strategy request is fulfilled.

SG 1| Sub Goal | The e-scooter shall be able to transport the user. → <<Feature>> Driving

Similar to SG 1, EP 1 describes a requirement that is translated and mapped on a requirement by a requirement engineer. The reference on the requirement represents that the requirement engineer has fulfilled the request from the strategy department.
.

EP 1| E-Scooter Property | The e-scooter shall be able to reach 20km/h. | Speed > 20km/h

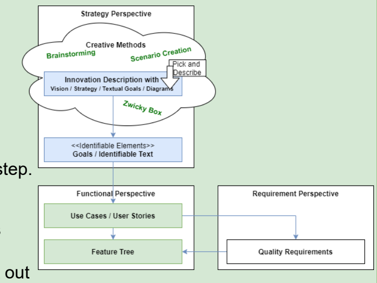
↓

ID: R1| Priority: 1 | Name: Speed | Text: The e-scooter speed shall reach 20km/h on flat ground with a normal person and no extra luggage.

A general problem is the jump from creative methods to the modeling viewpoints. How does this jump from the Strategy Perspective to the other Perspectives shall be done?

The jump from the creative methods to the modeling viewpoints is done by some additional steps in between (For a visual representation see the Working Process under IMoG.drawio - "IMoG Working Process" tab):

1. The results of the creativity methods are integrated into the Innovation description.
2. From there on the Traceable Elements are defined.
3. Creating abstract User Stories and Use Cases have been identified as a reasonable step.
4. From there on, the Feature Tree and Requirements can be identified.



Example: The OEM used a "Zwicky Box" as a creativity method. The "Zwicky Box" result was integrated into the Innovation description. The Speed parameter from the "Zwicky Box" was identified as one Identifiable Element. Then the User Stories and Use Cases where build (left out here) and the model activities with Feature Trees and Requirements were started. As a result, the requirement R1 resulted. For further details, see the Strategy Perspective example as well as the Functional Perspective Example and Quality Perspective Example in their respective draw.io files.

EP 1| E-Scooter Property | The e-scooter shall be able to reach 20km/h. | Speed > 20km/h

↓

ID: R1| Priority: 1 | Name: Speed | Text: The e-scooter speed shall reach 20km/h on flat ground with a normal person and no extra luggage.

6. Functional Perspective

The Functional Perspective is the second perspective in IMoG and describes the features (end-user visible characteristics) and functions (traceable tasks or actions that a system shall perform) of the innovation (see Figure 6.1). The purpose of the Functional Perspective is to capture the features and functions required for the innovation before diving deep into solutions. The Functional Perspective is thus used in the early phases of innovation modeling and is part of capturing the problem space.

The Functional Perspective model bases on the well-known feature trees [9]. Feature Trees are a subclass of Feature Models, which restrict themselves to tree structures. This restric-

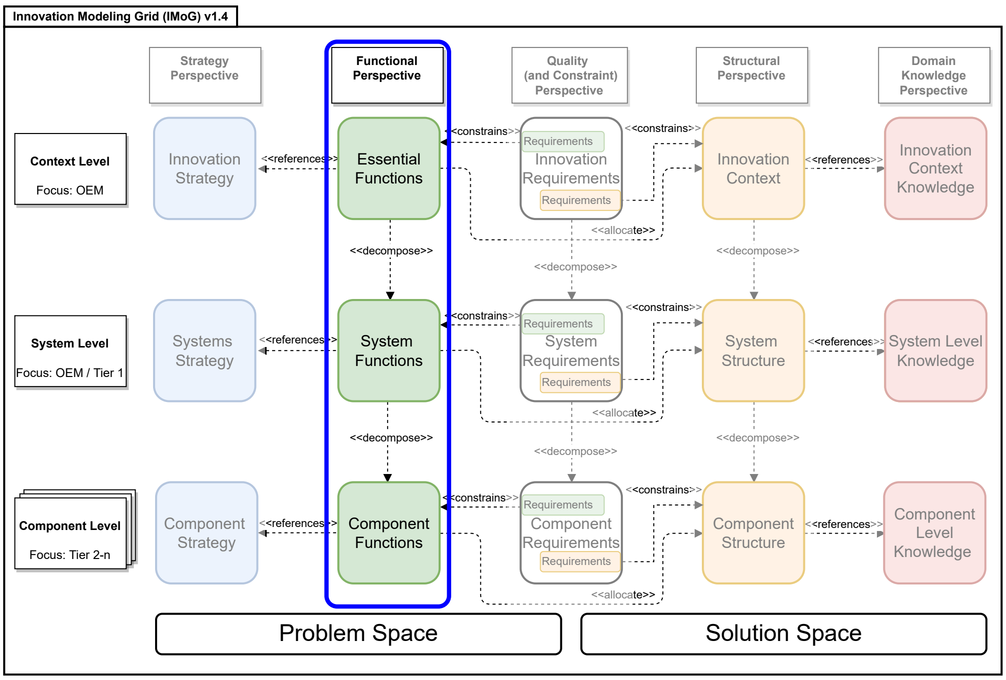


Figure 6.1.: **Location of the Functional Perspective in IMoG**

tion reduces the complexity and increases the ability to maintain an overview. Feature Trees put a high focus on variability and are often used in product line engineering. The high-level innovation modeling in IMoG on the other side, does not focus too much on variability and the modeling of details are expected to be after the innovation modeling, being part of the subsequent design and engineering phases. That noted, having the ability to model a bit of variability may support in expressing innovation dependencies.

IMoG also makes some extensions to feature trees, however these are mostly of cosmetic nature and can be directly translated to the default feature trees. This translation allows the use of available Feature Tree analysis tools.

The chapter is structured as followed: In Section 6.1 the meta model and its model elements are presented. In Section 6.2 an example of the Functional Perspective is given. The strengths and limitations of the Functional Perspective are discussed in Section 6.3. A FAQ finalizes the description in Section 6.4.

6.1. Model elements

The meta model (see Figure 6.2) builds on the FODA (Feature Tree model and Feature Diagram model, [9]) and includes all relevant concepts of FODA. The meta model has - as the top level unit of the Functional Perspective - the Functional Perspective Model. It contains a set of Blocks (FP) with Relations (FP) between them. Additionally Groups of Blocks (FP) are contained. Blocks (FP) represent the basic units of functionality known from Feature Trees and are extended with several attributes, Blocktypes and an Abstraction Level (which can be either Context Level, System Level, Component Level or of Type custom Abstraction Level). A detailed description of the attributes can be found in the Block (FP) description. Unlike the original FODA model definition [9], where functions are explicitely not part of the Feature Tree, Blocks (FP) are here further categorized into features and functions for specifying what explicitly a 'Block' means. A feature represents a logical unit of behavior that is too abstract to be mapped on structural solutions, while a function represents a mappable unit onto the structural solutions. For a flexible mapping, each feature shall have a set of functions. The ability to model functions allows the seamless tracing from features onto functions and later onto solutions on the Structural Perspective.

Several types of Relations between Blocks (FP) can be made, including and extending the typical relations from Feature Trees. The Relations (FP) split up into Parent-Child Relations, Constraint Relations and Variation Point Relations. The Parent-Child Relations include the Alternative-Relation, the Or-Relation, the Mandatory-Relation and the Optional-Relation

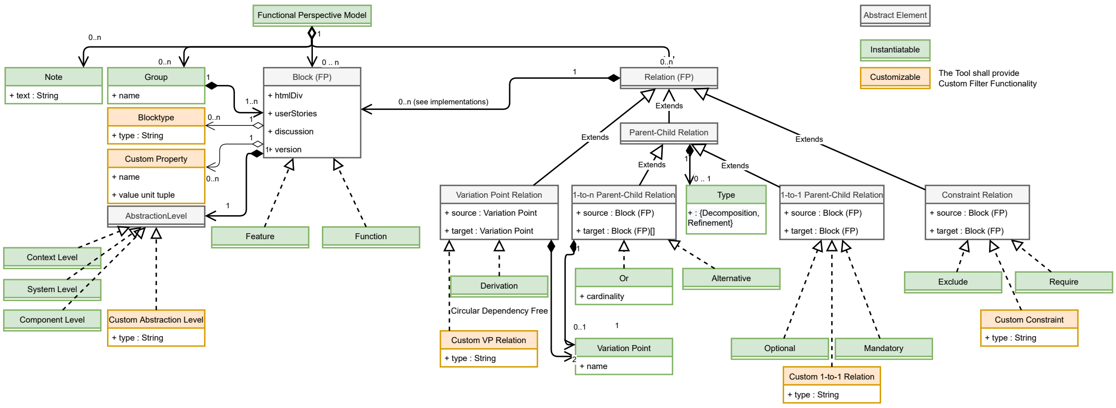


Figure 6.2.: **The model elements of the Functional Perspective.**

known from Feature Trees. Additionally the Parent-Child Relations can be optionally labeled as 'Refinement' or 'Decomposition' or a custom Parent-Child Relation can be used. The Constraint Relations include the known extensions to Feature Trees to express restrictions on configurations: The Require and Exclude relations. If not enough, custom Constraint Relations can be added. The last extension made to Relations (FP) are the Variation Point Derivation Relation to represent similar alternative choices. The following model elements Section will dive into more details.

Meta Model Element:

Functional Perspective Model

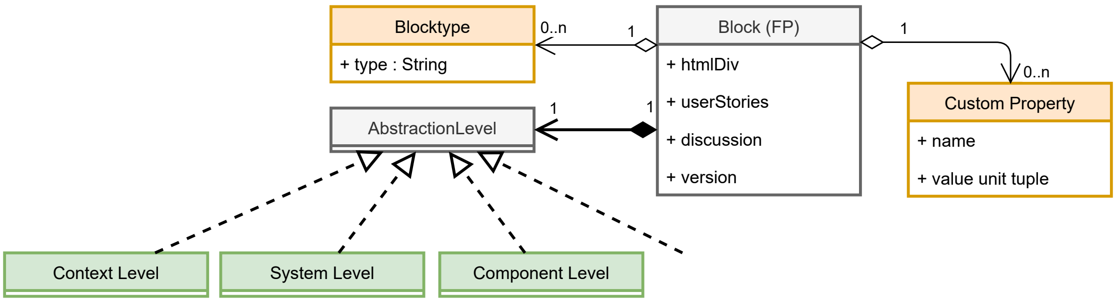
Description:

Functional Perspective Model The *Functional Perspective Model* is the diagram of the Functional Perspective of an innovation. It contains all model elements of the Functional Perspective.

Example: A full Functional Perspective Model example is shown in Section 6.2.

Meta Model Base

Meta Model Element:

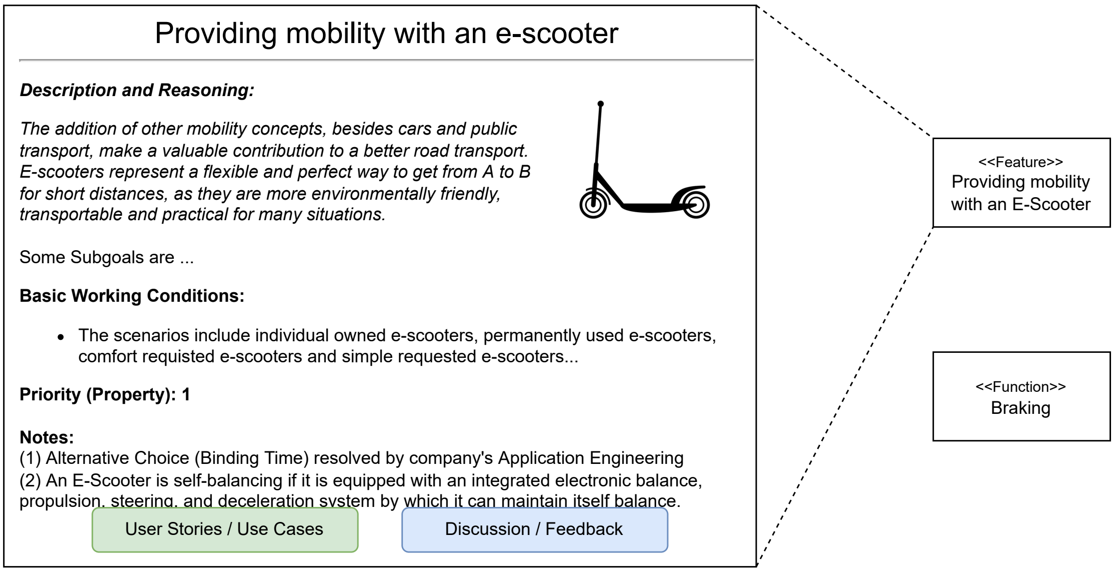


Description:

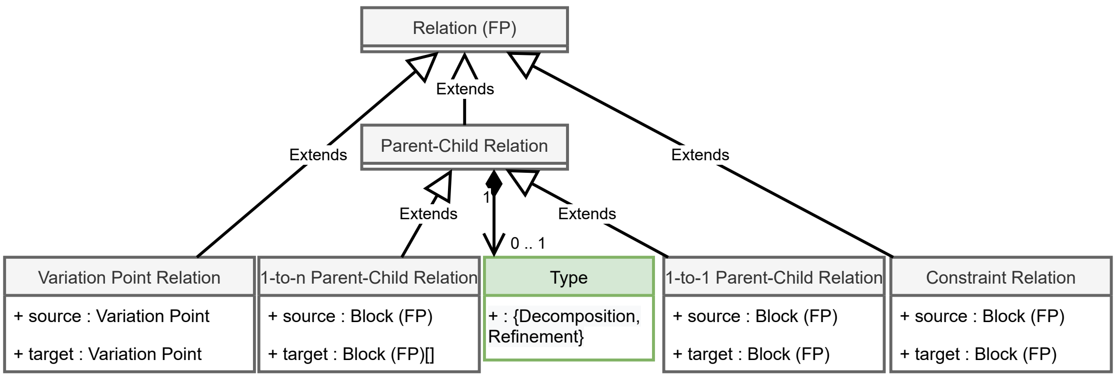
Block (FP) (Read: Block on the Functional Perspective)
The *Block (FP)* is the abstract Block element of the Functional Perspective which is implemented by *Features* and *Functions*. It defines the attributes of the Blocks:

- The Abstraction Level of the Block defines the level of abstraction the Block represents. It can be either *Context Level, System Level, Component Level* or from the type *Custom Abstraction Level*.
- An optional *Custom Block Type* can refine the category of the Block further.
- The *HTMLDiv* represents the description of the Block to solve the problem of lack of clarity by adding information next to Feature Trees. The description shall answer shortly "What the Block shall provide?", the Reasoning behind the Block and its basic conditions to work and if the Block has alternative choices, then additionally the binding time of the choice. The binding time being part of the *HTMLDiv* is considered enough here. It does not have to be a Block property like proposed in the original Feature Tree publication [9]. There is no template needed. Images or drafts provide valuable information.
- Optional Custom Block Properties can be defined for additional tooling analysis including filtering and consistency checks.
- The *User Stories, discussion* and *version* enhance the Block description.

Example: The following block shows an example of a Block (FP) with its attributes.



Meta Model Element:



Description:

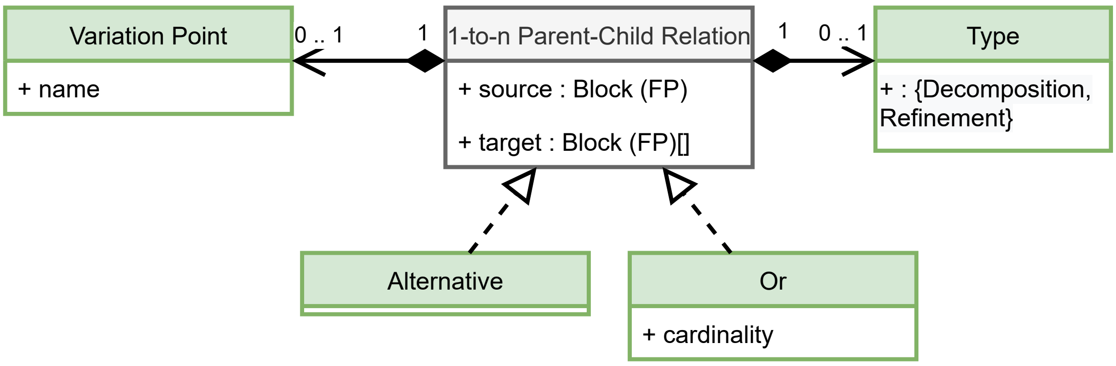
Relation (FP) **(Read: Relation on the Functional Perspective)**
The abstract *Relation (FP)* describe relations between *Blocks (FP)* or respectively *Variation Points* on the Functional Perspective. *Relations* are further categorized by

- *1-to-1 Variation Point Relations*
- *Parent-Child Relations* between Blocks (FP) of either category *1-to-n Parent-Child* relation or category *1-to-1 Parent-Child* relation. The *Parent-Child Relations* can be specified by an optional type, which can be either of value *Decomposition* or *Refinement*. Note that the additional stereotypes are similar to the relation differentiation *{Specialization (Refinement), Decomposition, Parametrization}* outside the model definition in the original Feature Tree publication [9].
- 1-to1 Constraint Relations between Blocks (FP)

Each of the Relations are described in more detail on its own.

Example: An example is shown for each relation under their respective description.

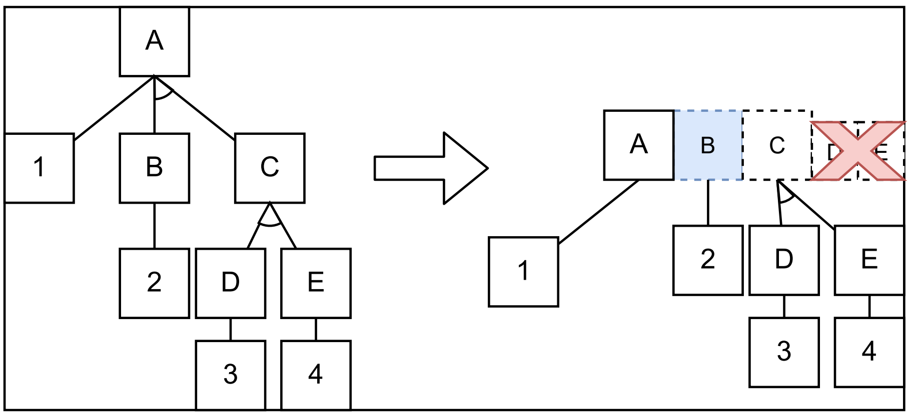
Meta Model Element:



Description:

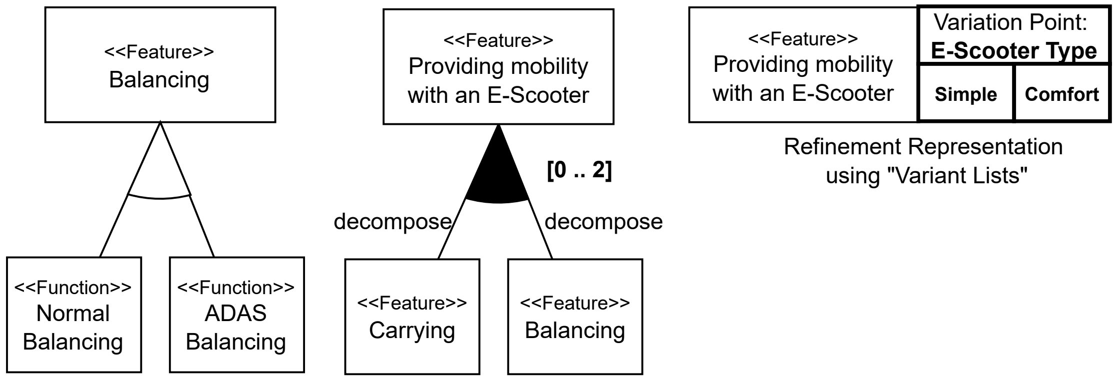
1-to-n Parent-Child Relations
The *1-to-n Parent-Child Relations* build the Foundation for the well known *Alternative* and *Or* Relation from Feature Trees. *Alternative* and *Or* Relations can connect one parent Block (FP) with multiple child Blocks (FP). They are used to describe Decomposition or Refinement Choices of the Parent Block. Additionally, 1-to-n Parent-Child Relations can own Variation Points. *Variation Points* represent the description of the choice or variability and are written optionally. For Refinement Relations, the special Refinement Representation using *Variant Lists* can be used. However, the normal Representation and the Variant List Refinement Representation are both valid.

Alternatives Representation

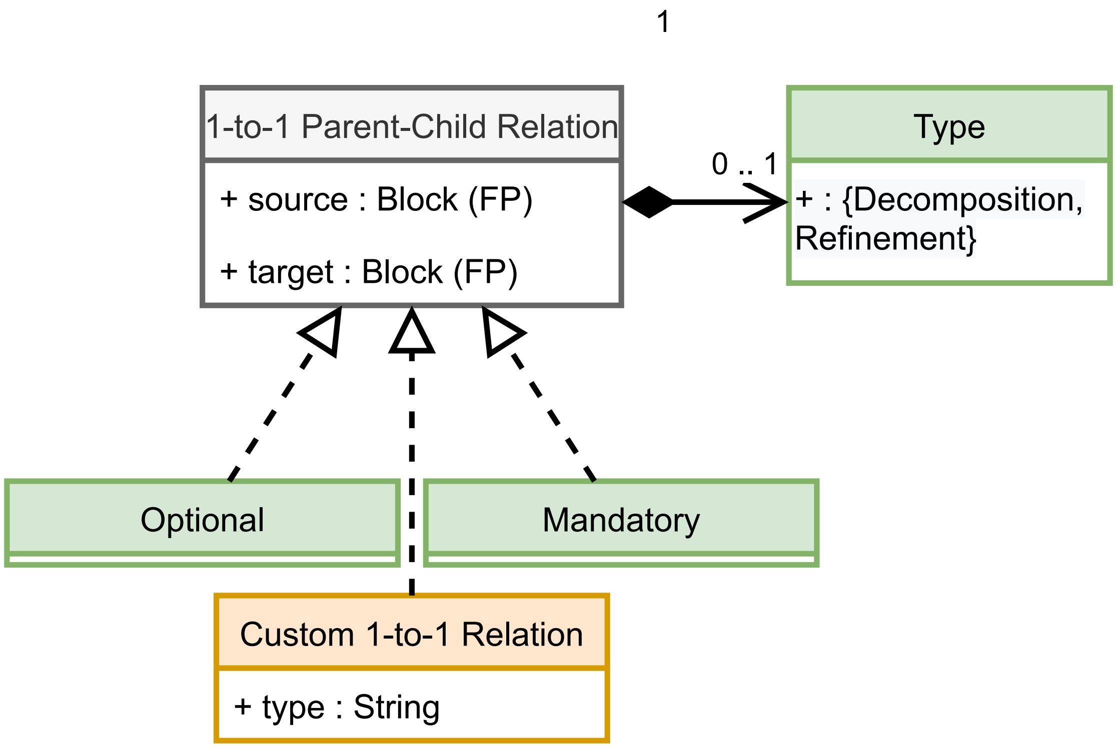


Both Representations are allowed! The Refinement Representation is just useful for depths up to 1 of Alternatives, such that any further Refinement alternative depth is represented as a normal Feature Tree Alternative! The Gain of the Refinement Representation lies in the compact representation and is thus enabled for usage in IMoG Feature Trees!

Example:



Meta Model Element:



Description:

1-to-1 Parent-Child Relation
The *1-to-1 Parent-Child* relation connects two Blocks (FP) with each other. Up to now there are three relations defined. The *Optional* relation, the *Mandatory* relation and the *Custom 1-to-1* relation. The *Custom 1-to-1* relation allows to describe additional relations between Blocks (FP). Custom 1-to-1 relations are not analyzed. The other relations are described under their respective description. As before mentioned, the *1-to-1 Parent-Child* relations can be specified by an optional type, which can be either of value *Decomposition* or *Refinement*.

Example: An example is shown for each relation under their respective description.

Model Elements

Model Elements:
Block Types:

1. Features
2. Functions

Relation Types:

1. And Relation (Mandatory Sub-Features) with
   - Refinement / Decomposition Relations
2. Optional Features (+ Optional Relations) with
   - Refinement / Decomposition Relations
3. Xor Relation (Alternative or Variant) with
   - Refinement / Decomposition Relations
   - Variation Point
   - Variant List (Refinement Representation)
   - Cyclefree Variation Point Selection Derivation
4. Or Relation with Cardinalities with
   - Refinement / Decomposition Relations
5. Constraint Relations (Require / Exclude)
   - Constraint Grouping

Miscellaneous:

1. Notes

In the following the model elements are introduced. First the two Block Types are introduced and then the relations between the Blocks are presented. Lastly the "Notes" element is introduced.

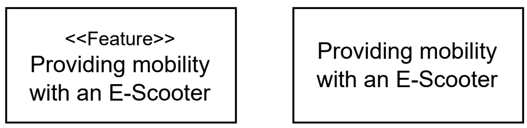
Block Types

Meta Model Element:

Description:

Feature
The *Feature* is a Block (FP) of the Functional Perspective and is next to *Functions* one of the two existing Block (FP) elements. A Feature defines a logical unit of behavior. It semantics originates from Feature Models [9]. However, Features are additionally understood here as actionable, uncountable items and shall be described like an activity. A Feature is considered as too abstract to be mapped on structural solutions. The Stereotype «Feature» can be omitted.

Example:

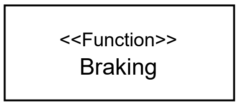


Meta Model Element:

Description:

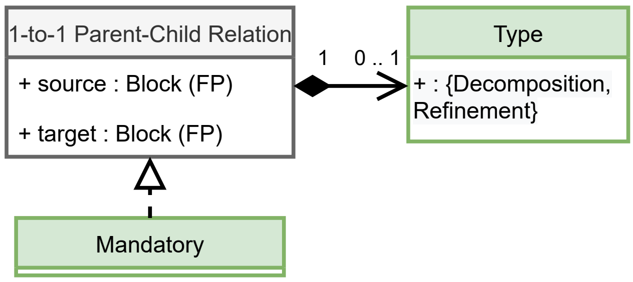
> **Feature**
> The *Function* is a Block (FP) of the Functional Perspective and is next to *Features* one of the two existing Block (FP) elements. A Function defines a logical unit of behavior that shall be implemented by structural components. Functions are understood here as actionable, uncountable items. and shall be described like an activity.

Example:



Relations

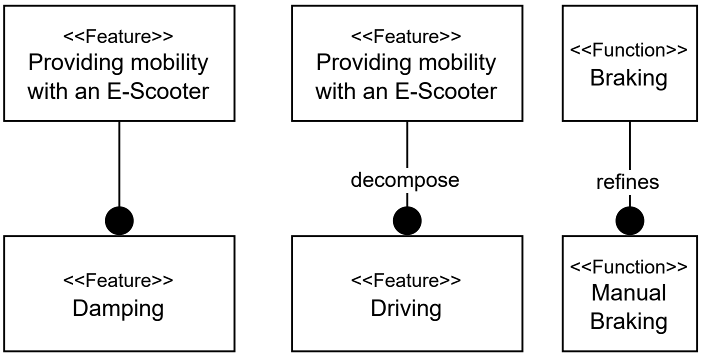
Meta Model Element:



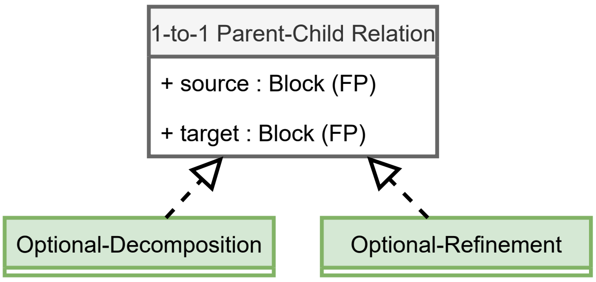
Description:

> **Mandatory Relation**
> The *Mandatory* relation connects one parent (always the top one) Block (FP) with a child (always the bottom one) Block (FP). The *Mandatory* relation describes that the child Block must be provided once the parent Block is part of the configuration. The relation has two additional stereotyped forms: The *Mandatory-Decomposition* relation and the *Mandatory-Refinement* relation. The *Mandatory-Decomposition* relation describes, that the child Block is a decomposed element of the parent Block. The *Mandatory-Refinement* relation on the other hand describes that the child Block is a refinement of the parent Block. If the Stereotype is omitted, then the *Mandatory* relation is interpreted as a *Mandatory-Decomposition* relation.

Example:



Meta Model Element:

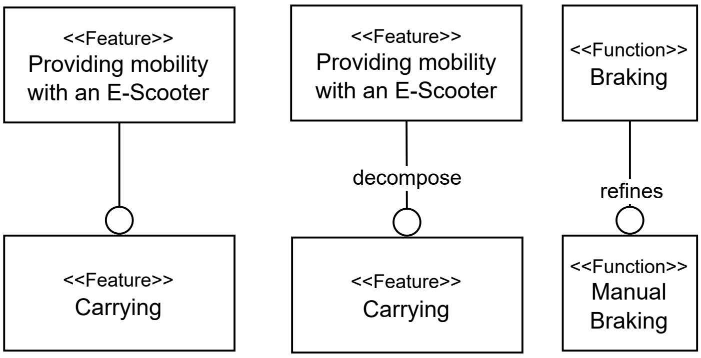


Description:

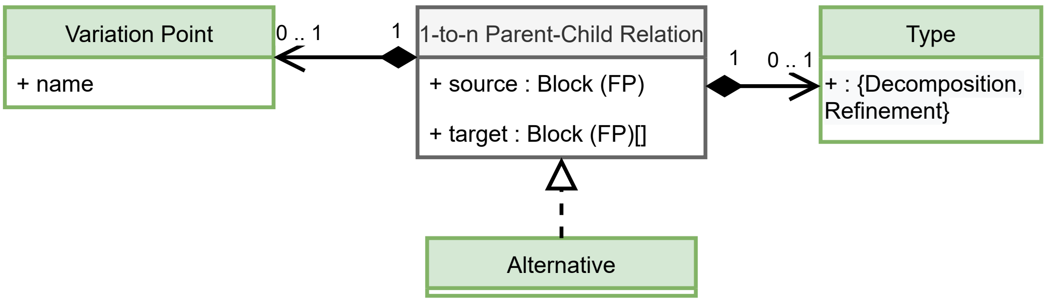
Optional Relation

The *Optional* relation connects one parent (always the top one) Block (FP) with a child (always the bottom one) Block (FP). The *Optional* relation describes that the child Block may be optionally provided once the parent Block is part of the configuration. The relation has two additional stereotyped forms: The *Optional-Decomposition* relation and the *Optional-Refinement* relation. The *Optional-Decomposition* relation describes, that the child Block is a decomposed element of the parent Block. The *Optional-Refinement* relation on the other hand describes that the child Block is a refinement of the parent Block. If the Stereotype is omitted, then the *Optional* relation has no additional semantics assigned.

Example:



Meta Model Element:



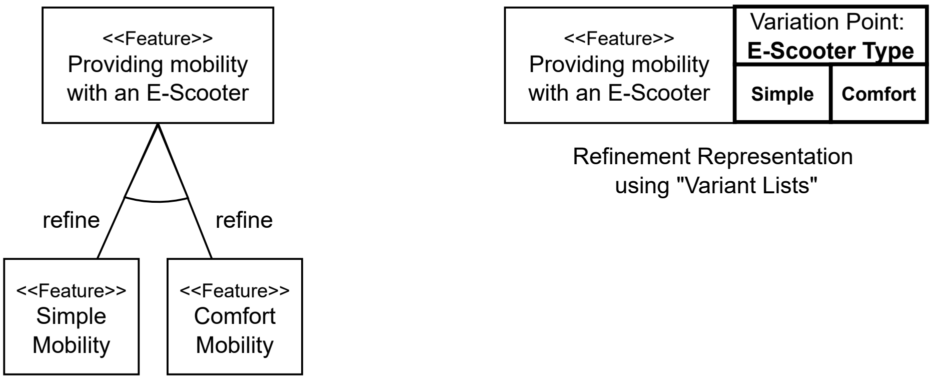
Description:

Alternatives and Variation Points

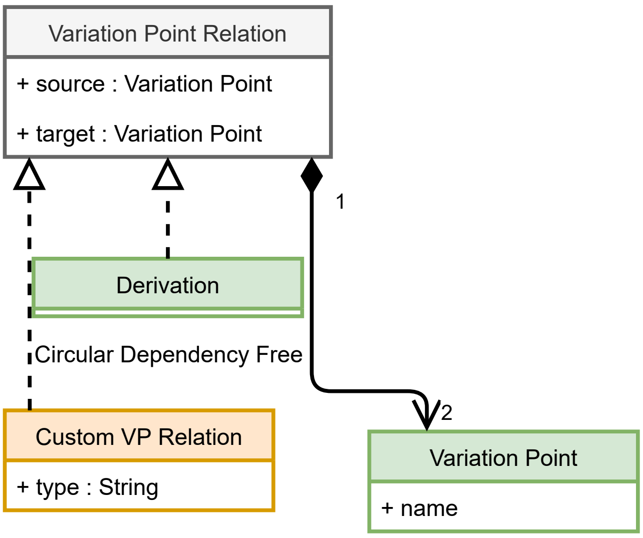
The *Alternative* relation is a well known element from Feature Trees. It represents a choice of Blocks (FP) from which exactly one option can be taken. *Variation Points* represent the description of the choice or variability and are written optionally with the *Alternatives*. *Alternatives* are used to describe Decomposition or Refinement relations of the parent Block. For Refinement relations, the special Refinement Representation using 'Variant Lists' can be used. However, the normal representation and the Variant List Refinement representation are both valid.

As mentioned before, the *Alternatives Relations* can be specified by an optional type, which can be either of value *Decomposition* or *Refinement*.

Example:



Meta Model Element:

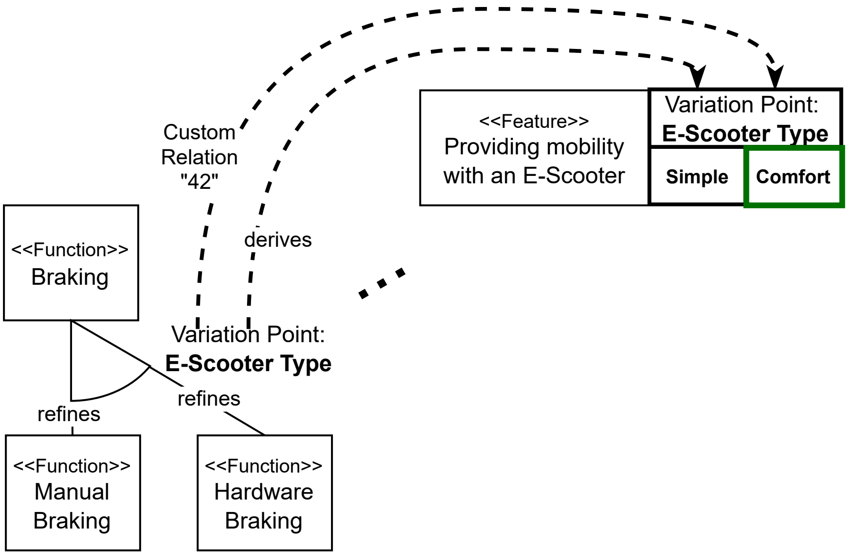


Description:

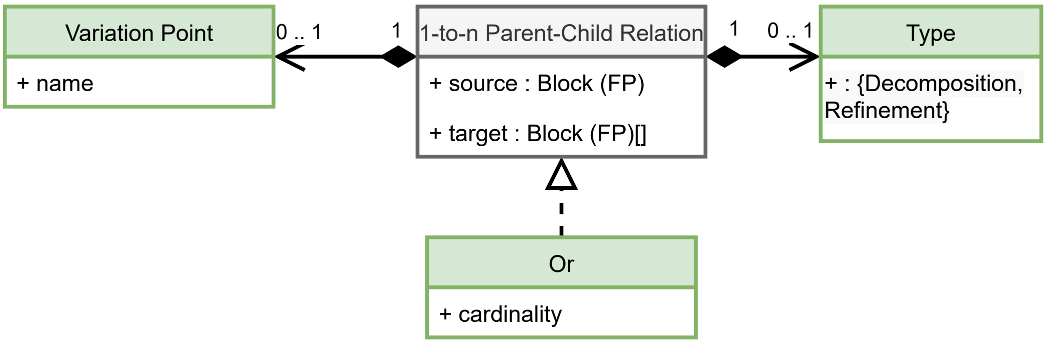
Variation Point Relation

The *Variation Point* relation connects two *Variation Points* with each other. Up to now there are only the *Derivation* relation and the *Custom VP* relation defined. The *Derivation* relation represents the derivation of the choice of the target *Variation Point* given that the choices are the same. The *Derivation* relation can only derive from Variation Points with at least a level higher than the trees depth. This avoids cycles. With this defined, some global configuration can be defined and locally used on the fitting places. The *Derivation* relation can replace several *Require* relations. The *Custom VP* relation allows to describe additional relations between Variation Points. Custom VP Relations are not analyzed.

Example:



Meta Model Element:

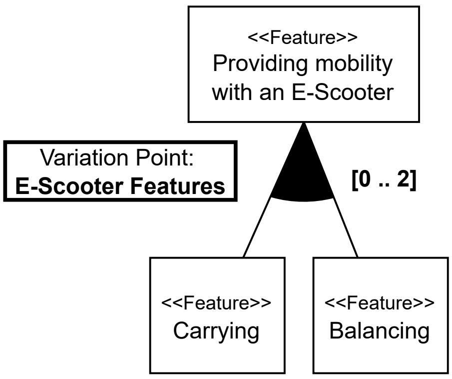


Description:

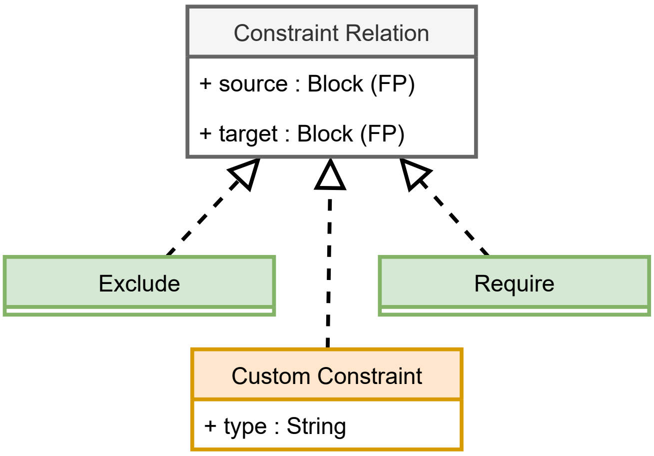
Or Relation with Cardinality

The *Or* relation is a well known element from Feature Trees. The *Or* relation represents a choice of Blocks (FP) from which one or more options can be taken. The *Or* relation is a generalization of *Alternatives*. The *cardinality* describes how many Blocks can be chosen to create a valid configuration. *Variation Points* represent the description of the choice or variability and are written optionally with the Or relations. As mentioned before, the Or relations can be specified by an optional type, which can be either of value *Decomposition* or *Refinement*. Or relations are mostly used to describe *Decomposition* relations of the parent Block. *Refinement* relations need to have a *cardinality* of [1,1] to be valid thus the *Alternative* relation is used for them instead.

Example:



Meta Model Element:

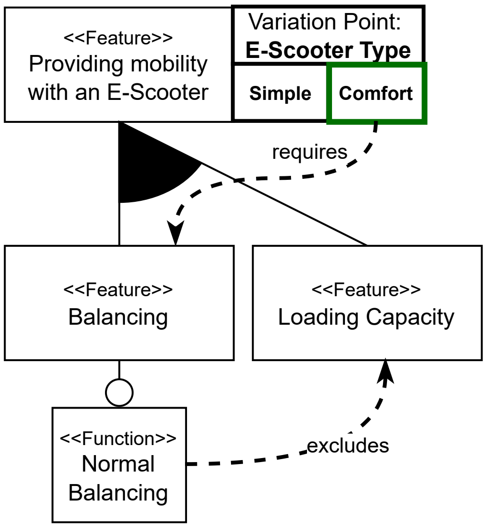


Description:

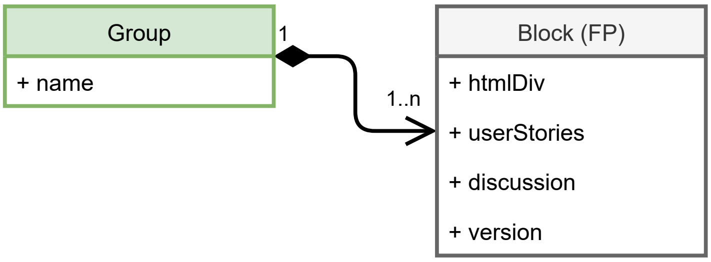
Constraint Relation
The *Constraint* relation connects one Block (FP) with any other (even non child) Block (FP). The *Constraint* relation describes a restriction to the space of configurations and thus how the constraint Block relates to the other Block if the other Block is part of the configuration. There are currently three Constraint Relation Types:

- The *Require* relation A→B specifies that if A is part of the configuration, then B must be part of the configuration too.
- The *Exclude* relation A→B specifies that if A is part of the configuration, then B is not allowed to be part of the configuration.
- The *Custom Constraint* allows to describe additional constraints between Blocks (FP) and own a custom type. *Custom Constraints* are not analysed.

Example:



Meta Model Element:

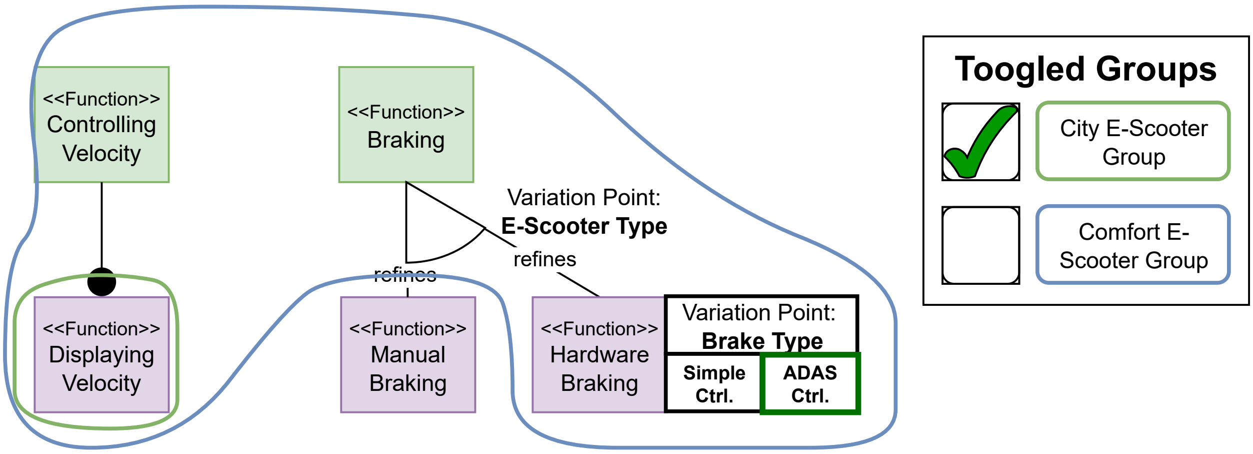


Description:

Grouping

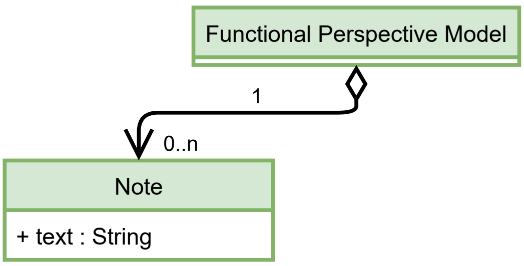
The *Grouping* is an additional usability feature. A *Group* represents a set of together belonging Blocks (FP). If one Block of a Group is part of the configuration, then every other Block in the Group must be part of the configuration too. A Group can be rewritten as *Require* relations between every two members of the Group. With one major difference: The Groups shall be toggable in the tool before the analysis is started, as such constraint-set is typically only wanted in experiments of configurations with the Feature Tree. The toggability in the tool represents a preprocess before the analysis and does not increase the expressiveness from Feature Trees nor does it increase the size of the problem of the analysis.

Example:



Miscellaneous

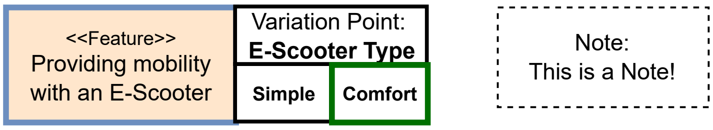
Meta Model Element:



Description:

> **Note**
> The *Note* can be used to add information to the model that can not or should not be modeled. Notes should be used sparsely!

Example:



6.2. E-Scooter example

The example of the Functional Perspective describes the features and functions identified for the innovation of "Providing mobility with an e-scooter".

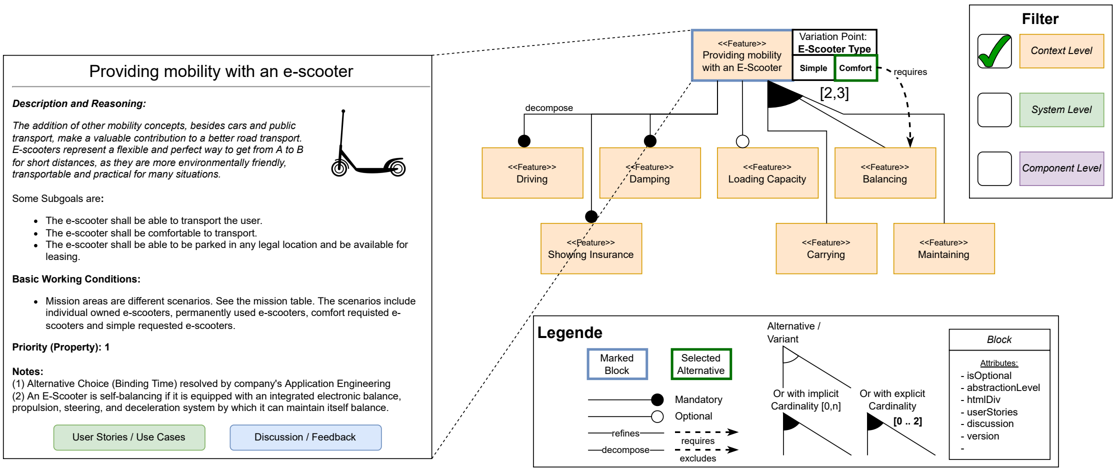


Figure 6.3.: **The identified features and functions for the innovation of "Providing mobility with an e-scooter".**

Figure 6.3 shows the feature model including the context level only. The root feature "Providing mobility with an e-scooter" is marked here, showing the description and properties of the root block. It includes the reasoning behind this innovation, some goals from the strategy perspective, some basic working conditions, notes as well as references to the use cases and user stories. The root feature has three mandatory subfeatures, the "Driving" feature, the "Damping" feature and the "Showing Insurance" feature, and one optional "Loading Capacity" feature. The root feature has a variation point focusing on the type of the e-scooter: either a simple and cheap version of the e-scooter or a comfort version. Note, that the variation point can be represented like an Alternative relation with a name. Finally, the e-scooter has three features as a choice, from which two must be at least be taken. This choice includes the "Carrying" feature, the "Balancing" feature and the

"Maintaining" feature. The choice is more for showing a complete example than having a decent reasoning behind the choice.

Figure 6.4 now also includes the system level and component level (shown in green blocks and purple blocks respectively). This figure shows also constraint relations, grouping and variation point relations.

6.3. Functional Perspective: Strengths and Limitations

The Functional Perspective targets the modeling of the problem space basing on the well known feature trees. This tailoring on features and functions from the problem space alone makes it easy to apply to any innovation. The well known basis on feature trees makes the Functional Perspective straightforward to use for any experience modeler. The basis on feature trees also enables to use the tooling and analyses from feature trees for the Functional Perspective. The extensions provided in the Functional Perspective are for the purpose of adding further information, for filtering and for a better usability and are of cosmetic nature only. The distinction between *Features* and *Functions* is well used in the automotive domain and thus supports well the domain of this project. While these extensions do in fact represent a small learning overhead, the overhead is reasonable. These extensions can be translated directly into basic feature trees. That noted, a feature model expert can model the Functional Perspective without knowing about the extensions. Feature trees are also great at modeling a mix of *Decomposition* and *Refinement* in one model. Additionally, the Functional Perspective can be constrained via Requirements from the Quality Perspective, leading to a very sophisticated model.

As limitation, the Functional Perspective is not designed to capture early structure and interfaces between features. However, this is also good, because it stops the modeler from thinking too much about solutions and their interfaces. It is also well known that feature models can grow very large due to the vast amount of variants, however in the context of abstract innovation modeling, these models should be manageable. Also worth discussing is the purpose of feature models as a basis for the Functional Perspective. Feature models are made for modeling variability and often used in product lifecycle management. Innovation modeling on the other side is abstract and has not too much variability to model. While this clash of purposes may be debatable, the IMoG applications showed that the Functional Perspective is good to use.

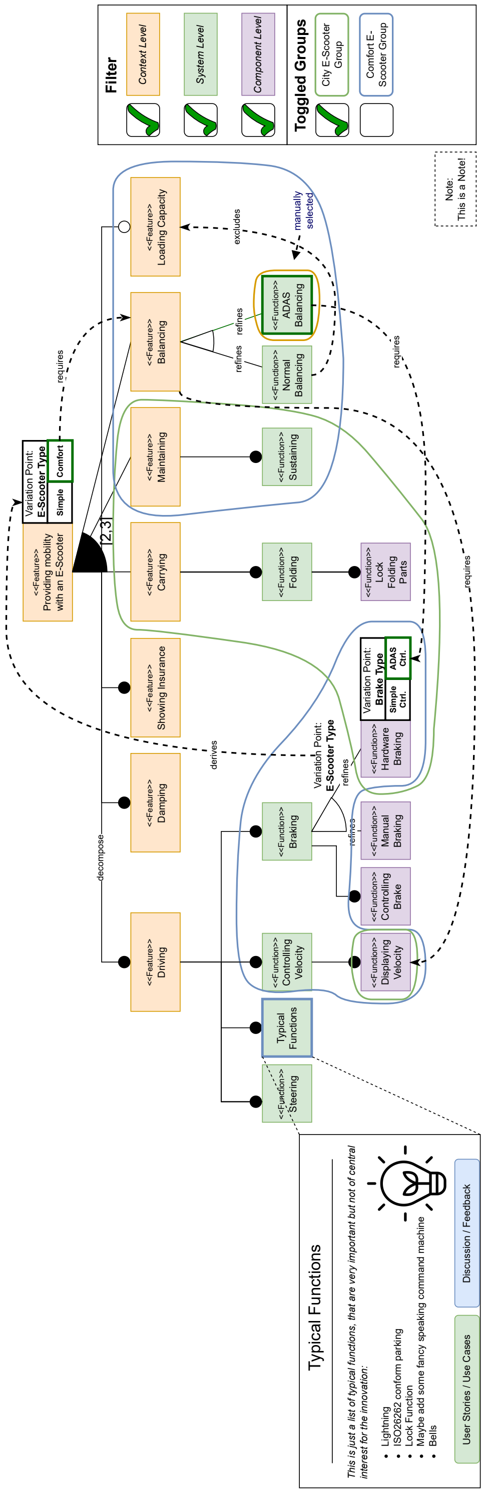


Figure 6.4.: **The identified features and functions for the innovation of "Providing mobility with an e-scooter" including the system level and component level.**

6.4. Functional Perspective FAQ

The FAQ splits up into the following categories:

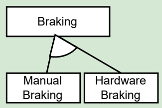
- Questions and Answers about the general Feature Trees
- Questions and Answers the Tooling
- Questions and Answers about the concepts an dependencies
- General Questions and Answers about the Functional Perspective

6.4.1. Feature Tree Base

What is a Configuration?

Taken from the Glossary: A feature Configuration is a chosen set of Features that represent an innovation where every choice is resolved.

Example: If the whole model consists only of the *Braking* Feature and the *Manual Braking - Hardware Braking* Choice, then the model has two configurations. Note that the Alternative-Relation can be represented by an Or-Relation with Cardinality [1,1] too:

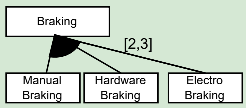


1. One Configuration would be the set of chosen Features and Functions {*Braking, Manual Braking*}.
2. The other Configuration would be the chosen set {*Braking, Hardware Braking*}.

What are Cardinalities used for?

Cardinalities describes how many Blocks (FP) shall be chosen from *Alternative Or Choices* to create a valid configuration.

Example: If the whole model consists only of the *Braking* Feature and the *Manual Braking, Hardware Braking - Electro Braking* Choice with the Cardinality [2,3] then the model has four configurations:

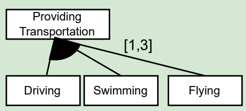


1. The Configuration with 2 Variants taken: {*Braking, Manual Braking, Hardware Braking*}.
2. The second Configuration with 2 Variants taken: {*Braking, Hardware Braking, Electro Braking*}.
3. The third Configuration with 2 Variants taken: {*Braking, Manual Braking, Electro Braking*}.
4. The Configuration with all Variants taken: {*Braking, Manual Braking, Hardware Braking, Electro Braking*}.

What are Or-Relations used for?

Or-Relations are used whenever more than one Variant shall be selected, or when the the number of chosen Variants shall have a set minimum and maximum bound.

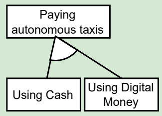
Example: The innovation shall *Provide Transportation* from *A-to-B*. Ideally it shall overcome all obstacles in the most convenient manner. This could include the Functionality of *Driving, Swimming* or *Flying*. However, providing one of the three choices is considered sufficient here.



When to resolve choices (Early or Late Binding)?

While deciding which choices to take is not really relevant for the context of innovations, Feature Models lack clarity from the side of the resolution of the choices. This results in misunderstandings that are harder to resolve later on. To understand the problem, a *Paying the autonomous taxis* Feature with the Alternatives *Using Cash* and *Using Digital Money* Example is used. Now two interpretations arise:

1. The provider of the autonomous taxis has two taxi configurations. One that provides payment methods by using Real Cash and one that provides payment methods by using Digital Money. The Resolution is this case lies within the company.
2. The provider of the autonomous taxi has one taxi configuration that provides both payment methods. The user of the taxi have then to choose one of those payment methods. Here the user resolves the choice.



There are upsides and downsides for both interpretations. To overcome this problem the modeler shall be aware of it and resolve the problem in the description of the *Paying autonomous taxis* Feature. Being part of the descripton is considered enough here. It does not have to be a Block attribute like proposed in the original Feature Tree publication [Kang90]. Note that for simplicity, the Variation Points do not have any *binding time* attribute.

6.4.2. Tooling

Can a user select Multiple Refinement Choices from one Variation Point at once?

Yes, cited from Model Element Description: "[..] Multiple *Alternatives* of the same *Variation Point* shall be selectable too.Then the Analyse is restricted to choose one *Alternative* out of the selected Set of *Alternatives* in all analysed configurations. [..]". This includes Refinement Alternatives.

6.4.3. Concepts and Dependencies

General FAQ

Why does the Functional Perspective extend Feature Trees (and Diagrams) and does not just use the standard FODA - Feature Trees from [Ka90]?

The extensions can be categorised into five categories why the extensions where made:

1. **Raising Expressiveness**: Known Extensions like Or Relations with *Cardinalities* are added to model restrictions in the Configurations that can not be expressed with standard Feature Trees.
2. **Adding Version Control and Tracing Capabilities**: The Block (FP) Properties *version* and the references to the *User Stories* are added to support typical requirement engineering processes.
3. **Adding Custom Domain Support**: Innovation Modeling is quite an abstract Domain. The Extensions with the *Custom Blocktypes*, *Abstraction Level* and *Relations* are added to refine the model elements to better fit the domain of an innovation and increase its Usability.
4. **Resolving Interpretation Problems**: A lack of Information and Clarity creates Misunderstandings. The original Feature Trees [Kang90] identified that problem and added information next to Feature Trees. Here *HTML Divs* and *Descriptions* are added to Blocks as additional model elements. Relations can be differentiated into the two main Engineering Concepts *Refinement* and *Decomposition*. *Variation Points* are added to Alternative Choices to explicitely specify the variation object and lastly the distinction between *Features* and *Functions* were added to explicitely state, which elements shall be mapped on structural solutions.
5. **Compact Representation and Usability**: Feature Trees can get very large, thus a compact representation is desirable. *Variation Lists* reduce space of modeling Alternatives and *Variation Point Selection Derivation* reduce repititions among Variation Points. Lastly *Grouping* is added for fast testing certain sets of Configurations and *Abstraction Levels* as Block Properties are added for better Filter Functionality.

Note, that points 2-5 do not increase Expressioness in the sense that Tools for Feature Trees can still be used without the additional informations of the points.

There are several ways to restrict the set of configurations. Which are they?

The ways to restrict the set of configurations are:

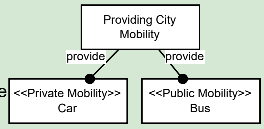
- Replacing Optional Relations with Mandatory Relations
- Removing Choices,
- Cardinalities,
- Require / Exclude Relations,
- Groups,
- Variant Point Selection Derivation
- Manually selected Blocks and selected Sets of Alternatives (Tooling)

Each element is described under *Model Element Description* Section in more detail.

How to handle domain specific categories and domain specific dependencies?

Domain specific categories shall be added by using domain dependent custom stereotypes (aka categeries, aka types) for features and functions to better filter them. The same shall be done for Relations: Each general (Decomposition) Relation shall be stereotyped with custom types. Additionally the Tooling shall provide the possibility of adding profiles.

Example: For the *city mobility design* domain the Feature *Providing City mobility* may have two Features as childrens (*Car, Bus*). The domain has several important categories like private and public mobility, which are used as custom stereotypes here. Additionally the Relation is typed with the custom stereotype "provide" too. The tooling may provide coarse analysis techniques for custom stereotypes.



Features and Functions

When to use Features and when to use Functions?

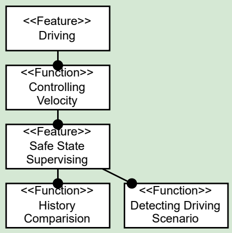
Features shall be used whenever the Block (FP) describes a characteristic or logical or behavioral unit of an innovation that is too abstract to be referenced on some Block in the solution space.
Functions on the other side shall be used whenever the Block (FP) describes a logical or behavioral unit that shall be fulfilled by certain Blocks in the solution space. Functions are seen as a set of logical units that describe a Feature as mappable units. Thus - when the Functions are mapped - the Feature is also considered to be fulfilled.
However there is no clear line whether some characteristic or logical unit is a Feature or Function. Use as appropriate! Note that, making use of not using any Stereotype (Feature, Function) is fine too. The Block will be interpreted as a Feature then (For analysis purposes)!

Example: *Driving* is considered a Feature, as there are multiple Blocks in the solution space that fulfill the Driving Feature together and it seems unreasonable to assign *Driving* to one specific Block. *Braking* on the other hand, can be reasoable assigned to a Block or set of Blocks in the solution space. Thus it is considered a Function.

Can a Function have a Feature as child?

Yes, Functions can have Features as childs. This allows to decompose Functions into abstract, but very specialized Behavior.

Example: The Feature *Driving* is decomposed in the Function *Controlling Velocity*. *Controlling Velocity* is considered a mappable Function but also requires *Safe State Supervising* ability. *Safe State Supervising* is however really abstract and not mappable on solutions thus it is considered a Feature. The Feature shall be realized by a *History Comparision* Function and by a *Detection of the Driving Scenario* Function.

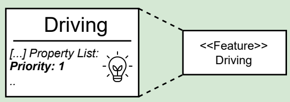


If all child Functions of a Function are mapped, shall the parent function be mapped on the solution space too?

Yes definitely. Else the integration of all subfunctionality may be forgotten!

Do Features and Functions have properties?

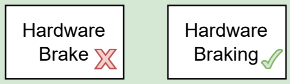
Features and Functions can own custom properties if the attached HTMLDiv is insufficient. However custom analyses (like consistencychecks) and tooling (like filtering) must be provided to make them useful.



Does naming conventions for Features and Functions exist?

Yes, Features and Functions are understood here as actionable items. To underline this, they shall be named like an activity.
Many literature examples describe system solutions instead of actions, which we explicitly do *not* want.
They shall be stated as systems or components on the Structural Perspective.

Example: The Feature *Hardware Brake* shall not be used as it implies being a component.
Instead it shall be named as *Hardware Braking* (an action).

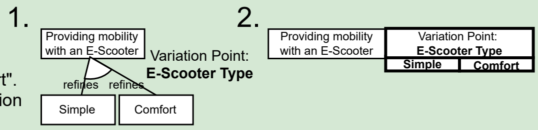


Variation Points - Variant List Representation - Variation Point Selection Derivation - OVM

What is a Variation Point?

From the Glossary: A Variation Point is a representation of a variable item of the real world or a variable property of such an item within domain artefact enriched by contextual information. [Pohl]
A Variation Point can be interpreted as a named Alternative Choice.

Example: The Feature *Providing Mobility with an E-Scooter* (1.) has an Alternative Choice of choosing the E-Scooter Type "Simple" or "Comfort". The E-Scooter Type is the Name for the Variation Item (Thus the Variation Point) with the Choices "Simple" and "Comfort". The Alternative Choice represents a Refinement of the E-Scooter Type, thus the Refinement Representation (2.) is eligible too.

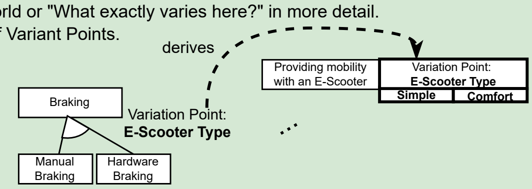


Why add Variation Point names to Alternatives (Variants)?

There are two different reasons why a name for an alternative choice is useful:

1. When the user want to specify the variable item of the real world or "What exactly varies here?" in more detail.
2. When the user want to make use of the *Derivation* Relation of Variant Points.

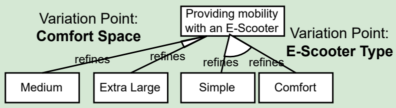
Example: The Variation Point E-Scooter Type is labeled to specify the object of variation in more detail, plus for using the Derivation Relation extension.



Can a Feature or Function Block have Multiple Variation Points?

The model itself does not prohibit multiple Variaton Points for a Feature or Function Block. However the number of Variation Points using the Refinement Representation is restricted to one for one Feature or Function Block for Readibility. Note that - while not prohibited - the use of multiple Refinement-Relation based Variation Points can cause wrong models when the Refinements are not composable (see the example). For more details see the Model Element Description of *Alternatives and Variation Points*.

Example: The Feature *Providing Mobility with an E-Scooter* has two different Refinement based Variation Points. One specifies the the E-Scooter Type with "Simple" and "Comfort" while the other specifies the Comfort Space with "Medium" and "Extra Large". The Variation Points are providing a contradiction, since one expects that an E-Scooter of Type "Simple" does not have any Comfort Space.



In this case it is better to restructure the model by either refining the "Comfort" E-Scooter with the Comfort Space Variation Point or by fusing both Variation Points to one Variation Point with the choices {*Simple, Medium Spaced Comfort, Extra Large Spaced Comfort*}.

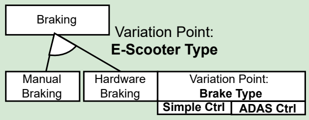
Do Variant Lists have Cardinalities?

No, Variant Lists do not have Cardinalities. They represent an Alternative Choice with Refinement Relations and Cardinalies of [1,1] are only reasonable for Refinements.

When to use Variation Lists and when not?

Variation Lists are a special Respresentation for Refinement-Alternative Choices. They can be used to reduce the needed representation space for alternative choices with a Variation Point name. The best use of the Representation is, if the user want to highlight a variance choice in the feature model. It is recommended to use this representation whenever possible. One shall refrain from this representation, when the representation form is unknown to a wide range of readers and the extension having a negitive impact on the reability for such readers. This may be the case if the Functional Perspective Feature Tree is small and the extension seems unreasonable high to learn for the next reader. Note that, the depth of this Representation is one, thus Refinement of Refinements can not be represented in this special Representation Form. If such a Refinement of Refinements have to be specified, the use of the Feature Model Representation is reasonable.

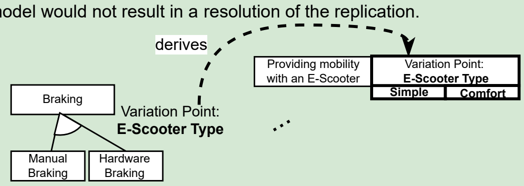
Example: The *Hardware Brake* is one Brake Type of the *Braking* Function, that is chosen by the Type of the E-Scooter. The *Hardware Brake* however has two own Brake Types too. The use of the special Representation here seems reasonable in a large model, however may be omitted in a small model for readibility.



When to use Variation Point Selection Derivation?

Variation Point Selection Deriviation allows to reuse the selection of the derived Variation Point again. This results in a restriction of the number of configurations of an innovation. Variation Point Selection Deriviation shall be used with caution as it represents a kind of replication of a choice. It should only be used if a Restructioning of the model would not result in a resolution of the replication.

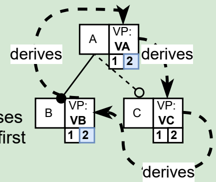
Example: *Braking* can be realised through a *Manual Brake* or through *Hardware Braking*. The E-Scooter Type shall however be the decision ground for the choice of the Braking type. A *Manual Brake* shall be chosen when the E-Scooter Type is "Simple". A *Hardware Brake* shall be chosen when the E-Scooter type is "Comfort". A Restructuring of the model is unfeasible, as the choice is far away from the E-Scooter Type choice. Thus the use of Variation Point Selection Derivation seems reasonable.



What does Circular Dependency Freeness for Variation Point Selection Derivation mean?

Circular Dependency Freeness means that no Feature or Function can derive any choice from a Feature or Function that has a deeper level in the model tree than the Feature or function itself (Meaning deriving from one of its child, sipling child, childrens child, and so on is not possible). This resolves the problem of Circular Dependencies and is mostly prohibited for avoiding confusing and simplifying analysis implementation.

Example: The Variation Point from A is derived from the Variation Point of C. Variation Point of C is derived from Variation Point from B. Variation Point from B is however derived from Variation Point from A. It now raises the question, where the choice is firstly decided and who derives from whom. While this can be technically solved by deciding the Variation Point with the lowest depth first, this is surely something that causes confusion and has no real point of modeling it that way. Generally deciding a Variation Point on a lower level first is considered unnatural and confusing itself. Thus the Prohibition of deriving from lower level Variation Points seemed reasonable.



What is the Difference between the Orthogonal Variability Model (OVM [Pohl]) and the integrated Variability model used here?

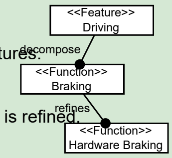
There exists a number of differences between the OVM:

- In both approaches Variation Points are used. However Variation Points are treated differently. In OVM they are a stand-alone model element, while they are only a named alternative choice here.
- Variants are treated similary. In OVM they are a stand-alone model element that restricts certain configurations via Groups. In the integrated model here Variants are only normal Alternatives from Feature Trees.
- In both approaches the concept of Groups are defined. However Groups are treated different again. In OVM Groups are chosen by the OVM Choices and here Groups are chosen by the user.
- OVM separates variability from features, such that product lines can use the same feature tree and a differentiation variability model for different countries. This comes at the cost of a degree of replication between the Feature Tree and the variability model. Here this kind of common Feature Tree does not exist at the "gain" of less replication.
- The OVM owns several additional Dependencies between OVM elements. These are not existing here (Learnability Pro and Expressability Con):
  - OVM Variants
  - OVM Variation Points (alternative, or) with dependencies (mandatory, optional, require, exclude)

Conclusion: When there are not existing several feature trees (as for SPL), then the disadvantages of OVM makes the integrated model superior. This is exactly the case for GENIAL!

Relations

Where is the difference between the Refinement and Decomposition Relations?



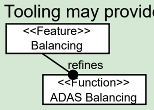
A Decomposition (or *has-a*) Relation describes a partitioning of a Feature or Function into its subfunctions and subfeatures.
Example: Braking is one important function that must be realized to achieve the Driving Feature.

A Refinement *(or is-a)* Relation of a Feature or Function describes a fuller specification of the Function or Feature that is refined.
Example: Hardware Braking is one way of how Braking can be realized.

Those modelers - who do not need the difference in Refinement and Decomposition - can abstract the Relation by leaving the Relation Type out.

When shall Relations be labeled as Refinement or Decomposition Relations?

There is no right or wrong answer to the question when to label Relations with "refine" or "decompose". A basic guideline is to label Relations whenever the label provide some kind of additional value to the reader. For example, if it is unclear how exactly a function or feature correlates with its parent. If a Relation is not labeled, then it does not have a default interpretation either! Note that the Tooling may provide additionally Filter Functionality and Analysis based on the Relation Type.



Example: It is unclear what the Difference between Balancing and ADAS Balancing is. ADAS could describe a logical unit on its own that is needed for Balancing or a Refinement (and thus an exclusion to other alternates). Here it was meant as a simple Refinement.

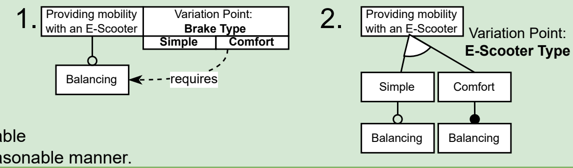
When to use Require / Exclude Relations?

Require-Relations and Exclude-Relations are ways to restrict the number of configurations of an innovation. Those Constraint Relations shall be avoided to maintain Readability and Understandability whenever possible. Only if

1. a restructure of the Hierachy of the Features and Functions (and their parents) seems unreasonable
2. and if the omission of the constraint is not possible

those Constraint Relations shall be used. 1. may be seen unreasonable in certain cases, because a restructure may lead up to redundancy and thus a higher count of Feature and Function Blocks or would require a Constraint Relation at another point in the model.

Example: For the *Mobility*-Feature the *Comfort* E-Scooter requires a *Balancing* Unit. A *Simple* E-Scooter can have optionally a *Balancing* Unit too. There are two ways to represent this dilemma. 1. uses a Require Constraint Relation to visualize the dependency. 2. explicitely models the *Balancing* Units as two seperate Features. In this simple example 1. seems more reasonable to avoid redundancy but in a large model, both may be used in reasonable manner.



Miscellaneous

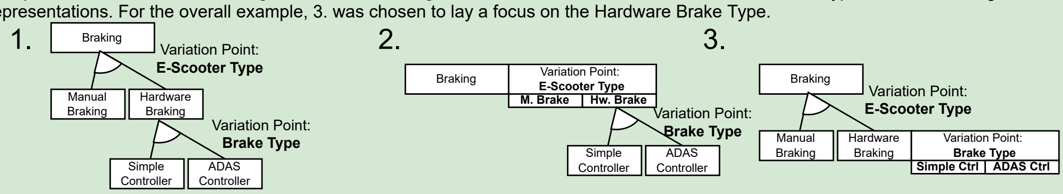
How to represent Refinements of Refinements?

There are three ways to represent Refinement of Refinements:

1. Using the Standard Feature Tree Representation for both Refinements only. This is not recommended.
2. Using the Variant List Refinement Representation for the first Refinement and the Standard Feature Tree Representation for the second Refinement.
3. Using the Standard Feature Tree Representation for the first Refinement and the Variant List Refinement Representation for the second Refinement.

The ways 2. and 3. are the recommended ways. Depending on where to lay the focus, the modeler has to decide to either choose 2. or 3. Note that Refinements of Refinements do not have a special Representation for more depth to avoid Readibility and Understanding problems.

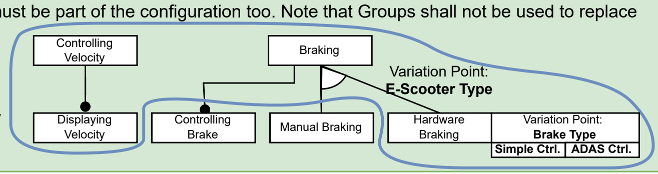
Example: The Refinement of *Braking* to *Hardware Braking* and then the Refinement of the Hardware Brake Type has the following three Representations. For the overall example, 3. was chosen to lay a focus on the Hardware Brake Type.



What are Groups used for and when *not* to use Groups?

Grouping is an additional Usability Feature that is used when the user wants to toggle a certain set of Features and Functions as required for the next analysis or for restricting the focus on the model visually. It semantics requests that if one Block (FP) of a Group is part of the configuration, then every other Block in the Group must be part of the configuration too. Note that Groups shall not be used to replace non-temporary Require-Relations.

Example: The blue Group it used for focusing visually on a certain set of Braking Blocks that are connected to the Velocity Display. Any started analysis require now that all Blocks in the Group are part of the configuration.



User Stories and Use Cases are referenced in the Features and Functions. But how are User Stories and Use Cases connected to each other?

User Stories and Use Cases are preliminary activities to find and refine Features and Functions. They are only viewed as references in the Functional Perspective. The creation and changing of User Stories and Use Cases is not part of any IMoG Perspective. A dedicated Tool for handling User Stories and Use Cases is expected. The creation and handling on how to decompose Use Cases and User Stories is thus up to the creator and their dedicated Toolchain. Nonetheless, references of User Stories and User Cases to Features and Functions can be used to test User Stories / Use Cases coverage.

6.4.4. General Stuff

How does tracing between User Stories / Use Cases and Feature / Function looks like and how does one ensure traceability if one user story / use cases changes? What is impacted?

There exists currently no default Tooling for creating and handling User Stories and Use Cases. Thus the way how tracing looks like is not yet defined. The implementation may include files for each User Story and Use Case or model references into a PLM tool. However this is a tooling question and not a focus in the definition of the Functional Perspective and its contents. The only thing precise yet is that there shall be User Stories and Use Cases referenced in the Features and Functions!

Regarding the change of a User Story and Use Case: It is not automatically possible when changing the User Story and Use Case directly, because the tooling is not fixed. However a Change Impact Analysis showing the direct and indirect dependencies of a User Story and Use Case to Features and Functions shall exist to enable tracking the affected entities.

Feature Trees and Feature Diagrams do not scale well as most other graphical languages. Do Feature Trees and Feature Diagrams provide enough scalability?

In the context of modeling future Innovations the scalability is sufficent. There are a multiple minor reasons which add to this assessment:

- There is not much information available about future innovations. Additionally the information has a high level of uncertainty. Thus modeling focuses on describing innovations without many abstractions and model elements.
- The Functional Perspective shall not contain information about solutions. Thus only ideas with their functionality are modeled. It is expected that the model is sufficiently small.
- In the context of public roadmapping, partners do not want to share Intellectual Property and want to focus on a common set of elements. Thus the number of elements is rather small than high.
- The tooling shall provide information filter like the proposed hiding of the HTML DIVs into the Blocks. Thus additional information shall not clutter the screen.

7. Quality Perspective

The Quality Perspective is the third perspective in IMoG applications and targets the coverage of mostly non functional requirements for both the problem space and the solution space (see Figure 7.1). The Quality Perspective contains for example constraints from the legislation, robustness requirements and performance requirements. Functional requirements should not exist in the Quality Perspective (hence its name) because the Functional Perspective already covers the features and functions of the innovation. Exceptions that cannot be handled on the Functional Perspective are fine. The Quality Perspective takes requirements from both sides: The problem space and solution space and represent an interface between the spaces.

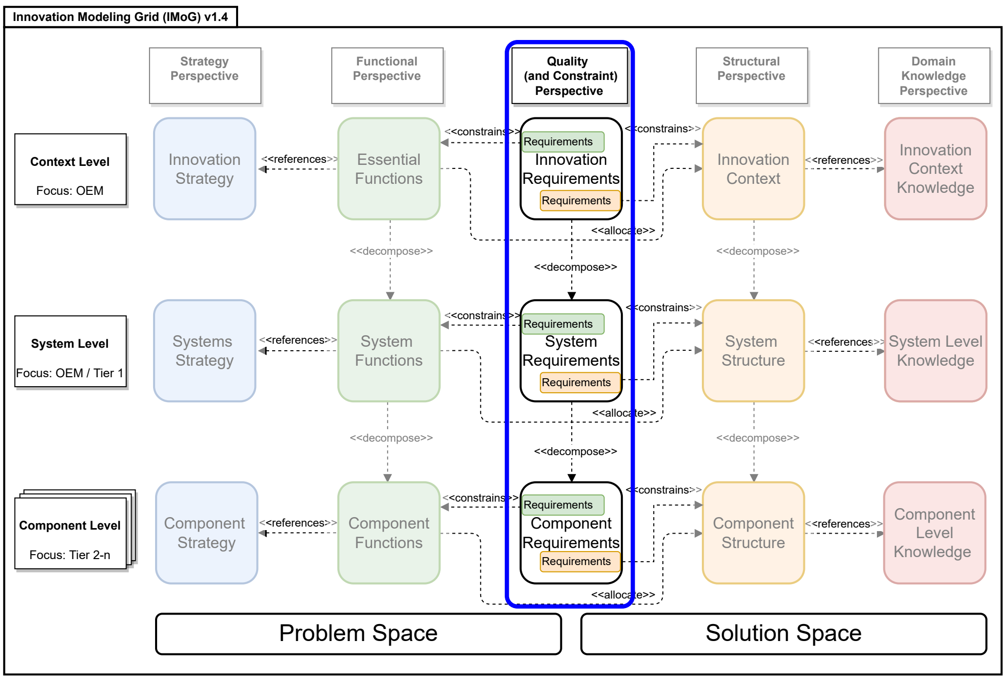


Figure 7.1.: **Location of the Quality Perspective in IMoG**

The Quality Perspective contains two representations: The table format for representing the requirements data. The dependency view for representing the inter Perspectives relations as well as the decomposition of the requirements (parent-child relation) With the Quality Perspective based only on generic tables, some recommendations for the data fields of each requirements were made. Special extensions do not exist.

The chapter is structured as followed: In Section 7.1 the meta model and its model elements are presented. In Section 7.2 an example of the Quality Perspective is given. The strengths and limitations of the Quality Perspective are discussed in Section 7.3. A FAQ finalizes the description in Section 7.4.

7.1. Model elements

The meta model of the Quality Perspective (see Figure 7.2) consists only of two important model element: the *Requirement* and the *Requirement Relation*. The *Requirement* element has quite a few attributes that are recommended to fill. This section will emphasize on presenting the attributes.

The *Quality Perspective Model* contains as its main members the *Requirement* element. The *Requirement* has a bunch of attributes. Some of those are modeled as an own Block to allow the restriction of the data types in form of Enums (Variable types with defined possible values). The Stereotype, the abstraction level and the assignee are part of them. To keep them extensible, each of them has a customization interface. Additionally, custom attributes can be defined. Each requirement can have relations. There are currently three types of relations:

- the *Parent-Child* relation to represent a requirement *Decomposition* or requirement *Refinement*.
- the *Constraint* relation for describing which (problem space / solution space) Block shall satisfy the requirement and
- the *Custom* requirement for extensibility.

Each attribute and relation is described in more detail in the following.

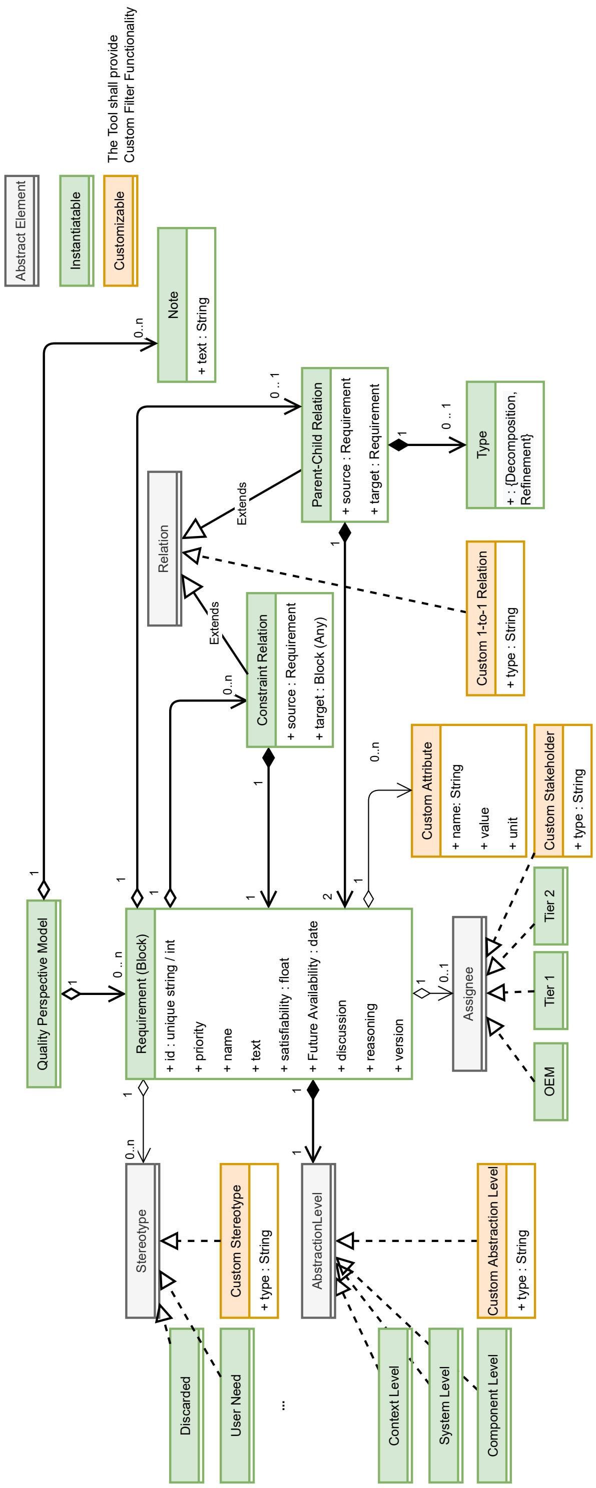


Figure 7.2.: **The model elements of the Quality Perspective.**

Model Elements:

- Quality Perspective Model
- Requirement
- Relations
- Defined Stereotypes
- Notes

Model Elements

In the following the model elements are introduced in four parts. First the *Quality Perspective Model* is introduced, then the *Requirement* with all its attributes and associated Enums is described. Afterwards, the *Relations* are described. Lastly, the defined *Stereotypes* and *Notes* are presented.

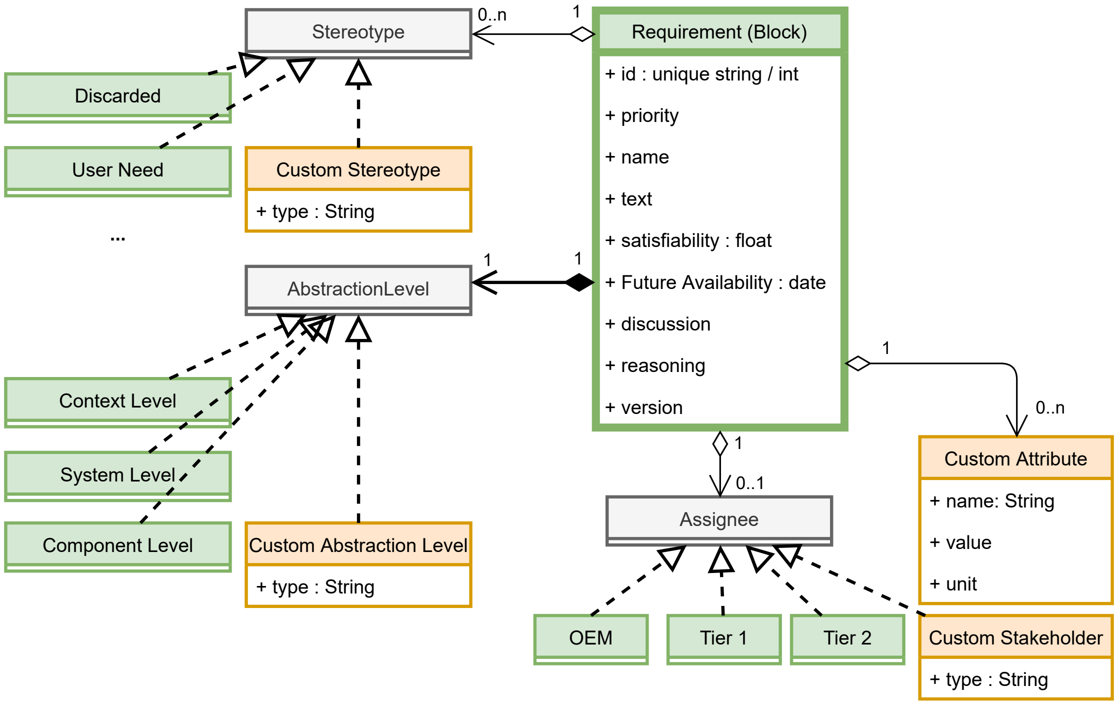
Meta Model Element:

Description:

Quality Perspective Model
The *Quality Perspective Model* is the diagram of the Quality Perspective of an innovation. It contains all model elements of the Quality Perspective.

Example: A full Quality Perspective Model example is shown in Section 7.2.

Meta Model Element:



Description:

Requirement (Block)

The *Requirement (Block)* is the main element of the Quality Perspective and represents any Quality Requirement, user need, constraint and so on. It is a quite complex element due to its importance. It defines many attributes. Each attribute can be imagined as a column in a requirements table. Except the outsourced description of the relations, each attribute is described in the following:

- The *id* for a requirement to identify and trace it. It typically takes a unique number of a unique string as an scheme. Both are possible here.
- The optional *priority* for giving the requirement an importance. The value interpretation has to be defined by the stakeholder. Default is an empty field.
- The *name* of the requirement to better identify it (for humans).
- The *text* of the requirement. It can be described as natural language or as formal sentences. If it is formulated as formal sentences then the requirement shall have a proper *Stereotype* to allow analysis to identify and parse it.

Requirement (Block) continued

- The *satisfiability* of the requirement. A numerical number between 0 and 1 which estimates the chance the requirement is fulfilled according to the year and other parameters. The satisfiability may be given by a formula instead of a static value.
- The *Future Availability* describes the year date when the requirement will be relevant. In the days / years before the given date this Requirement shall be 'ignored'.
- The *discussion* field provides a platform for discussing the requirements within the value chain.
- The *reasoning* field allows to give a rationale for the requirement definition.
- The *version* field is for proper version management and to identify updated requirements.

The requirement has additionally attributes with predefined value ranges (so called Enums). These are represented by an own entity in the meta model:

- The *Abstraction Level* of the requirement defines the level of abstraction the requirement represents. It can be either *Context Level, System Level, Component Level* or from the type *Custom Abstraction Level*.
- Optional *Stereotypes* can refine the category of the requirement further. There are some Stereotypes predefined, like *Discarded* (Requirement), *User Need* and so on. However one can define their own *Custom* Stereotypes. A list of predefined Stereotype is presented in the FAQ.
- The optional *Assignee* of the requirement represent the responsibility owner of the requirement. It can be either of the predefined type *OEM*, *Tier 1* or *Tier 2* or set by a string to a *Custom* Stakeholder.
- If all the above attributes are not enough, or there is an attribute missing for the specified domain, then one can define a *Custom* Attribute with a *name*, a *value* and an *unit*.

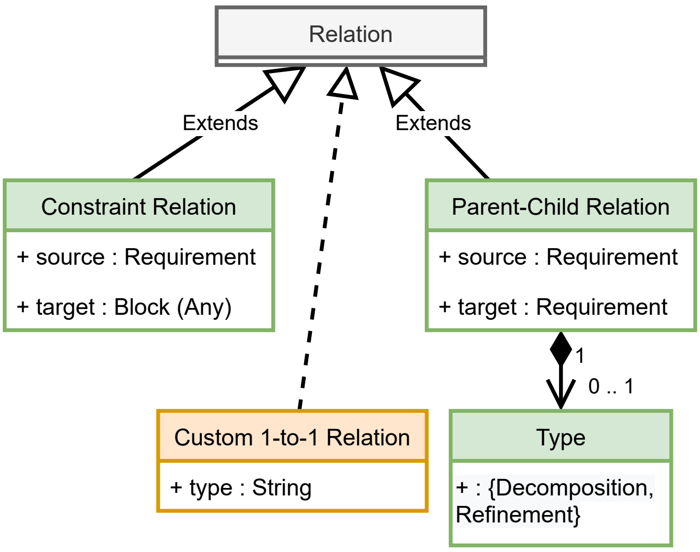
Example: An example with 4 requirements is shown below. The requirements attributes are presented shortly:

- The *ids* are represented as numbers for this example.
- The *priority* for three requirements were determined. The value 1 describes here the most important priority.
- The first safety requirement is expected to be in all cases fulfilled (thus *satisfiability* is 1=100%) to be conform to the German traffic rules. The other requirements should be fulfilled up to a certain degree according to the expectations.
- Each requirement has a *name* and a *text*. All of them are described as natural text with some being unfinished.
- The *Future Availability* is set to Now. Meaning the requirements shall be already considered.
- The *parent* requirement and the *targets* represent relations, they are not further described here.
- Each requirement has *Stereotypes*. The first requirement is considered to be part of the safety concept while the three other requirements are considered user needs.
- All four requirements are considered to be part of the *Context Level* abstraction level.
- The *assignee* of these requirements is either the *OEM* or the *Tier 1*.
- The *reasoning* and the *discussion* are not further detailed here.
- The *version* number of each requirement.

ID	Priority	Satisfiability	Name	Text	Future Availability
1	1	1	Safety	The e-scooter shall have 2 brakes.	Now
2	1	0,8	Braking	The braking power shall be greater than ..	Now
3		0,8	Damping	The cushed power shall be >= ..	Now
4	2	0,9	Weight	The weigth shall be < ..	Now

ID	Parent Req.	Labels / Sources	Target	Abstraction Layer	Assignee	Reasoning	Discussion	Version
1	-	Safety Concept	Braking	Context Level	OEM	...	...	2
2	-	User Needs	Braking	Context Level	Tier1	...	...	1
3	-	User Needs	Damping	Context Level	Tier1	...	...	1
4	-	User Needs	Carrying	Context Level	OEM	...	...	1

Meta Model Element:



Description:

Relation (QP) **(Read: Relation on the Quality Perspective)**
The abstract *Relation (QP)* describes Relations between requirements or between a requirement and a target Block on the Functional Perspective or on the Structural Perspective. Relations are further categorized into

- *Constraint* relations to describe which Block on the Functional Perspective or the Structural Perspective shall fulfill the specified requirement. This constraint relation is often called «satisfy» relation when reversed. However, to avoid confusion, only «constraint» relations shall be used.
- *Parent-Child* relations between requirements. The *Parent-Child* relations can be specified by an optional type, which can be either of value *Decomposition* or *Refinement* to specify the relation type.
- *Custom 1-to-1* relations between requirements can be described by using *Custom* attributes of the requirement.

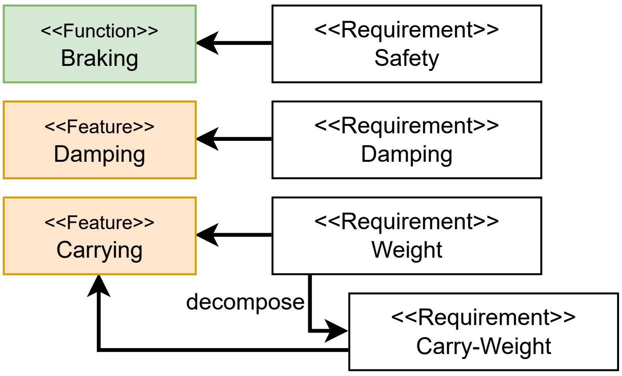
Example: An example for the *Constraint* Relation and the *Parent-Child* relation is shown below. The *Custom 1-to-1* relations are skipped for now.

ID	...	Name	Parent Req.	Labels / Sources	Target	Abstraction Layer	Assignee
1		Safety	-	Safety Concept	Braking	Context Level	OEM
3		Damping	-	User Needs	Damping	Context Level	Tier1
4		Weight	-	User Needs	Carrying	Context Level	OEM
5.2		Carry-Weight	4 (Dcmp.)	User Needs	Carrying	System Level	Tier1

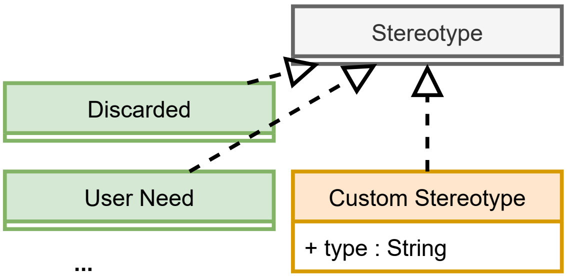
The four requirements above have two important attributes that represent relations:

- The *Parent* requirement represent a *Decomposition* or *Refinement* relation. Only the "Carry-Weight" requirement has a parent: The weight requirement. The exact type of the relation is in this example not described.
- Each requirement has a *target*. The targets describe in all four cases a function of the Functional Perspective (which can be found in the associated file).

The same requirements represented in the Relations View would be depicted as followed:



Meta Model Element:



Description:

Stereotypes
The *Stereotype* can refine the category of the requirement further. A requirement can have multiple stereotypes. However, it is recommended to not apply two that contradict each other or are of the same category. The following Stereotypes are predefined: Requirement Categorization Stereotypes:

- **Quality Requirement**
  - **Performance Requirement**
  - Technical Professional Guess
  - User Need (non functional)
- **Constraint**
  - Safety Requirement
  - Security Requirement
  - Legal Constraint
  - Technology Requirement

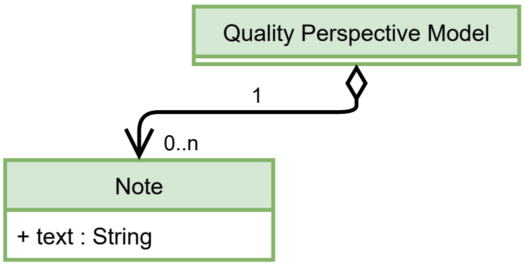
Requirement Status:

- Discarded
- Proposed
- Confirmed (The default interpretation if no requirement status is given)

There is some overlap in the definitions of the categories, for example between Quality Requirement and User Need. If one can not decide which category to choose, then take the one that feels as best fit. The categories are only used for Filtering Purposes, thus a miscategorization is not that harmful. The three bold categories are of special interest of some suppliers. Maybe these shall be given some special treatment?
It is possible to define *Custom* Stereotypes.

Example: ToDo!

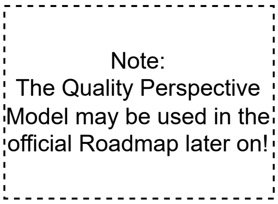
Meta Model Element:



Description:

Note
The *Note* can be used to add information to the model that can not or should not be modeled. Notes should be used sparsely!

Example:



7.2. E-Scooter example

The example of the Quality Perspective comprises the innovation requirements of "Providing mobility with an e-scooter" in two views called Requirements Table View and Relations View.

The Requirements Table View (see Figure 7.3) contains several identified quality requirements, user needs, constraints and so on for each abstraction level. The Requirements Table View focuses on showing all details of each requirement. The requirements of the component level are filtered out here (to keep the example small). The requirements are listed as relational data bases thus SQL queries shall be executable. The exact details of those requirements are not further explained.

The Relations View (see Figure 7.4) hides the attributes and shows the requirements only by their names. The attributes shall be visible when marked with the mouse in an extra window. This view focuses on presenting the Parent-Child relations between two requirements and the Constraint relations of the requirements to the Blocks of the other perspectives.

7.3. Quality Perspective: Strengths and Limitations

The Quality Perspective contains simply requirements tables and relation views. The Quality Perspective does not capture (many) functional requirements, because these requirements should be handled in the Functional Perspective. Otherwise, there is nothing special about the Quality Perspective. One noteworthy strength is the tracing of requirements to features, functions and solutions, because these links are already weaved into IMoG.

ID	Priority	Satisfiability	Name	Text	Future Availability	Parent Req.	Labels / Sources	Target	Abstraction Layer	Assignee	Reasoning	Discussion	Version
1	1	1	Safety	The e-scooter shall have 2 brakes.	Now	-	Safety Concept	Braking	Context Level	OEM	...	...	2
2	1	0,8	Braking	The braking power shall be greater than ..	Now	-	User Needs	Braking	Context Level	Tier1	...	...	1
3	2	0,8	Damping	The cushed power shall be >= ..	Now	-	User Needs	Damping	Context Level	Tier1	...	...	1
4	2	0,9	Weight	The weigth shall be < ..	Now	-	User Needs	Carrying	Context Level	OEM	...	...	1
5.1	3	0,95	Carry-Bar	The ergonomy of the handbar shall be better than ..	Now	4 (Dcmp.)	User Needs	**Carrying.Simple**	System Level	OEM	...	...	2
5.2	3	0,7	Carry-Weight	The weigth shall be < ..	Now	4 (Dcmp.)	User Needs	Carrying	System Level	Tier1	...	...	2
5.3	3	1	Carry-Platform	The platform shall be be smaller than ..	Now	4 (Dcmp.)	User Needs	Carrying	System Level	OEM	...	...	1
6	1	0,75	Stability	The stability of each subcomponent of the e-scooter shall be > ..	Now	-	Safety Concept	Balancing	System Level	OEM	...	...	1
7	4	0,8	Modularity	Each subsystem of the e-scooter shall be build and built-in modular.	Now	-	Safety Concept	Maintaining	System Level	OEM	...	...	1
8	2	0,7	Badge	The stability of the badge shall be > ..	Now	-	User Needs	Show. Insurance	System Level	OEM	...	...	3
9.1	1	0,9	Load Capacity-Stability	The stability of each subcomponent of the e-scooter shall be > ..	Now	6 (Refine)	Safety Concept	Loading Capacity	System Level	OEM	...	...	1
9.2	1	1	Load Capacity-Engine	The engine power shall be > ..	Now	0 (Dcmp.)	Engine	Engine	System Level	Tier1	...	...	1
9.3	1	1	Load Capacity-Brake	The braking power shall be greater than ..	Now	2 (Dcmp.)	Safety Concept	Loading Capacity	System Level	Tier1	...	...	3
10	1	1	StVO	The e-scooter shall satisfy all constraints from the eKFV.	Now	-	Government	Driving	Context Level	OEM	...	...	2
11	1	0,7	Distance	The e-scooter shall reach a driving distance > ..	Now	0 (Dcmp.)	User Needs	Driving	Context Level	OEM	...	...	1
12	4	0,1	Flying	The e-scooter shall be able to fly and reach a flying distance of > ..	2055	-	User Needs	Driving	Context Level	OEM	...	...	4
13	1	0,8	Rainy	The e-scooter shall have a water resistant cover.	Now	-	User Needs	Maintaining	Context Level	OEM	...	...	1
14	1	0,8	Dewy	The e-scooter shall have a strong water resistant cover.	Now	-	User Needs	Maintaining	Context Level	OEM	...	...	1
15	1	0,75	Bad Balance	The e-scooter shall have a self balancing unit.	Now	-	User Needs	Balancing	Context Level	OEM	...	...	1
16	1	0,75	Assistance	The e-scooter shall have driving assisting functionality.	2030	-	User Needs	Balancing	Context Level	OEM	...	...	1
17	1	0,75	Comfort	The e-scooter shall have comfort functions like self balancing.	Now	-	User Needs	Balancing	Context Level	OEM	...	...	1
0	1	0,5	Range	The car shall be able to drive 100km with one charging cycle.	Now	-	User Needs	E-Scooter	Context Level	OEM	Else it isn't feasible	...	1

Filter

- [x] Context Level
- [x] System Level
- [ ] Component Level
- SQL Query — Go!

Figure 7.3.: **Requirements Table View: The requirements for the innovation of "Providing mobility with an e-scooter".**

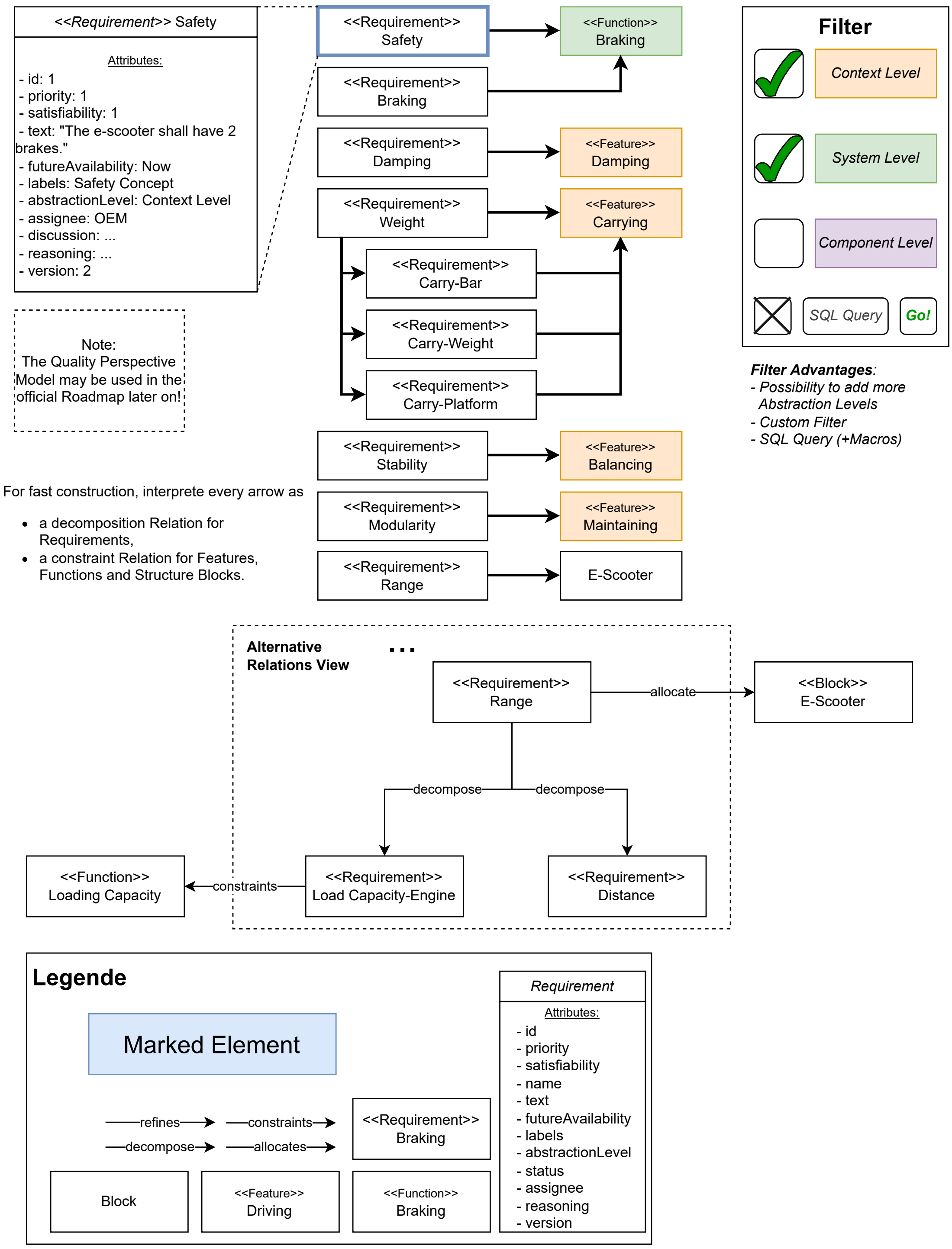


Figure 7.4.: **Relations View.**

7.4. Quality Perspective FAQ

The FAQ splits up into one part:

- Questions and answers about the general requirements on the Quality Perspective

7.4.1. Requirements

General Requirements FAQ

Do Requirements have attributes?

Yes, they do! Each Requirement has per default several attributes next to the name and the text. Moreover, if the default attributes are not sufficient, one can extend the Requirements with Custom Attributes.

<<*Requirement*>> Safety
Attributes:
- id: 1 - future Availability: Now
- priority: 1 - satisfiability: 1

Example: The Requirement 'Safety' has an *id* of 1, a *priority* of 1, a *satisfiability* of 1 and is already relevant.

Is it possible to define Variants or Alternatives of Requirements?

No. The reason for this is to avoid ambiguity and model complexity. The functionality and the solution technologies shall be interchangeable. The Quality Requirements, Performance Requirements may be broad or given with a high uncertainty. However, they shall not be modeled multiple times!

Is it possible to label Requirements with labels like "User Need", "Quality Requirement" or "Constraint"?

Yes. "User Needs", "Quality Requirements" or "Constraints" can be added as Stereotypes in the column "Labels / Sources". Moreover, if the default stereotypes are not sufficient, one can define Custom Stereotypes. A list of predefined Labels can be found under the Model Elements section.

ID	Name	Labels / Sources
1	Safety	Safety Concept
4	Weight	User Needs

Example: The Requirement with the name 'Safety' was part of the Safety Concept and is internally differentiated to the Stereotype 'Safety Requirement'. Thus the Custom Stereotype 'Safety Concept' is introduced. The Requirement with the name 'Weight' goes well with the default Stereotype 'User Needs'.

I can not decide which Stereotype to take? Which one is recommended?

There is some overlap in the definitions of the Categories, for example between Quality Requirement and User Need. If one can not decide which category to choose, then take the one that feels as best fit. The categories are only used for Filtering Purposes, thus a miscategorization is not that harmful.

How to handle domain specific categories and domain specific properties?

Domain specific categories shall be added by using Custom Stereotypes and labeling the domain Requirements with them. Domain Specific properties shall be described by Custom Attributes, which represent added columns to the Requirements table.

Example: ISO 26262 defines a methodology for Functional Safety which defines process steps. Safety Requirements shall have a process step assigned to them. Additionally, ISO26262 defines ASIL levels for different components. These shall be assigned to the Requirements as well. The process steps are given by Custom Stereotypes and the ASIL level is given by a Custom Attribute.

Name	Labels / Sources	ASIL
Safety	Safety Concept	A
Weight	User Needs	D

Requirements Attributes

Which attributes of a Requirement shall be defined and filled?

Only those attributes that provide a use to the innovation shall be defined. The general rule is: The fewer attributes are defined and the fewer attributes are filled out the better is the usability of the whole Perspective. Otherwise the acceptance may suffer in the form that adding new Requirements and updating old Requirements gets tedious due to the many fields to fill. Maintainability will decrease due to increased time spend. With decreased maintenance, the acceptance will further decrease until the content is outdated.

Does naming conventions exist for Requirements?

No, however defining a convention that is tailored to the specific innovation is recommended.

Does the content of the Requirement column 'targets' represent model element relations or just text?

The column contains relations to the respective Blocks.

ID	Name	Target	...
1	Safety	Braking	..

Abstraction Levels have defined Stakeholder assigned. The Assignee column contains Stakeholders as well. Does this overlap create Problems?

The defined Stakeholders of the Abstraction Levels are giving only a general direction who may be responsible. The main reason for the existence of the Abstraction Levels is to decrease the complexity by dividing the Requirements into categories. The optional Assignee column can be used to make exceptions to the general responsibility of the Abstraction Level or to underline that somebody has the responsibility.

ID	Name	Abstraction Layer	Assignee
1	Safety	Context Level	OEM
3	Damping	Context Level	Tier1

8. Structural Perspective

The Structural Perspective is the fourth perspective in IMoG and focuses on the modeling of the solution space of the innovation in an abstract manner. (see Figure 9.1). Based on the functions, features and requirements from the problem space, the Structural Perspective draws the corresponding possible solutions. The Structural Perspective is located at the later phases of innovation modeling shortly before the roadmap is written.

The Structural Perspective bases on the well-known concepts and representations of systems engineering. The concepts of *Decomposition*, *Refinement* and *Variation* that are used on the perspectives are here of high importance too. Next to the three concepts, the

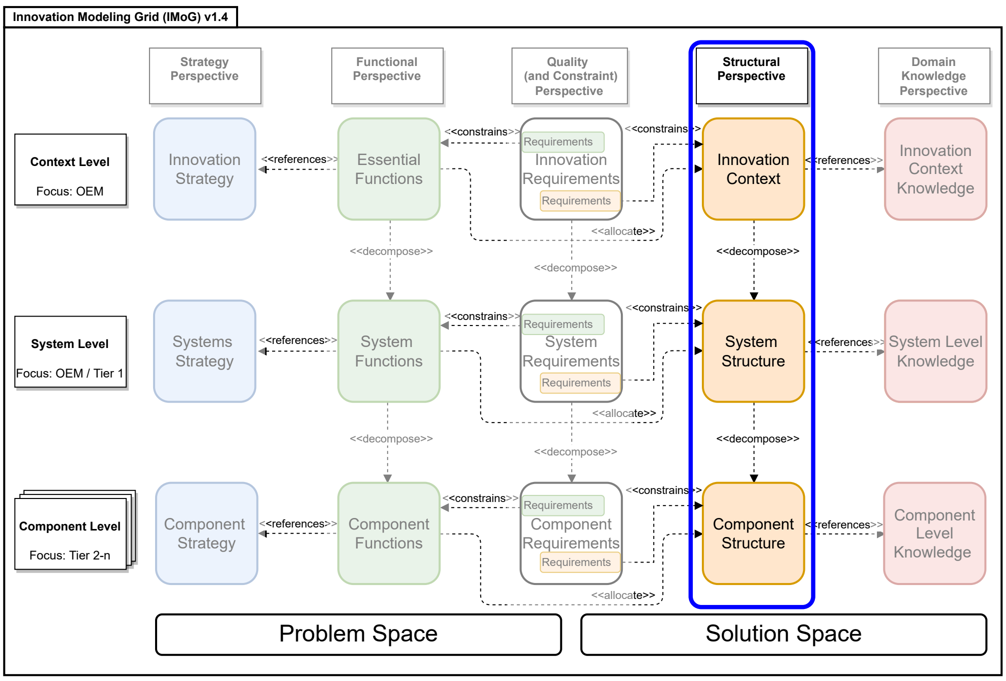


Figure 8.1.: **Location of the Structural Perspective in IMoG**

concept of properties and their relations provide the possibility to describe and compare solutions in detail. The description of solution alternatives is provided by variants for Blocks (SP) and Refinement Blocks. The representation of the Structural Perspective also follows the principles of systems engineering by using hierarchical Blocks and arrows / channels as relations.

The chapter is structured as followed: In Section 8.1 the meta model and its model elements are presented. In Section 8.2 an example of the Structural Perspective is given. The strengths and limitations of the Structural Perspective are discussed in Section 8.3. A FAQ finalizes the description in Section 8.4.

8.1. Model elements

The meta model of the Structural Perspective (see Figure 8.2) builds on the *Decomposition* and *Refinement* concepts. The meta model has the *Structural Perspective Model* as the top level unit of the Structural Perspective. The Structural Perspective Model contains a set of *Decomposition Models*. Decomposition Models represent the canvas fields with their sketchy system models known from system modeling tools like Cameo or Enterprise Architect. The Structural Perspective can have multiple top level models, however it is recommended to only take one unless more are needed.

A Decomposition Model consists of *Structural Model Elements*. The Structural Model Elements include *Blocks* (on the Structural Perspective) and *Relations* between them, *Packages* and *Notes*. Blocks (SP) represent solutions and own a bunch of attributes. Among them are a *name*, a *description*, a *discussion* chat, potentially an *internal model*, a *version*, a flag for the *selected refinement variants*, an *abstraction level* (either of type *Context Level, System Level, Component Level* or a *Custom Abstraction Level*), a *Decomposition Model* and a reference to a possibly refined p*arent block*, *Notes*, a *Stereotype* and possibly some *Refinement Groups*. Refinement Groups are the second important type of concept incorporated in the Structural Perspective model. They allow to refine the Blocks by giving additional information and properties. Each refinement is represented by a Refinement Block owning a set of properties. Like Blocks (SP), Refinement Blocks can have a custom or a defined Stereotype, either of type *Technology*, *Mission Profile* or *Application*. These definitions go loosely hand in hand with the content of the Mission Profile standard (MPFO). Additionally Blocks (SP) can own zero to any number of properties (defined by a name value and a unit). Blocks (SP) have two predefined properties: *Availability* and *Feasibility* properties. *Solution Space Descriptions* (using PMML) can be added to blocks to allow multidimensional adjustments on the property variable values. A detailed description of each attribute can be found in the Block (SP) description.

Two types of relations exist: *Channels* and *Arrow Relations*. The abstract base class *Relations (SP)* defines the basis of both. It contains an *HTMLDiv* for adding information, a *version* number, a *discussion* chat and a *text* label. Unlike relations on other Perspectives, these *Relations (SP)* are rather complex and independent elements. *Channels* represent an information exchange between two Blocks (SP). *Arrow Relations* are used for any other (often less-complex) types of relation that shall be represented by an arrow. The *Effect* relation extends the arrow relation by an *endpointType* and an *effectType*. Worth to note is that relation *Decomposition* and *Refinement* are not supported to keep the model simple. Use the properties to specify additional information instead. In a similar direction, analysis like *Block Interface* - *Channel* - *Block Interface* consistency is not supported. Such sophisticated analysis is rarely used in abstract innovation descriptions. These are kept for the development phases.

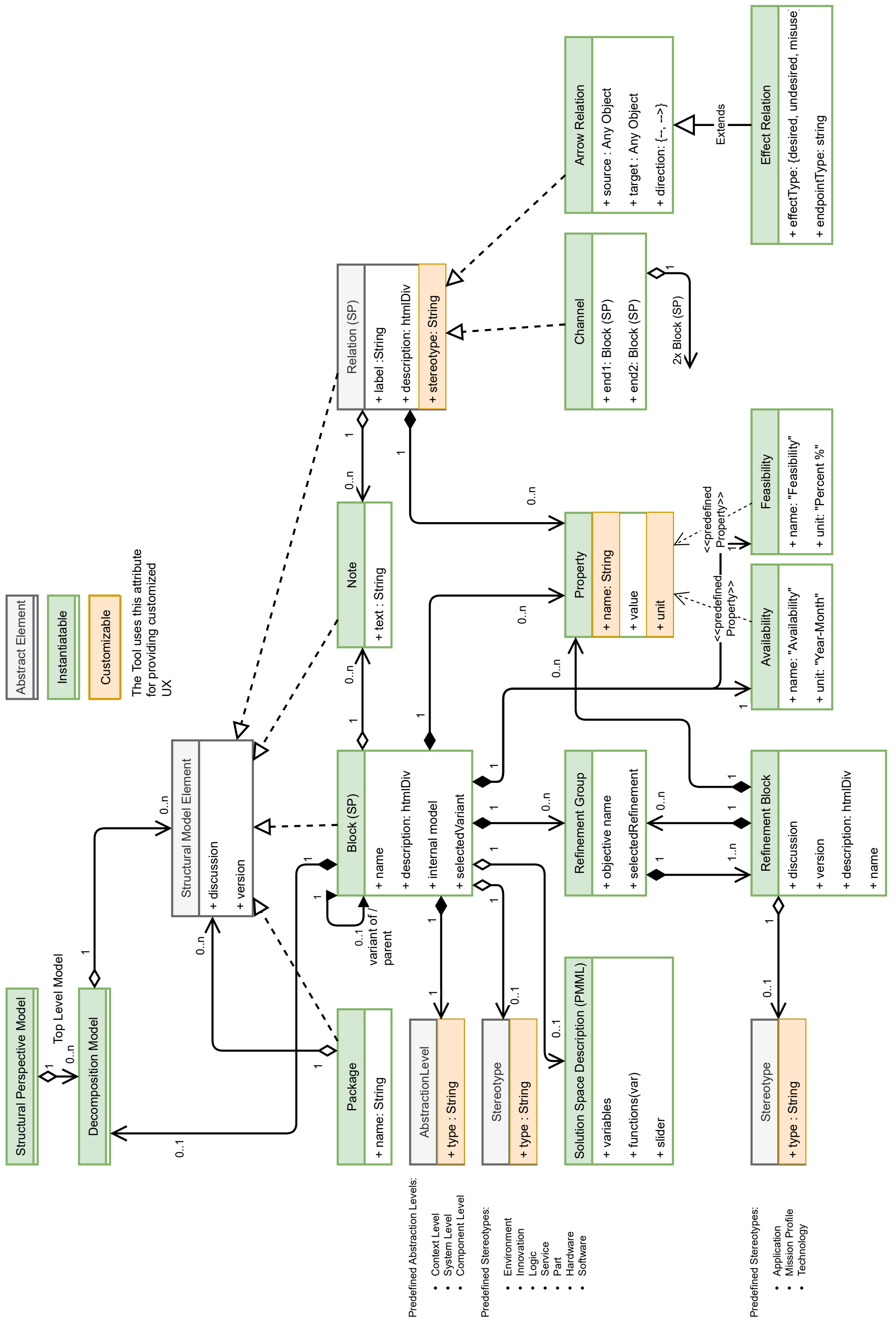
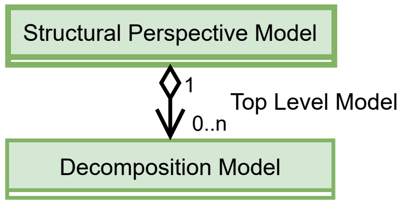


Figure 8.2.: The model elements of the Structural Perspective.

Meta Model Base:

- ↗ Structural Perspective Model
- ↗ Decomposition Model
- ↗ Structural Model Element
- ↗ Relation (SP)

Meta Model Element:

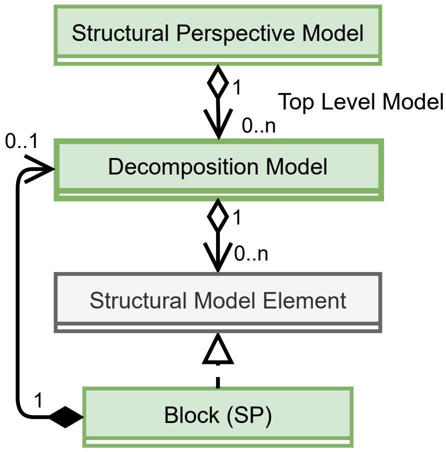


Description:

Structural Perspective Model
The *Structural Perspective Model* is the diagram of the Structural Perspective of an innovation. It contains a *Decomposition Model* which contains all model elements of the Structural Perspective.

Example: A full Structural Perspective Model example is shown in Section 8.2.

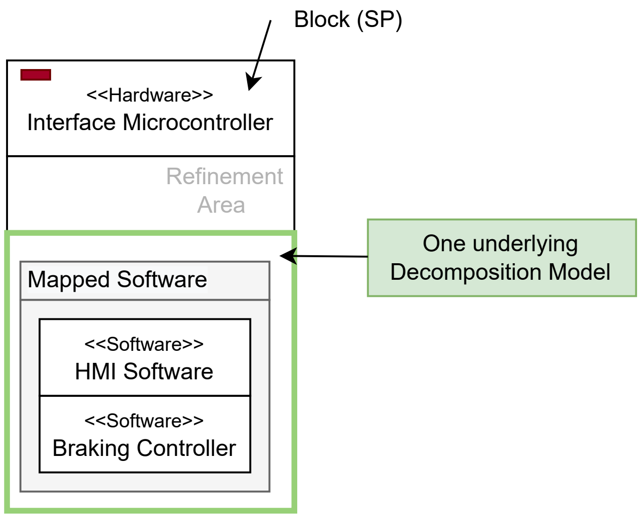
Meta Model Element:



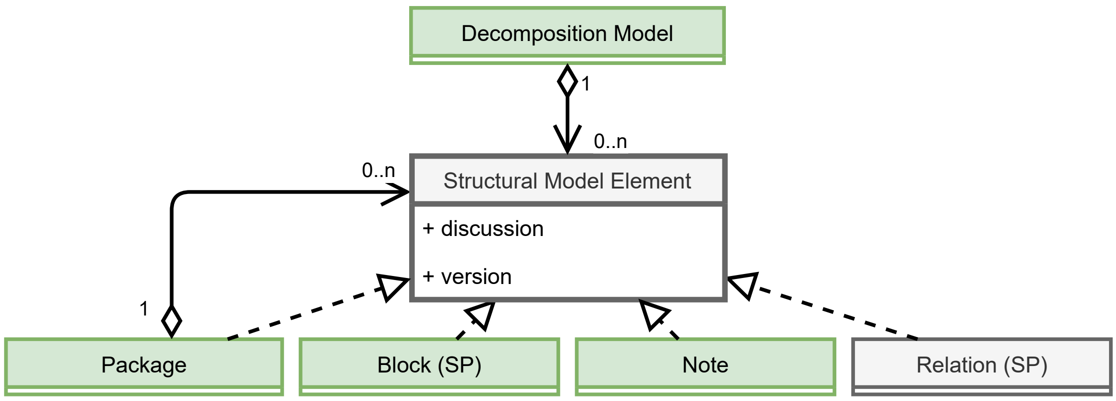
Description:

Decomposition Model
The *Decomposition Model* represents one of the major concepts (*Decomposition, Refinement, Variation*) and is fundamental for the modeling activities. The Decomposition Model can be thought of as the main canvas to draw the innovation on. The Decomposition Models consist of any number of *Structural Model Elements*, which include *Blocks* (of the Structural Perspective), *relations* between them, as well as *Notes* and *Packages*. It is used for describing the top level model of the overall Structural Perspective. Each Block can have its own Decomposition Model and thus the Blocks with their Decomposition Models span up a hierarchy.

Example:



Meta Model Element:



Description:

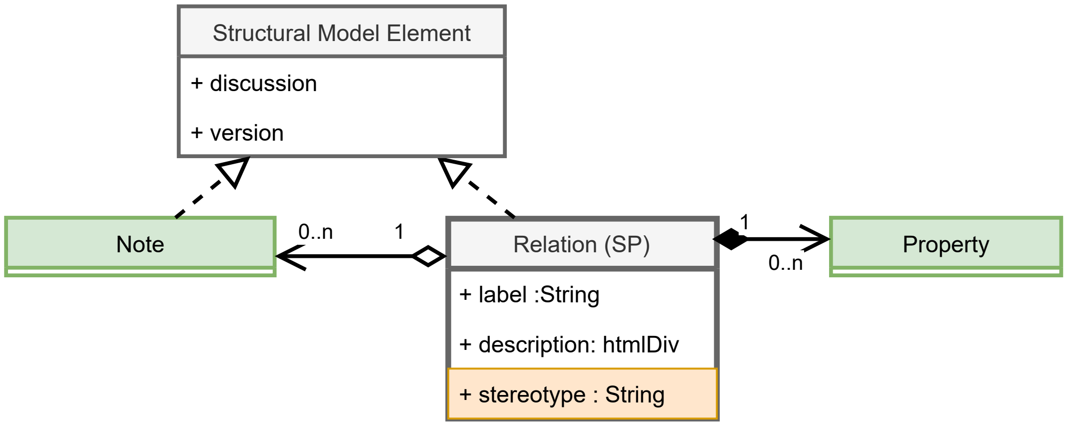
Structural Model Element)

The *Structural Model Element* is the abstract object any model element of the *Decomposition Model* derives: *Package*, *Block (SP)*, *Note* and *Relation (SP)*. It owns the following attributes:

- The *discussion* attribute to take comments about this element.
- The *version* attribute to support version control.

Example: An example is shown for each Structural Model Element under their respective description.

Meta Model Element:



Description:

Relation (SP) **(Read: Relation on the Structural Perspective)**
The abstract *Relation (SP)* describes relations between *Blocks (SP)*. Relations are implemented by *Channels* and *Arrow Relations*. In contrast to relations on the Functional Perspective or Quality Perspective, *Relations (SP)* can own labels and the following attributes:

- A *description* to solve the problem of lack of clarity of the relation by adding information. The description shall answer shortly "What shall the Relation represent?", the reasoning behind the relation and its basic conditions to work. There is no template needed. Images or drafts provide valuable information.
- A *stereotype* to define what type of relation it represents. The stereotype is not defined to be part of any set values, but can hold any string. This way, the stereotype can be used for describing any customized relation.
- A set of *Properties* can be used to specify the relation and to form a basis for any consistency analysis.
- Additionally the derived attributes *discussion* and *version* from the *Structural Model Element* as well as *Notes* to enrich the relations comprehensibility.

Each of the implementing relations are described in more detail on its own.

Example: An example is shown for each relation under their respective description.

Model Elements:
Block related elements:

- Block (SP)
- Package
- Note
- Refinement Group
- Refinement Block
- Solution Space Description
- Property

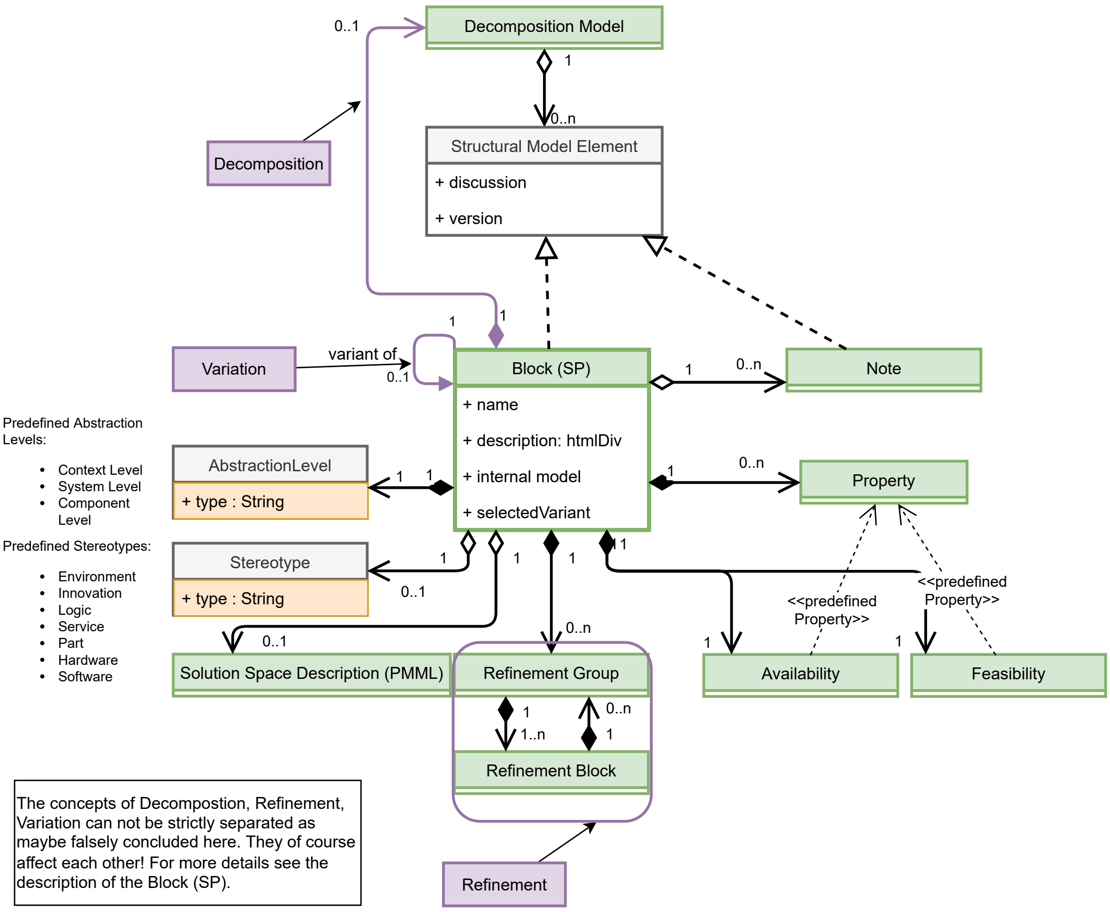
Relation Types:

- Channel
- Arrow Relation
- Effect Relation

In the following the model elements are introduced. First the seven Block relation elements are introduced and then the relations are presented.

Block related elements

Meta Model Element:



Description:

Block (SP) (Read: Block on the Structural Perspective)

The *Block (SP)* is one of the main elements of the Structural Perspective. It is used to represent any system and component that is modeled and is thus the most complex element. It derives the following attributes of the *Structural Model Element* definition:

- The *discussion* attribute to take comments about this element.
- The *version* attribute to support version control.

In addition to the Structural Model Element definition, it contains the following attributes:

- A *name*.

Block (SP) continued (Read: Block on the Structural Perspective)

- A *description* of the Block to solve the problem of lack of clarity by adding information. The description shall answer shortly "What shall the Block represent?", the reasoning behind the Block and its basic conditions to work. There is no template needed. Images or drafts provide valuable information.
- An *abstraction level* to define the level of abstraction the Block represents. It can be either a predefined abstraction level (*Context Level, System Level, Component Leve*l) or any other string.
- An optional *Stereotype*. For any *Block (SP)*, this could be any custom string or one of the predefined Stereotypes:
  - *Environment* for representing the environment of the innovation.
  - *Innovation* for representing the main focus in this model: the innovation.
  - *Logic* for representing a functional unit that is undefined if it is implemented as a physical unit, Hardware or Software.
  - *Service* for representing actions of executing some functionality requested by users.
  - *Part* for representing any physical unit including materials.
  - *Hardware* for representing electrical units that can execute algorithms.
  - *Software* or representing algorithms.
- Optional *Properties* to specify the Block in more detail and to form a basis for consistency analysis. Predefined properties are:
  - An *Availability* property for the estimation of the availability of the Block or in other words, "To which timestamp is the component available?".
  - A *Feasibility* property for the estimation of the feasibility if the Block is available and implementable to the given *Availability* timestamp.
- An optional *Solution Space Description* (in form of PMMLs) to represent important dependencies between the Blocks properties. The properties and the description form together the solution space.
- An optional *internal model* providing more specification details to enhance the Block description.
- The attribute *selected variant* represents if and which variant is chosen.

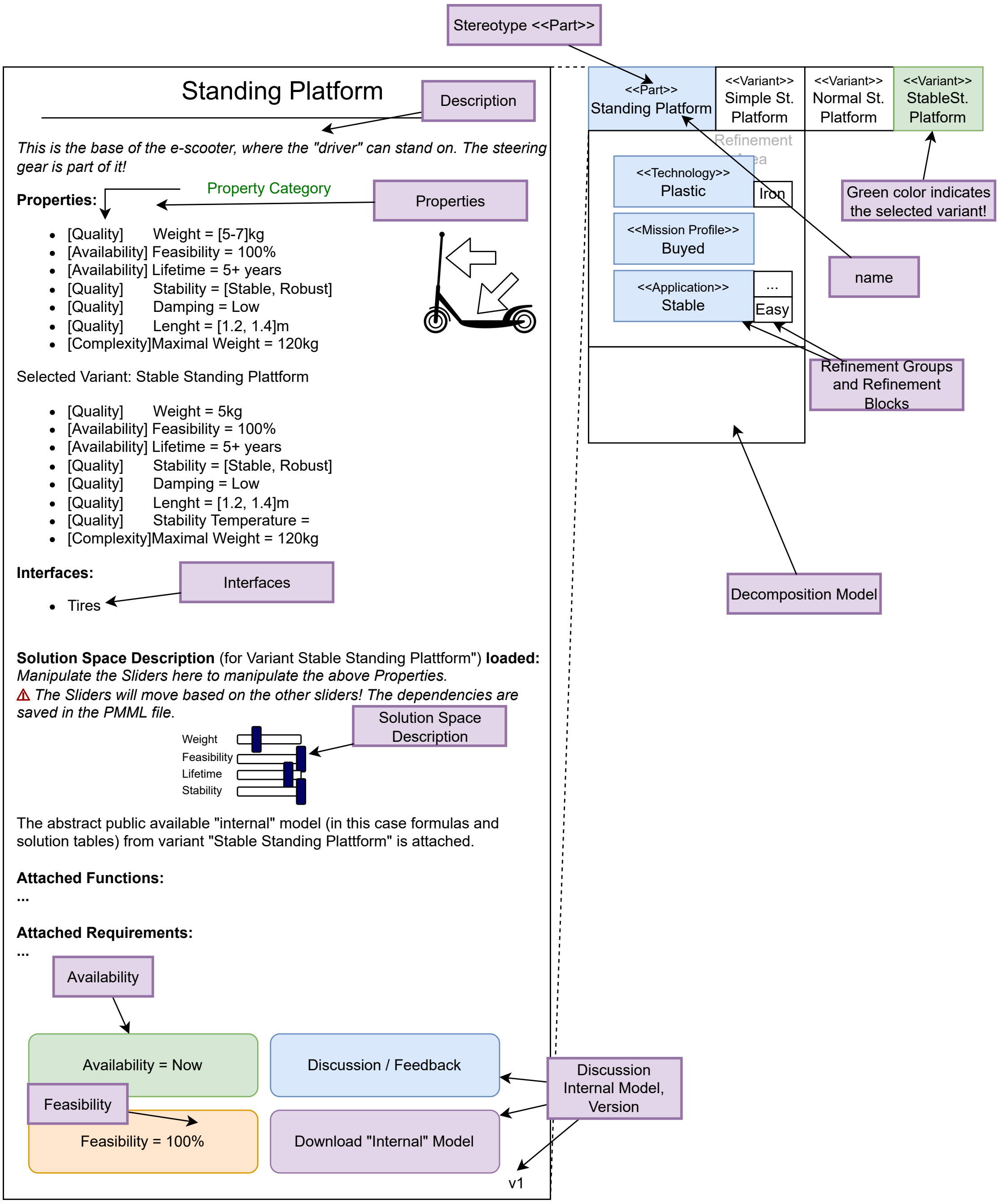
The Block (SP) supports the three main concepts (Decomposition, Refinement, Variation) in the form of more complex attributes:

- An optional *Decomposition Model* to model how the Block is set up. This element is mainly used to build the system model hierarchy. By representing the concept of 'Decomposition' it is especially important.
- An optional set of disjoint *Refinement Groups*, to model possible refinements of the Block's properties. The *Refinement Groups* can contain several *Refinement Blocks* to model varieties of the properties. The set of properties defined among all Refinement Groups and the block itself shall be disjoint!

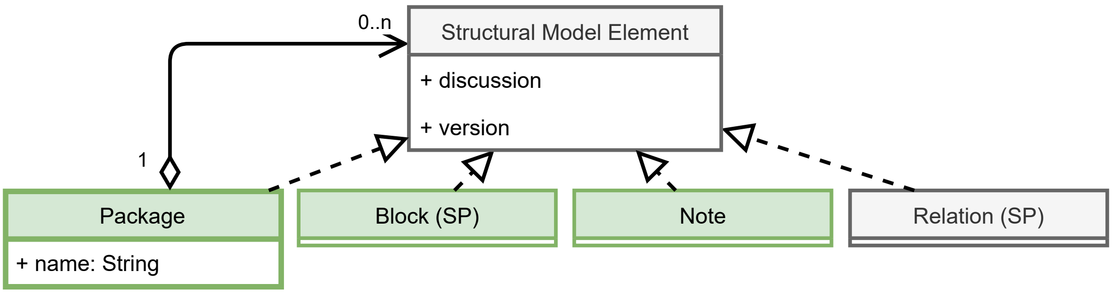
Block (SP) continued (Read: Block on the Structural Perspective)

- The *variant* attribute defines a set of variants - which are itself Blocks (SP) too - to represent deviations from the Block. Meanwhile each variant sets its parent reference to the block. A variant can thus only be part of one parent block! Variants represent the implementation of the concept of 'Alternatives' for the Structural Perspective. Selecting a variant of a Block affects the Blocks attributes in the following way:
  - *Overwriting (but the Block's original value shall be still represented in the tool)*:
    * The *name* of the variant overwrites the Block's name. E.g: The variant name 'Comfort E-Scooter' overwrites the Block's name 'E-Scooter'.
    * Same holds for the *discussion, version, description, abstraction level* and *Stereotype*.
  - *Extended:*
    * The *properties* of variant extend the Block's properties. If the variant property name is the same as the property of the block, then the variant property overwrites the blocks property. E.g: Property 'Weight' is set for the 'E-Scooter' and the variant 'Comfort E-Scooter'. In this case, the weight property from the 'Comfort E-Scooter' overwrites the weight property from the block 'E-Scooter'.
    * The *Solution Space Description* (SSE) of the variant extends the blocks SSE, if and only if both SSEs have at maximum one common property. If both SSEs have more than one common property (e.g: 'weight' and 'speed') then only the SSE from the variant is considered (because there is typically no function satisfying both SSEs).
    * The *Decomposition Model* of the variant extends the Decomposition model of the block. There is nothing like overwriting here. If both - variant and block - have a block with the same name in their decomposition model, then they are both considered! Because of the strict variant / parent reference, variants can own relations between blocks of their decomposition model and any blocks of decomposition model of the parent block! This makes the selection of variants a pretty powerful tool to change things!
    * The *Refinement Groups* and *Refinement Block* of the variant extends the Refinement Groups of the block if the Refinement Groups name are not the same. If any Refinement Group exists in both variant and block, then the variant Refinement Group overwrites the Refinement Group from the block.
  - *Unaffected:*
    * *Internal models* of variants take precedence in the list. However, because Internal models are only referenced and not used in the modeling methodology IMoG, there is no clear overwriting or extending in place. The user shall consider both or just one based on their needs.
    * The *selected variant* stays of course relevant (otherwise it would be unclear which variant properties affect the block!). If the variant itself has this property set and its own attributes are affected by their own variants, then this selection among the variants of the variant stays relevant too!

Example:


Stereotype <<Part>>
Standing Platform
Description
This is the base of the e-scooter, where the "driver" can stand on. The steering gear is part of it!
Property Category
Properties:
Properties
[Quality] Weight = [5-7]kg
[Availability] Feasibility = 100%
[Availability] Lifetime = 5+ years
[Quality] Stability = [Stable, Robust]
[Quality] Damping = Low
[Quality] Lenght = [1.2, 1.4]m
[Complexity]Maximal Weight = 120kg
Selected Variant: Stable Standing Plattform
[Quality] Weight = 5kg
[Availability] Feasibility = 100%
[Availability] Lifetime = 5+ years
[Quality] Stability = [Stable, Robust]
[Quality] Damping = Low
[Quality] Lenght = [1.2, 1.4]m
[Quality] Stability Temperature =
[Complexity]Maximal Weight = 120kg
Interfaces:
Interfaces
Tires
Solution Space Description (for Variant Stable Standing Plattform") loaded:
Manipulate the Sliders here to manipulate the above Properties.
⚠ The Sliders will move based on the other sliders! The dependencies are saved in the PMML file.
Solution Space Description
Weight
Feasibility
Lifetime
Stability
The abstract public available "internal" model (in this case formulas and solution tables) from variant "Stable Standing Plattform" is attached.
Attached Functions:
...
Attached Requirements:
...
Availability
Availability = Now
Discussion / Feedback
Feasibility
Feasibility = 100%
Download "Internal" Model
v1
Discussion Internal Model, Version
<<Part>>
Standing Platform
<<Variant>>
Simple St. Platform
<<Variant>>
Normal St. Platform
<<Variant>>
StableSt. Platform
Refinement
<<Technology>>
Plastic
Iron
Green color indicates the selected variant!
<<Mission Profile>>
Buyed
name
<<Application>>
Stable
...
Easy
Refinement Groups and Refinement Blocks
Decomposition Model

Meta Model Element:

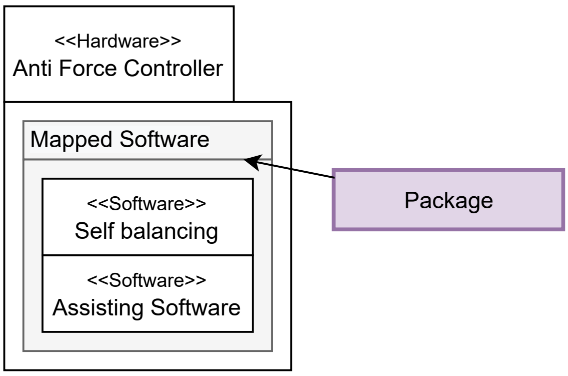


Description:

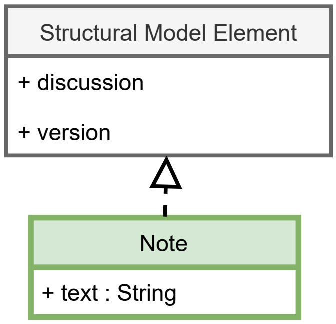
Package

The *Package* describes an collection of Structural Model Elements to allow to create a hierarchy in the Decomposition Model and distinct between Packages. There is nothing special about Packages otherwise: They have a *name* and a set of *Structural Model Elements*.

Example: A package named 'Mapped Software' with two blocks inside.



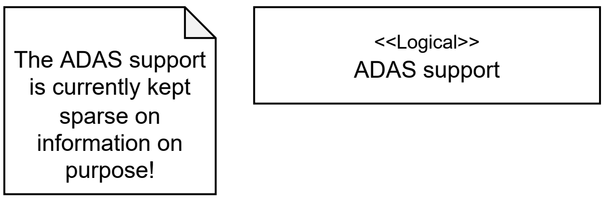
Meta Model Element:



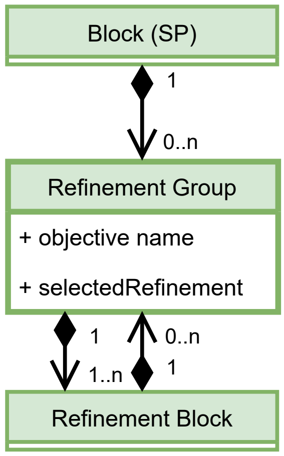
Description:

Note
The *Note* can be used to add information to the model that can not or should not be modeled. Notes should be used sparsely!

Example:



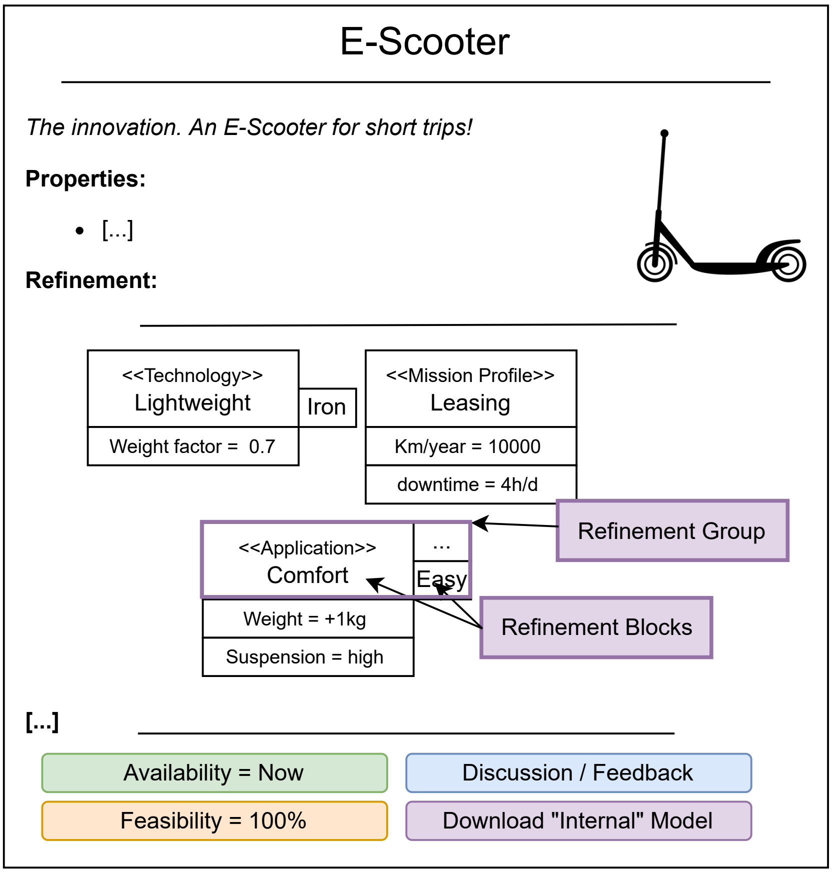
Meta Model Element:



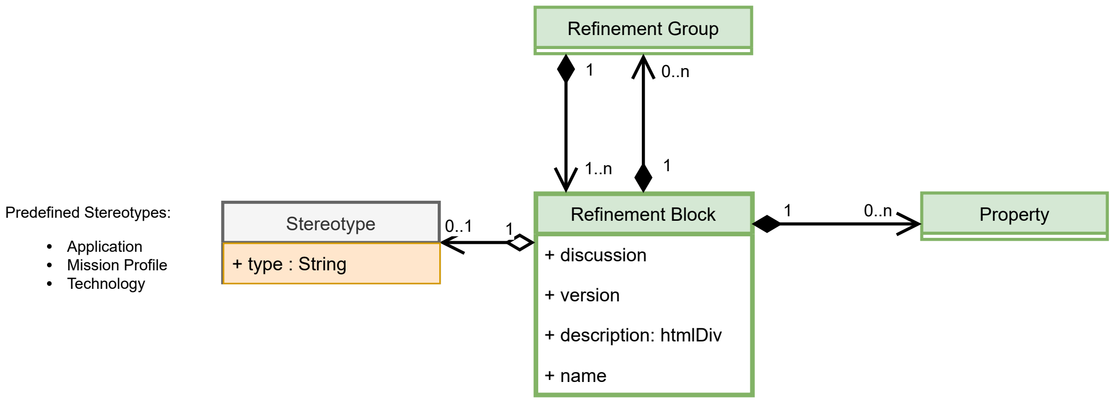
Description:

Refinement Group
The *Refinement Group* represents a container for modeling variety of Refinements. It contains a non empty set of *Refinement Blocks* and a marker (called *selectedRefinement*), which of the *Refinement Blocks* is currently selected. *Refinement Groups* can not exist on their own. They must be included by a Block (SP) or a Refinement Block. The Blocks (SP) and Refinement Blocks may contain none or multiple Refinement Groups. The Refinement Blocks inside a container are recommended, but not restricted to have the same stereotype. It makes much sense to group only blocks with the same stereotype instead of mixing say *Application* specifications with *Technology* specifications. The selected Refinement Block will be used to overtake Properties to the Block (SP) specification or to the Refinement Block specification, which are then used for consistency analysis.

Example: An example Refinement Area with one Refinement Group containing two «technology» stereotyped Blocks: 'Copper' and 'Iron'



Meta Model Element:



Description:

Refinement Block

The *Refinement Block* represents 'Refinements' of Blocks (SP). Refinement Blocks represent the 'Refinement' concepts and are thus of special importance. Refinement Blocks are a lighter variant of the Block (SP) and own fewer common attributes:

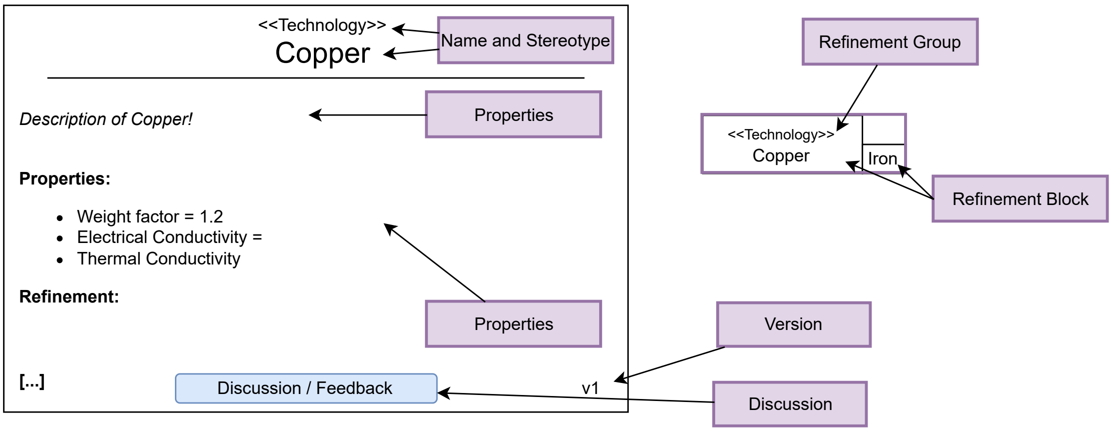
- A *discussion* attribute to take comments about this element.
- A *version* attribute to support version control.
- A *description* of the Block to solve the problem of lack of clarity by adding information. The description shall answer shortly "What shall the Block represent?", the reasoning behind the Block and its basic conditions to work. There is no template needed. Images or drafts provide valuable information.
- A *name*.
- Optional *Properties* to specify the Refinement Block and to form a basis for consistency analysis.

In addition to the common attributes above, the Refinement Block may contain an optional *Stereotype*: For any Refinement Block this could be a custom string or one of the predefined Stereotypes:

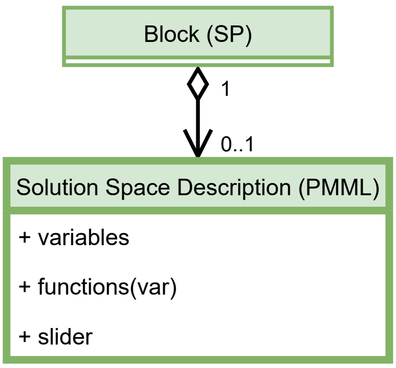
- *Technology* for representing information about influencing factors of technology and materials of the innovation.
- *Mission Profile* for representing environmental factors that influence the decision. It is especially used for restricting the environment to a special set of situations.
- *Application* for representing specialties about how the system or component is used.

Additionally, Refinement Blocks can be further refined. This can be achieved by adding *Refinement Groups* with Refinement Blocks to the block. Important to note is, that Refinement Blocks can not exist on their own and thus must be part of a Refinement Group. Refinement Groups are either owned by Blocks (SP) or Refinement Blocks. Thus it inevitable follows that any Refinement Block has on the highest level an Block (SP) as an owner.

Example:



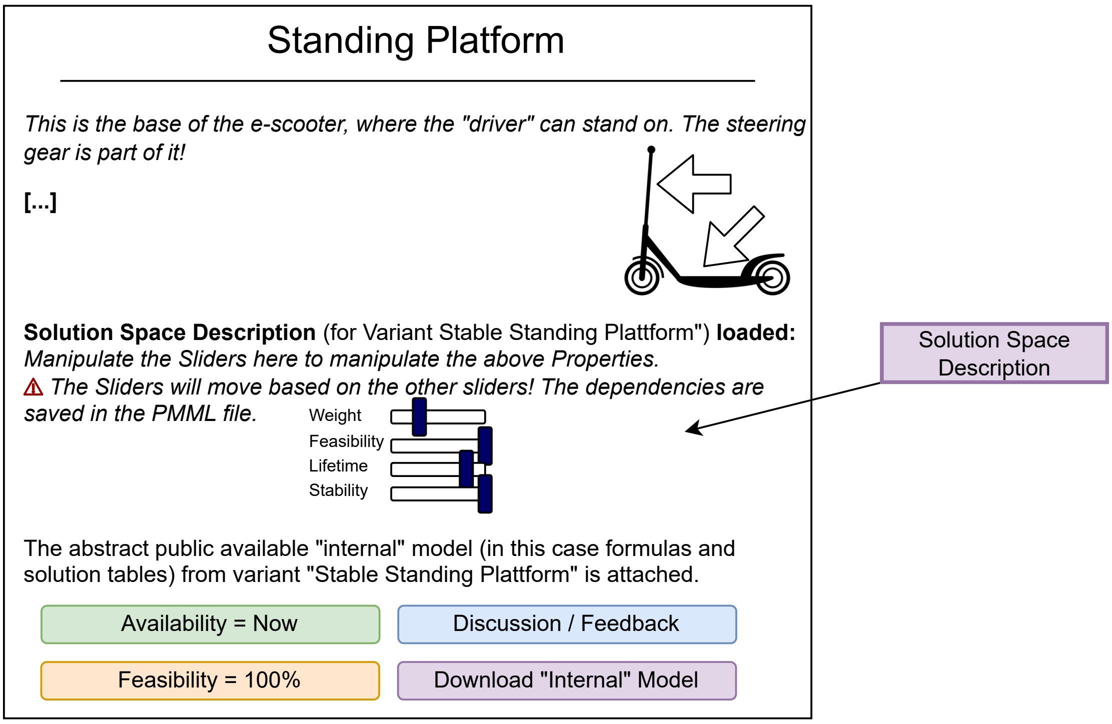
Meta Model Element:



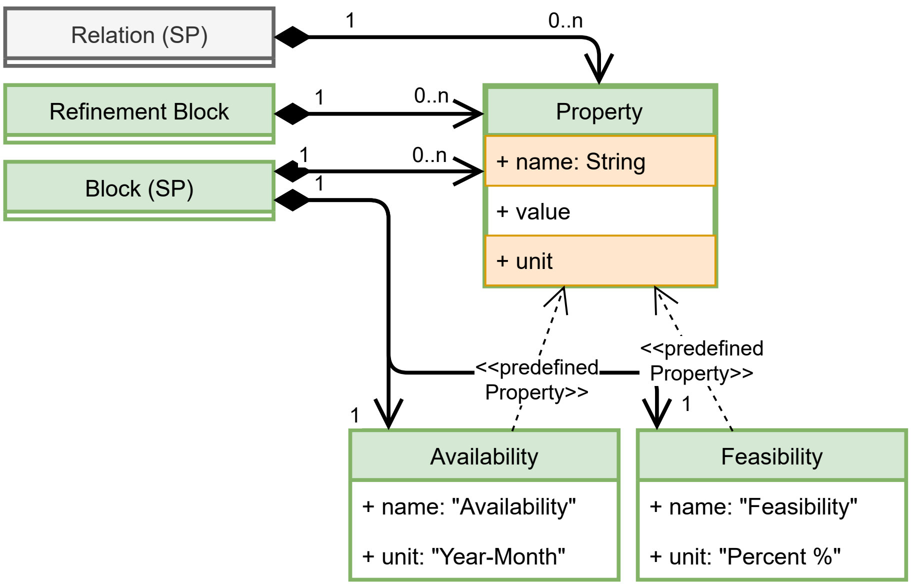
Description:

Solution Space Description (PMML)
The *Solution Space Description* enables the user to add additional functions how the Block properties relate to each other. The format is PMML and it describes the relation of a set of input variables to a set of output variables as multivariate functions. The chosen values shall be manipulate able by a graphical slider representation.

Example:



Meta Model Element:



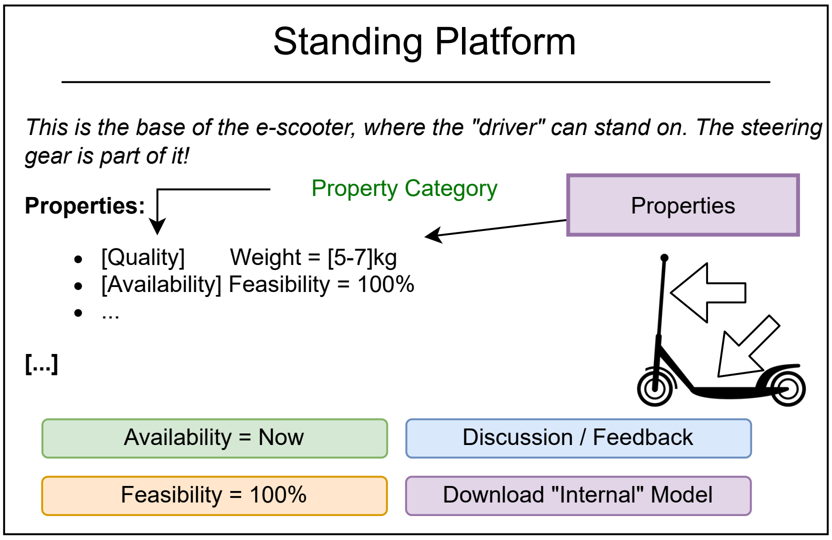
Description:

Property
Properties are part of the specification of Blocks (SP), Refinement Blocks and Relations (SP). Properties are used to provide important information and build a basis for further analysis. Properties can not exist on their own and must have any of the above mentioned elements as an owner. Two properties are predefined in the meta model: The *Availability* and the *Feasibility* properties are defined for every Block (SP):

- The *Availability* property is used for the estimation of the availability of the Block or in other words, "To which timestamp is the component available?".
- The *Feasibility* property is used for the estimation of the feasibility if the Block is available and implementable to the given *Availability* timestamp.

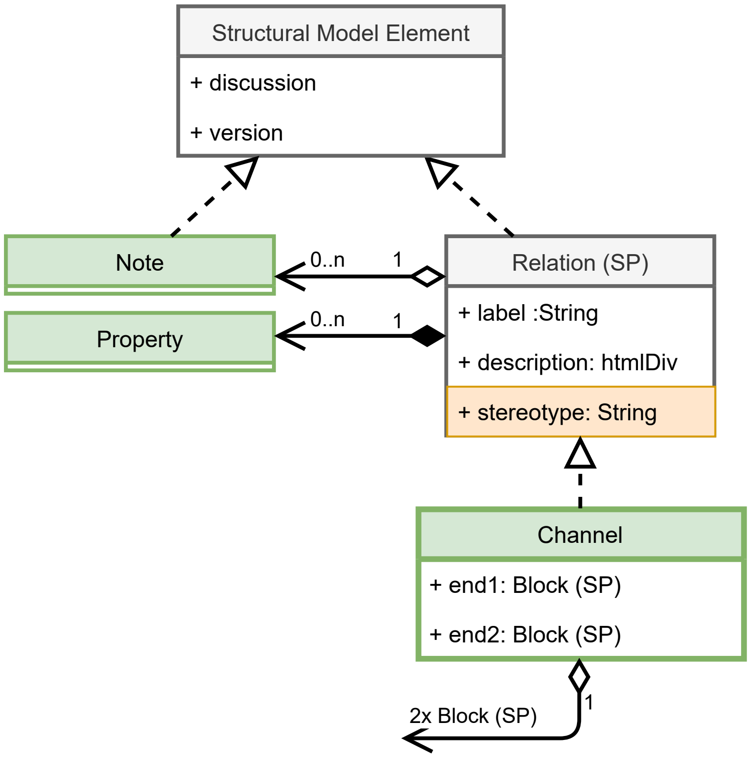
Properties own a *name*, a *value* and a *unit*. The *name* and *unit* are not preset to a specific set of values. They can be used for any customization to allow the user to add innovation specific properties. For any domain and innovation it is recommended to have a existing 'domain properties' set to build upon instead of starting with just the two predefined properties.

Example:



Relations

Meta Model Element:



Description:

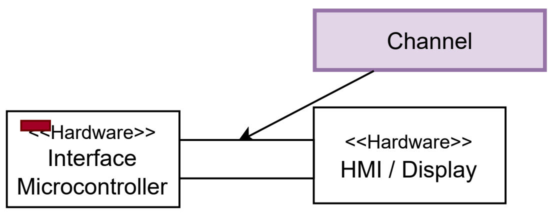
Channel

The *Channel* represents a communication medium between Blocks (SP). Channels are used, whenever the communication between two Blocks (SP) plays a significant role and needs a specification. Channels can trigger communication based analysis. The definition of the Channel relation reflects that the meta model is not pure generic but slightly customized to the microelectronic domain.

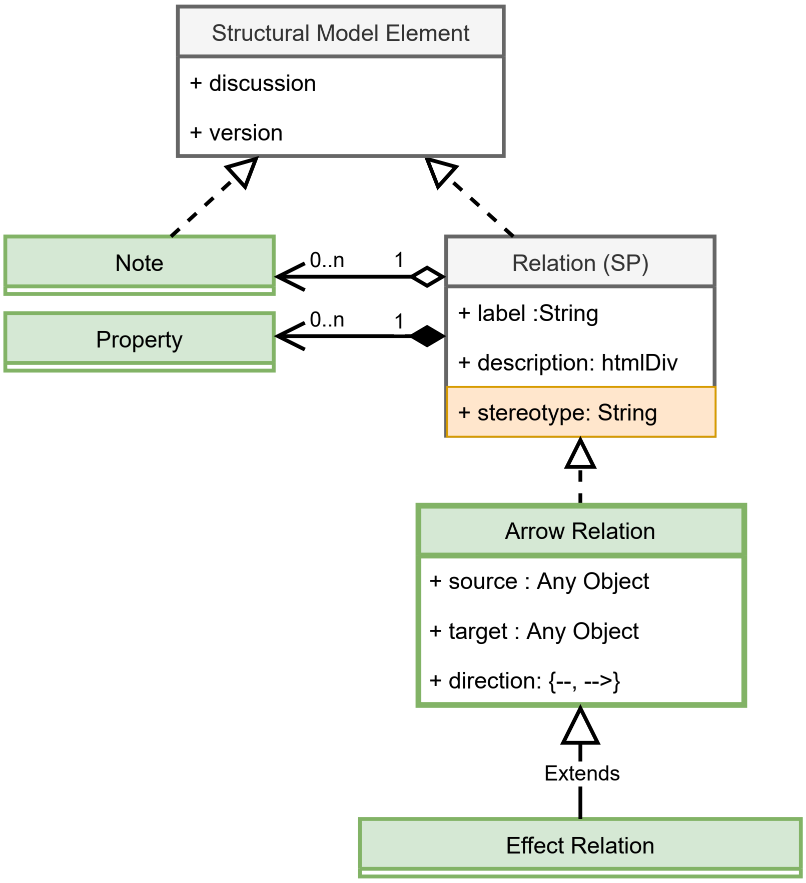
Channels implement the abstract Relation (SP). Channels thus own the following attributes:

- Two *Block (SP)* endpoints.
- A *label* and a *description* to solve the problem of lack of clarity of the relation by adding information. The description shall answer shortly "What shall the Relation represent?", the reasoning behind the relation and its basic conditions to work. There is no template needed. Images or drafts provide valuable information.
- A *stereotype* to define what type of relation it represents. The stereotype is not defined to be part of any set values, but can hold any string. This way, the stereotype can be used for describing any customized relation.
- A set of *Properties* can be used to specify the Relation and to form a basis for any consistency analysis. Additionally the derived attributes *discussion* and *version* from the *Structural Model Element* as well as *Notes* to enrich the relations comprehensibility.

Example:



Meta Model Element:



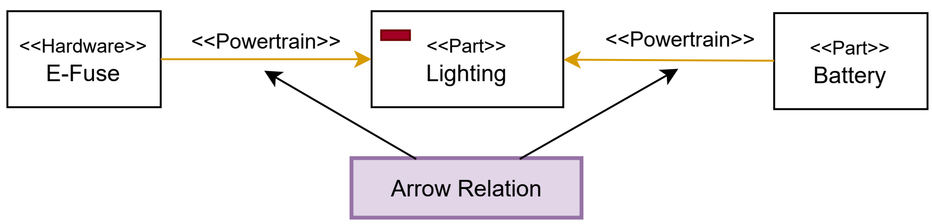
Description:

Arrow Relation

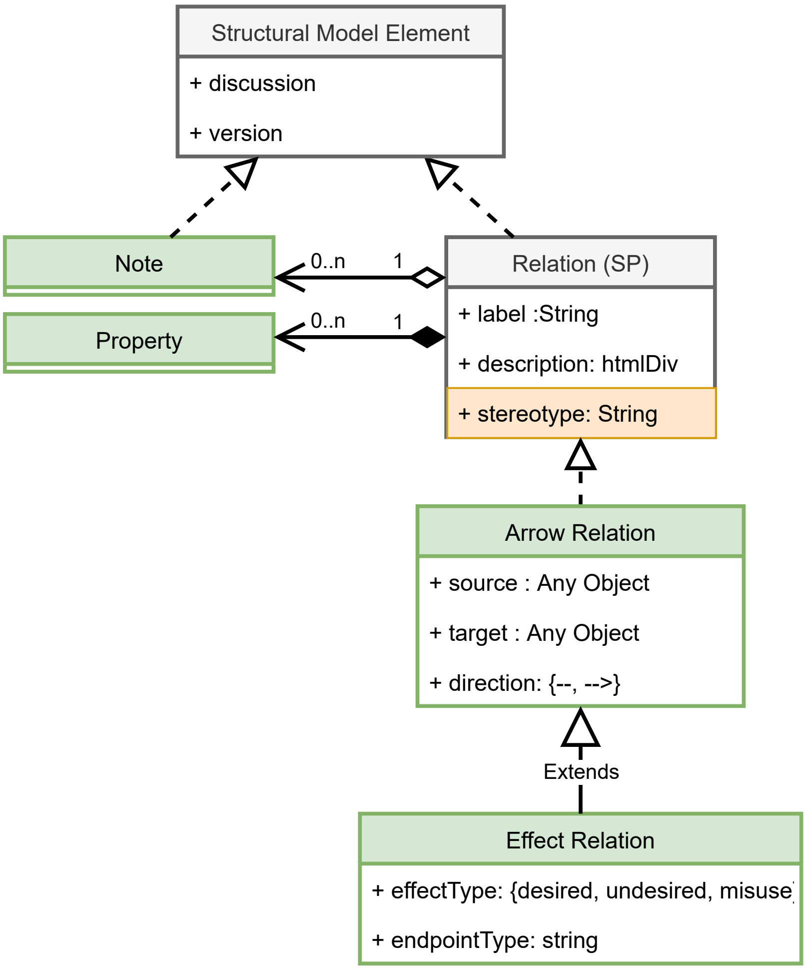
The *Arrow Relation* represents a bidirectional relation between two elements. The Arrow Relation remains pretty generic. The Arrow Relation implements the abstract Relation (SP). Arrow Relations thus own the following attributes:

- A *source* and a *target* endpoint.
- A *direction* either of the type *unidirectional* or *bidirectional*.
- A *label* and a *description* to solve the problem of lack of clarity of the relation by adding information. The description shall answer shortly "What shall the Relation represent?", the reasoning behind the relation and its basic conditions to work. There is no template needed. Images or drafts provide valuable information.
- A *stereotype* to define what type of relation it represents. The stereotype is not defined to be part of any set values, but can hold any string. This way, the stereotype can be used for describing any customized relation.
- A set of *Properties* can be used to specify the relation and to form a basis for any consistency analysis. Additionally the derived attributes *discussion* and *version* from the *Structural Model Element* as well as *Notes* to enrich the relations comprehensibility.

Example:



Meta Model Element:



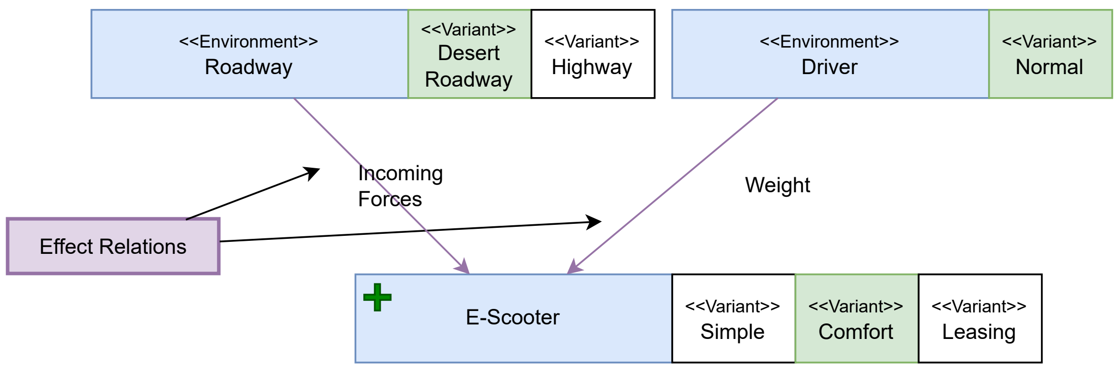
Description:

Effect Relation

The *Effect Relation* represents an effect between two elements. The Effect Relation originates from the well known effect chain analysis and can be used in a similar manner. Special types of Effect Relations can be either implemented by customizing the *label* attribute or by creating a new relation category by implementing *Custom* relations. The Effect Relation implements the abstract *Relation (SP)*. Effect Relations thus own the following attributes:

- An *effectType*, which can be set to either *desired*, *undesired* or *misuse*.
- A *source* and a *target* endpoint with the target endpoint having a *endpointType* (e.g. 'thermal' or 'acoustic' or 'radiation').
- A *direction* either of the type *unidirectional* or *bidirectional*.
- A *label* and a *description* to solve the problem of lack of clarity of the relation by adding information. The *description* shall answer shortly "What shall the Relation represent?", the reasoning behind the relation and its basic conditions to work. There is no template needed. Images or drafts provide valuable information.
- A *stereotype* to define what type of relation it represents. The stereotype is not defined to be part of any set values, but can hold any string. This way, the stereotype can be used for describing any customized relation.
- A set of *Properties* can be used to specify the relation and to form a basis for any consistency analysis. Additionally the derived attributes *discussion* and *version* from the *Structural Model Element* well as *Notes* to enrich the relations comprehensibility.

Example:



8.2. E-Scooter example

The example of the Structural Perspective comprises the innovation requirements of "Providing mobility with an e-scooter" in one view called Structural View. The example is divided into three sub views:

- The *top view* (see Figure 8.3) describes the solutions of the innovation "Providing mobility with an e-scooter" from a high level.
- The *Block view* (see Figure 8.4) shows – next to the context level elements from the top view – the solution-elements of the system level and the component level. Its purpose targets the exemplary representation of all model elements from the meta model.
- The *Tree view* (see Figure 8.5) represents the model as a tree. This view is useful when the model gets too large to handle visually or for searching purposes.

8.3. Structural Perspective: Strengths and Limitations

The Structural Perspective was the most difficult Perspective to design, because of the vast amount of possibilities to model solutions. The general concept to keep things abstract and simple remains the same for the Structural Perspective. The main design decisions of the other perspectives also hold here:

- The perspective provides also complex filtering mechanisms via abstraction levels and stereotypes.
- The concepts of *Decomposition*, *Refinement* and *Variability* are all well supported.
- The Structural Perspective is constrain able via requirements.
- The Structurl Perspective is still not part of the development or design phases!
- The Structural Perspective bases on Model Based Systems Engineering.

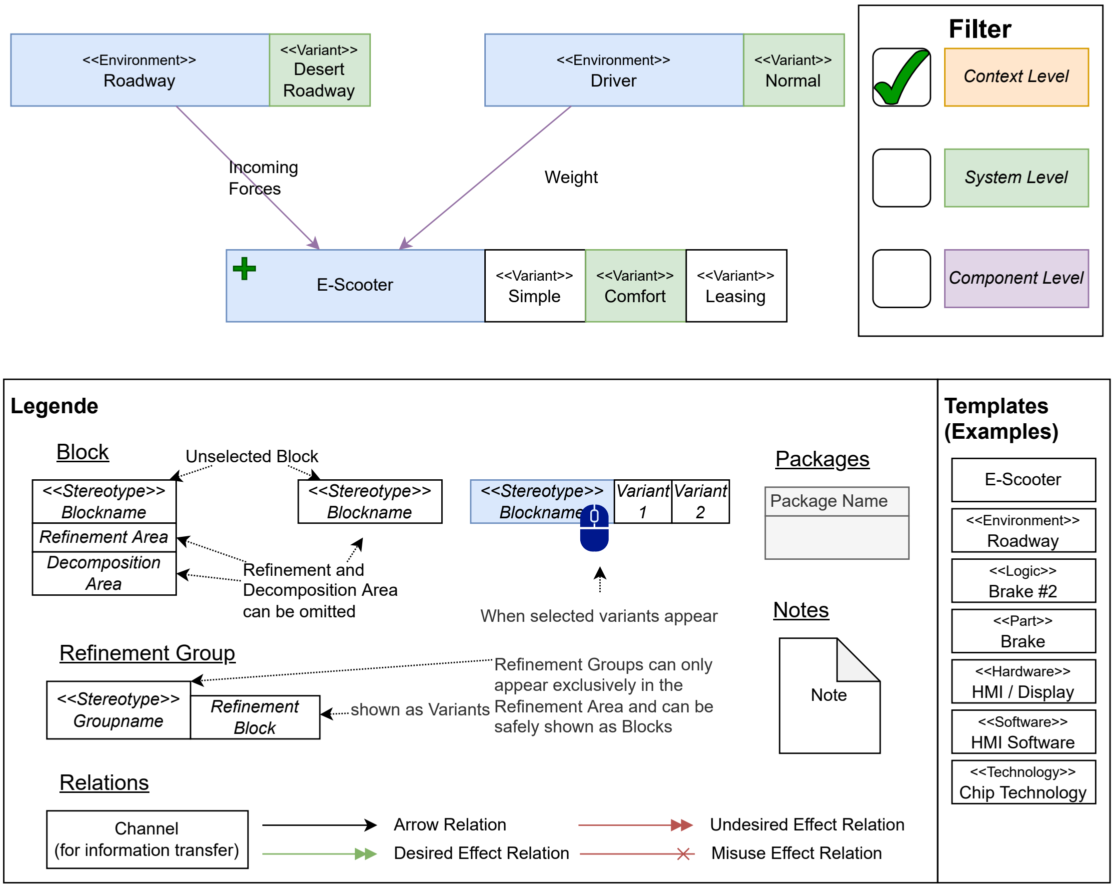


Figure 8.3.: **Top view: The solutions of the innovation “Providing mobility with an e-scooter” from a high level.**

The focus on the Structural Perspective is also to go not too deep into the behavior to keep solutions abstract enough for innovation modeling. In line with this abstract focus, the Structural Perspective also contains no complex elements like Ports for interface consistency checks. Nonetheless, abstract communication modeling is supported via channels and effect chains. The solution spaces itself can be well assessed via the use of Key Performance Indicators, which are simply chosen properties of Blocks and relations. The Structural Perspective also supports properties for Blocks and relations and provides more than only ’structure’.

As limitations, there are only few elements supported to model domain specific specialties. These domain specific elements should however be added based on the needs of the innovation. There exists also the high risk of ’model explosion’ making the model unmaintainable. However, innovation modeling tends to be abstract and thus manageable.

Overall, the Structural Perspective seems to be in a good shape for innovation modeling.

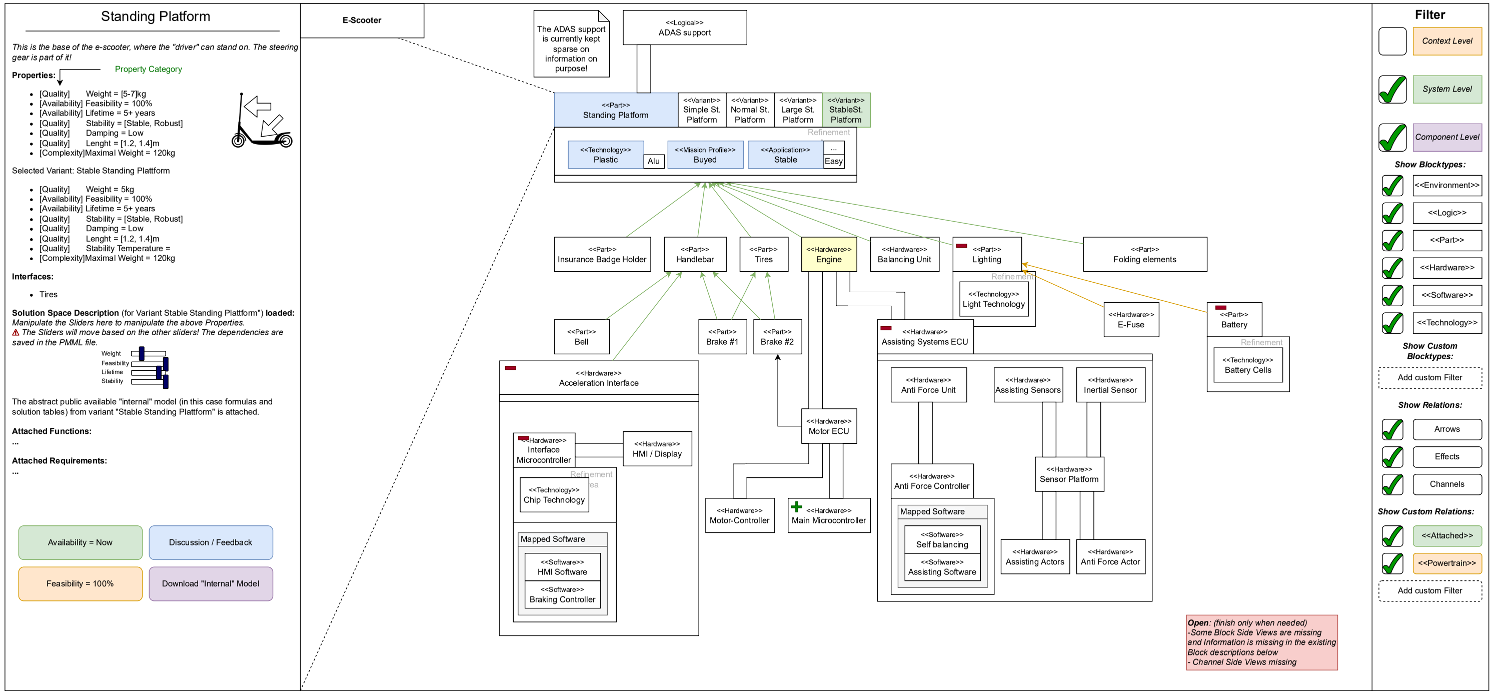


Figure 8.4.: **Block view: The solutions for the innovation of "Providing mobility with an e-scooter" of the solution-elements of the system level and the component level.**

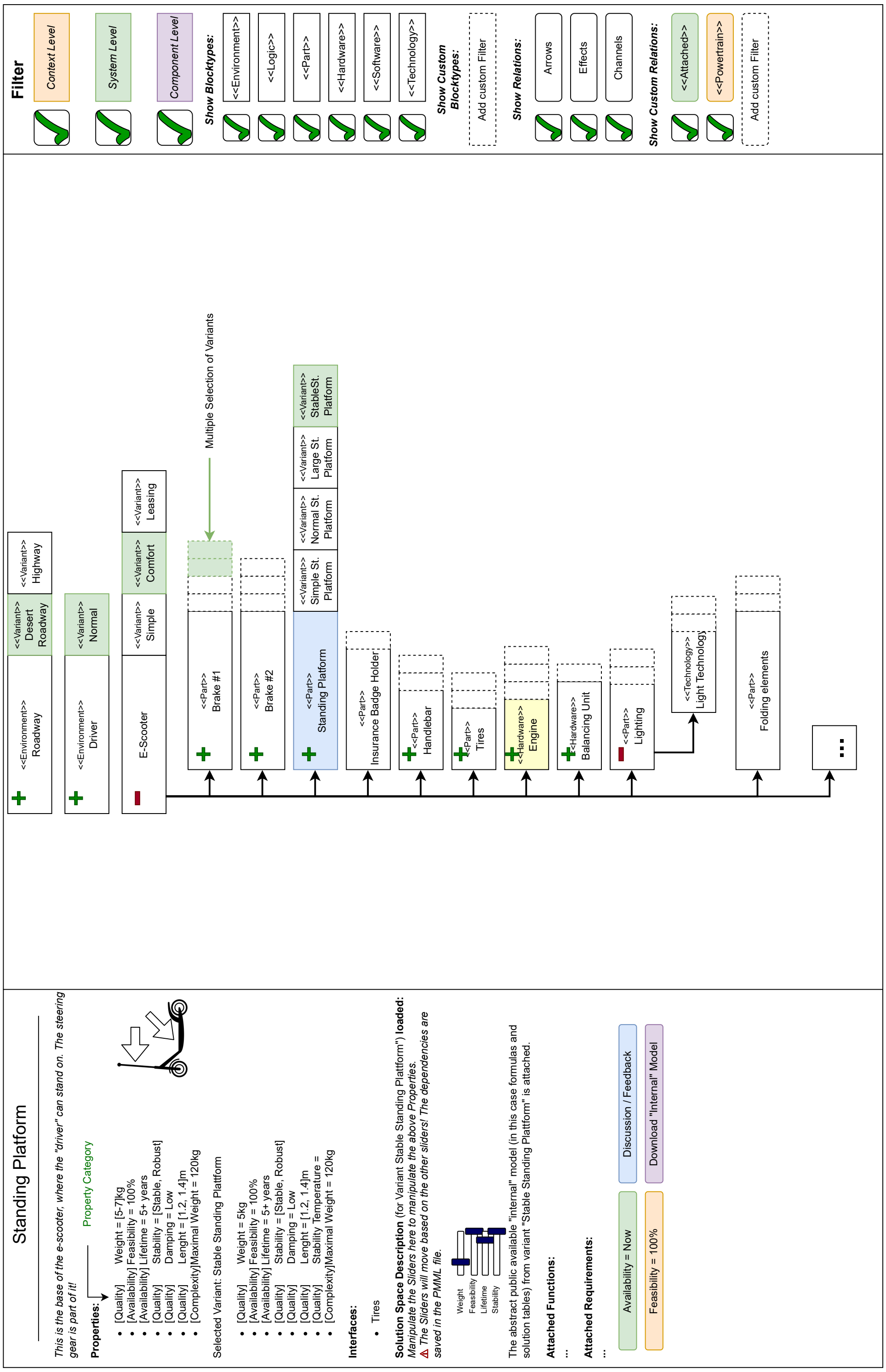


Figure 8.5.: **Tree View.**

8.4. Structural Perspective FAQ

Can a Refinement Block be decomposed into Blocks (SP)?

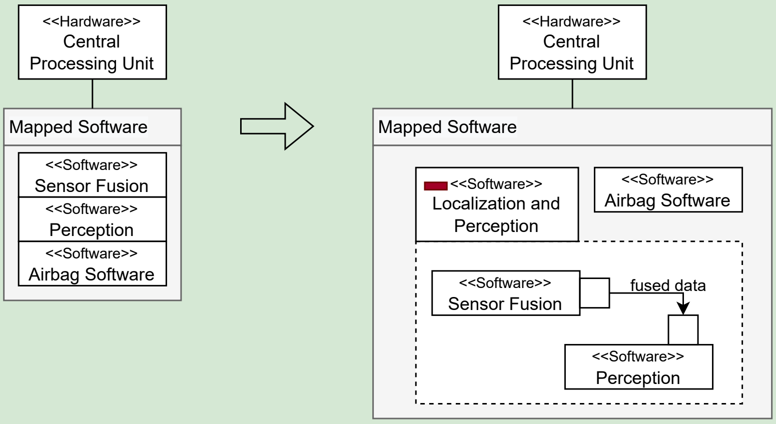
No, the meta model does not allow this.

How to define connections between Software Blocks?

Software Blocks appear at two places in the Structural Perspective:

1. In the Mapped Software containers. Here they are listed as Sets of Blocks where no interconnection can be drawn.
2. In the decomposion of some mapped Software Block. Here interconnection can be drawn.

Example: An example may be three Blocks of Software that are mapped on a Central Processing Unit: A Sensor Fusion Block, a Perception Block and an independent Airbag Software Block. A connection between the Sensor Fusion and the Perception shall be added. In the Mapped Software representation no interconnection can be drawn. One way to cope with this situation is by adding a Software Block that generalises the Sensor Fusion and Perception Software. This Block could be named Localization and Perception Software. The Localization and Perception Software is then decomposed into the Sensor Fusion and Perception Blocks with the new connection and ports added.



What is the commonality and difference between the Refinement and the Variability Concept for Blocks in IMoG and what is the purpose for this differentiation?

In general, a refinement of a given Block represents a fuller specification of this given Block (e.g. by adding a parameter).
In general, variability describes a variable parameter of a specification of a Block.
In fact, variability can be seen as multiple different refinements for a variable parameter of a common abstract specification of a Block.

Example 1: "Lidar Sensor A" optimized for high accuracy and "Lidar Sensor B" optimized for a high range are variations of characteristics of an abstract "Lider Sensor" specification.
Example 2: "Zone Architecture A" optimized for low delay in the network and "Zone Architecture B" optimized for a high bandwidth are variations of characteristics of an abstract "Zone Architecture" specification.
Example 3: "Electronic Brake" and "Manual Brake" are variations of the used technology of an abstract "Brake" specification.

The question of the difference between the two concepts and the purpose of this differentiation remains. In fact, variability could be represented by using only the concept of refinements. However, IMoG defines them as orthogonal concepts with two representations: The Refinement Groups and the "variant-of" relation. The variant-of relation allows to represent fuller specifications of a given Block regarding all Block attributes including changes in the properties and changes in the decomposition. Refinement Groups allow to define additional properties among all variants of the abstract Block. The purpose for this definition of refinement is to model several orthogonal refinements that only affect a few properties (and thus represents only small changes regarding all Block variants) without the need to represent them as alternatives with the need to redefine every property and suffer under the explosion of the number of variants. The purpuse for the differentiation is thus to handle the "solution space explosion".

Let's take a look back at the examples. The modeler has still the choice of representation for every example. Each example could be represented with a variant-of relation or with a Refinement Group regarding properties:
Example 1 would be most probably represented with a Refinement Group with the stereotype of "technology", because Sensor A and Sensor B have only different characteristics modeled as properties. A decompostion of these sensors is most probably not required.
Example 2 would be most probably represented with a variant-of relation as each zone architecture would not only differ in some properties, but also in its decomposition that is most probably in the intesest of being modeled too.
Example 3 is a "gray area" example, because both representations are feasible. If the different brakes should be modeled with their parts, then a representation with a variant-of relation is suitable. When these parts are not required to be modeled, then a representation with Refinement Groups might be more suitable!

How does the Block Properties relate to Variants Properties?

➔ Variant Properties overwrite Block Properties if they exists. Block Properties - if not existent in Variants - are inherited.
Example: Block E-Scooter.weight < 20kg is overwritten by Variant Heavyweight.weight < 25kg and Small.weight < 15kg.

Is it possible to do multiple selections of variants? What happens then?

➔ Yes, it is possible to select multiple variants of the same block. It means: I want, that all Variants are checked against the rest of the model variants (environment). With this it is possible to test for example the e-scooter in the environment "suburban" and "hilly alaska" at the same time. The checks are 1x1 and not property hulls.

How to represent Estimations of different stakeholders?

➔ Estimations are representated by using variants and requirements on those variants.

Does some categorization between Properties exists?

➔ Yes, Properties can be categorized in predefined and custom categories. A category does not any semantic and is only used for better usability. Categories are removed when properties are analyzed. Some predefined categories may include

- Quality
- Complexity
- Availability
- System Data
- Function Data
- State Data
- KPI
- ... more custom categories

How can Constraintnets help with analysing solution spaces?

➔ Constraintnets can be used to transform the property lists into constraints and then efficiently check their satisfaction in the whole nets (without the property categories of course).

9. Domain Knowledge Perspective

The Domain Knowledge Perspective is the fifth perspective in IMoG (see Figure 9.1). There has not been any more work done on defining the Domain Knowledge Perspective other than given in Chapter 4.

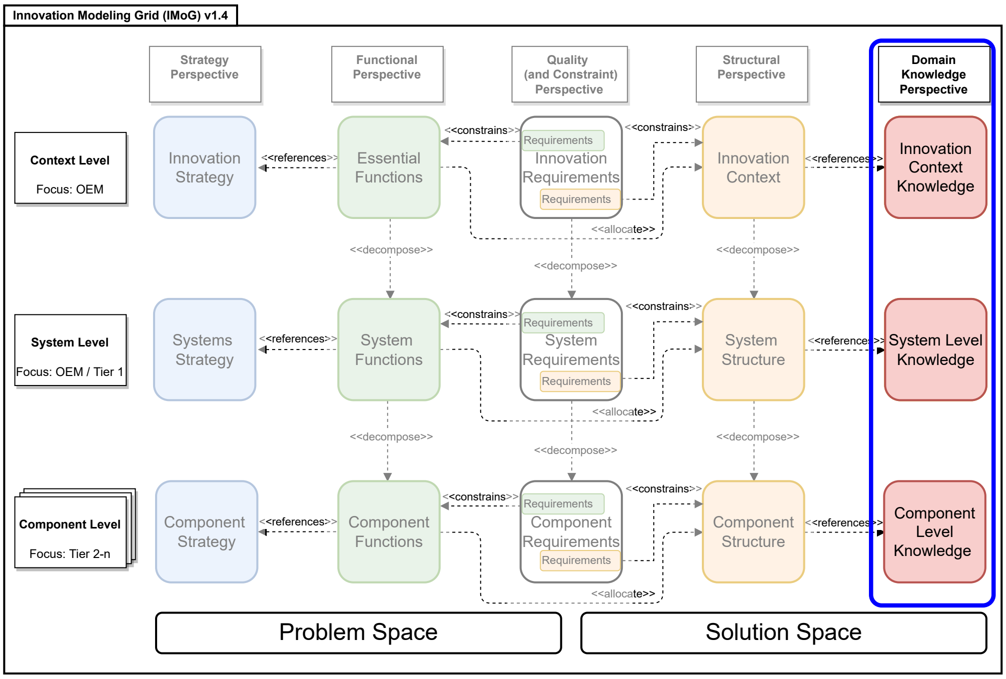


Figure 9.1.: **Location of the Domain Knowledge Perspective in IMoG**

Part III.

Tooling, Evaluation and Closing

10. Tooling Prototype

We created a tooling prototype for IMoG to evaluate our modeling methodology. The scope of the prototype is the Functional Perspective. The prototype was limited to this scope because the effort for consistent tooling between all perspectives within this project was too high. A sophisticated tooling is thus left open for an industrial development after this project. The Functional Perspective is based on the well-known Feature Models [9]. As already mentioned in Chapter 6 (Functional Perspective), we adjusted the meta model to our needs in the context of public committee-based road mapping. This includes the differentiation of "Features" and "Functions" and the addition of a description, an abstraction level and various attributes to each block. The abstraction level is primarily used for filtering purposes. A "configuration" of the Functional Perspective is defined similarly to Feature Models. However, configurations are currently not supported in the prototype.

We considered two approaches in achieving tooling support for the Functional Perspective:

1. Translating or implementing IMoG into an existing modeling language like UML or SysML to take advantage of the existing tools. This approach is faster and eases the integration of IMoG into the internal modeling processes of the industry companies.

2. Or by implementing a dedicated tooling prototype for IMoG by ourselves. This approach takes more effort, but brings a more elegant solution with a better learnability curve and a better user experience.

For the purpose of an IMoG evaluation, we chose the latter approach, which promises a better user experience and less distortion in an evaluation. Figure 10.1 underlines this reasoning by discussing what kind of a benefit we expect from using different kinds of tooling support for IMoG in a committee:

With *no tooling support* used in the committee, the expectation is, that IMoG generates a huge modeling overhead. The models would be drawn and the committee would need guidance on how and what to model. An IMoG expert would be essentially required to

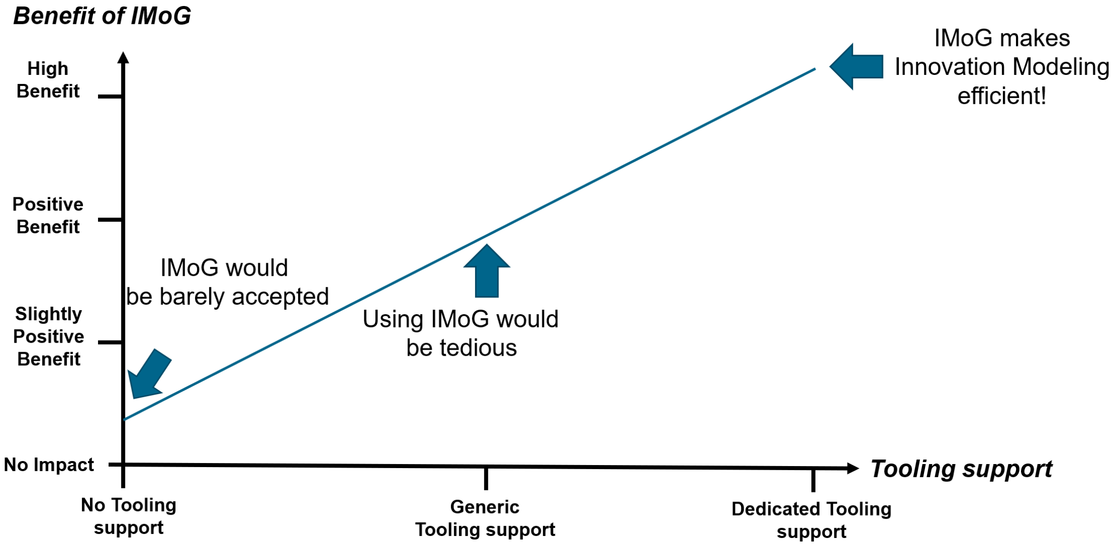


Figure 10.1.: **Comparison between the expected benefit of having no tooling support, generic tooling support with an existing tool and using a dedicated tool for IMoG. The horizontal axis represents the amount of tooling support in the committee, ranging from "No Tooling support" over a "Generic Tooling support" to a "Dedicated Tooling support". The vertical axis represents the expected benefit of IMoG in a committee. In general, the more the tooling is designed around its application, the higher is the expected acceptance!**

create the models. The strong separation of perspectives and abstraction levels would make it hard to remain efficient. Thus, IMoG would be barely accepted in the committee. Only the use of IMoG's process would provide a guidance and would lead to a slightly positive benefit in innovation modeling.

When *IMoG is translated to a generic modeling language* like UML or SysML and supported by a generic tooling, then IMoG would provide a positive benefit to the committee. IMoG would be roughly supported by different types of diagrams for perspective and view separation, but the modeling elements would not perfectly fit and would be cumbersome to handle. It would take the committee some time learning on how to handle the models. Only an expert would be able to set up IMoG in the external tool, such that it would be usable by others. All in all, the tooling would only have a moderate acceptance in the committee.

IMoG makes innovation modeling efficient when *fully supported by a dedicated tooling* *approach*. The dedicated tooling would support templates for perspectives and mapping views between different models. Each modeling element would have a dedicated interface corresponding for its considered use. There would be hardly any learning overhead, and the good user experience would be motivating. The tooling would understand IMoG's process and would request the necessary inputs from the user. A new user would be guided through the tooling and would not have to understand IMoG to the degree of an expert to provide meaningful content. IMoG together with its tooling would significantly reduce the time required for innovation modeling and would help to understand the innovation efficiently. The tooling would have a high acceptance in the committee.

10.1. Functional Perspective Prototype

The tooling prototype is publicly available under `https://genial.uni-ulm.de/imog-dev/` (We would like to thank the University of Ulm for providing their tool *IRIS* [1, 2] that we used as a basis for our IMoG prototype).

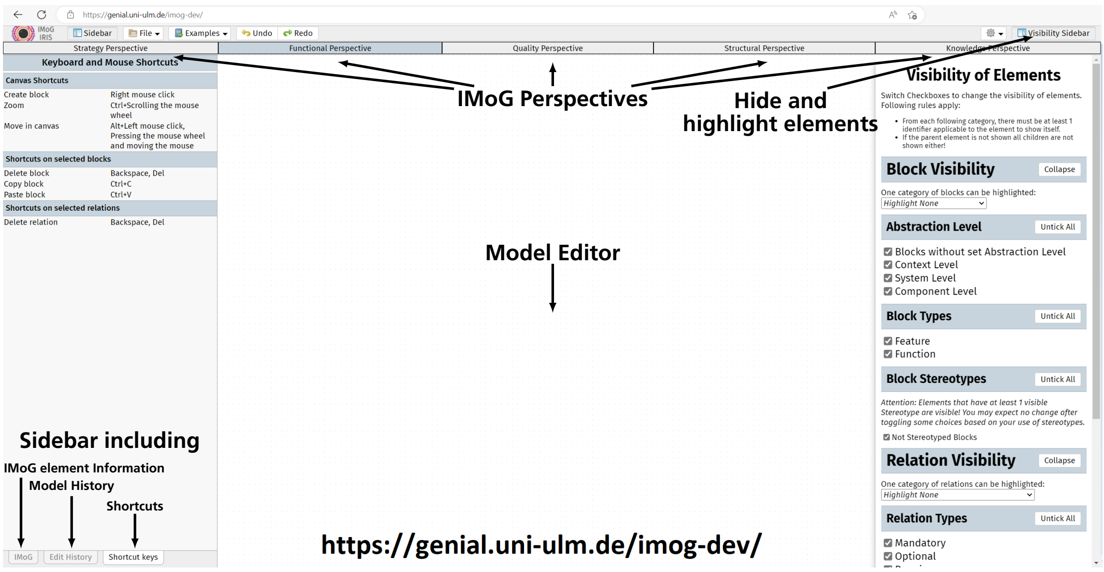
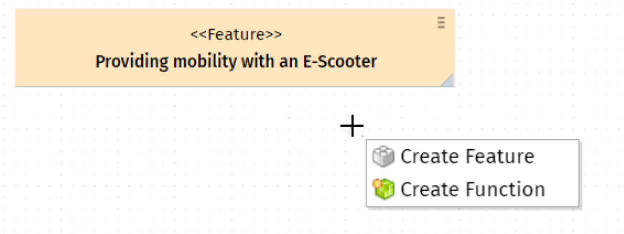
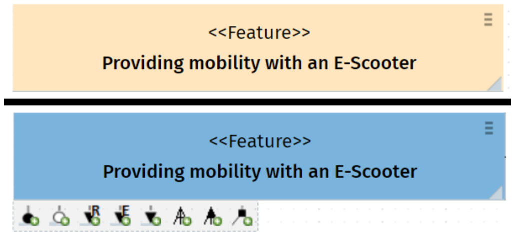


Figure 10.2.: **IMoG tooling - a first glance after opening the prototype in the web browser.**

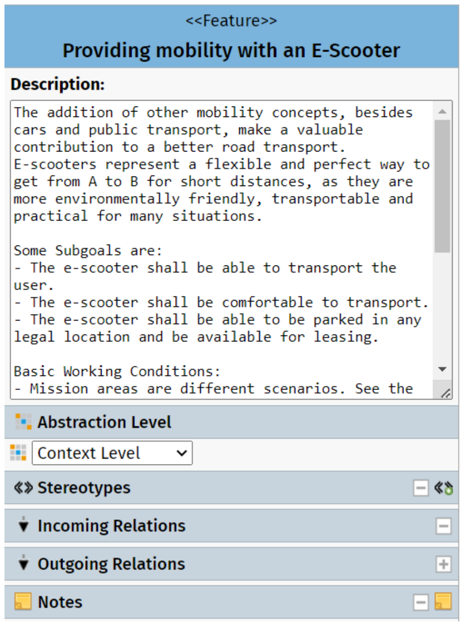
In the following, the model of the Functional Perspective of the e-scooter (see Chapter 6) is created in the prototype to present the features of the tooling. Figure 10.2 shows the first view on the tool, when opened. The first view is shortly described before the model



(a) **Context Menu: Create Features and Functions**



(b) **A Feature in both states: unselected (top) and selected (bottom)**



(c) **The description and information of the selected E-Scooter Feature**

Figure 10.3.: **The context menu, selected Features and the sidebar of the prototype.**

of the e-scooter is created. It contains a model editor in the center, one sidebar on the left and on the right side of the tool, an IMoG Perspectives toolbar and a menu toolbar. The entry "File" in the menu bar can be used for creating new models, saving models and loading models, the entry "Examples" can be used for loading example models like the e-scooter model, the "Undo" and "Redo" buttons and the entry "Settings" can be used for changing the user interface (e.g., the grid size of the model editor). The menu bar additionally includes toggles to open and close the "Sidebar" and the "Visibility Sidebar" on the left and right side of the editor. The Sidebar on the left is used to display and manipulate information about selected model elements, for tracking of changes in the model history (Edit History) and for presenting the Keyboard and Mouse Shortcuts. It is possible to view older model states through the model history. The Visibility Sidebar on the right can be used to filter model elements and highlight them. For example, it is possible to highlight all Function Blocks in a green color while hiding all mandatory relations. The IMoG Perspectives toolbar allows to select the current perspective and view. The Functional Perspective is, for example, selected in the image. It is possible to swap to different perspectives in the prototype, but the model editor is disabled for the other Perspectives. The model editor can be used to create the Functional Perspective model, like the e-scooter model.

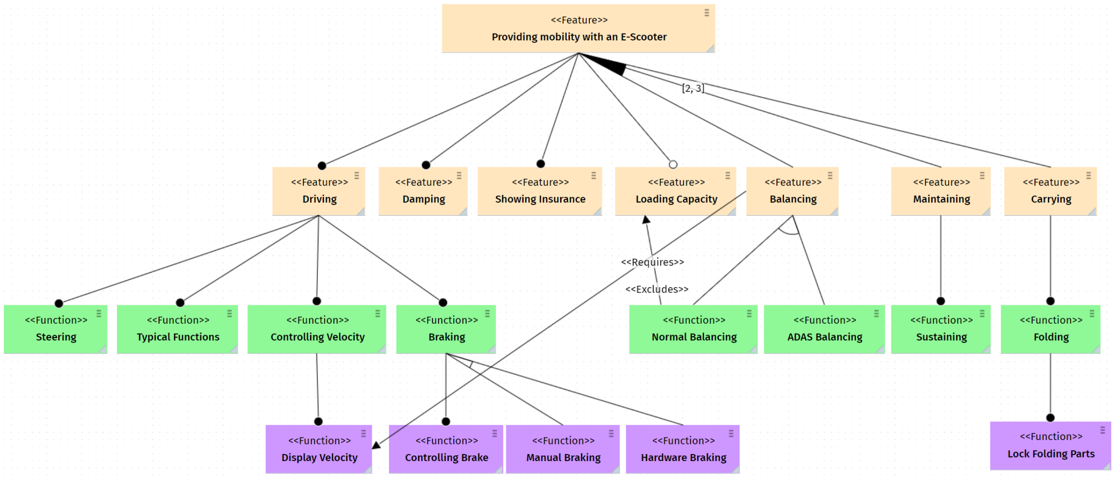


Figure 10.4.: **The model of the e-scooter in the prototype corresponding to the model presented on the Functional Perspective (see Chapter 6). It contains the root feature (yellow block), one layer of context level features (yellow blocks), one layer of system level functions (green blocks) and one level of component level functions (purple blocks) with the corresponding relations (mandatory, optional, constraint, . . .).**

The root feature of the e-scooter model can be created by using the context menu of the Model Editor (see Figure 10.3a) and by entering its name "Providing mobility with an E-Scooter". Each Feature and Function is represented by colored blocks. Each block can be selected to operate with (see Figure 10.3b), like by renaming them, resizing them, changing their type and color or duplicate them. When selected, the tab "IMoG" in the Sidebar will present further information about the block (see Figure 10.3c). This information can be manipulated, giving the blocks a description, changing their stereotype or setting their abstraction level. After adding further Features and Functions to the model, the Features and Functions can be related with each other. A relation can be created by selecting a block, choosing a relation from the relation toolbar under the block and then clicking on the target block. All eight relations of the Functional Perspective can be used, changed and enriched with labels and information. These include the "Mandatory" relation, the "Optional" relation, the "Requires" constraint relation, the "Excludes" constraint relation, the "Custom" relation, the "Alternative" relation, the "Or" relation with a given cardinality and the "Multi directional Custom" relation. Assuming the other Features, Functions and relations of the e-scooter model (see Chapter 6) are created, then the model of the e-scooter shall look similar to the model in Figure 10.4.

10.2. Tooling Evaluation

We have conducted a user experience evaluation with two other researchers. We gave them some predefined tasks to learn handling the user interface like creating some blocks, changing their properties, relating blocks with each other, saving and loading models and using the filtering features. Afterwards, they had to redraw a model given on paper to identify their use of the tooling. Furthermore, we asked them to open an example model and asked some more challenging questions about the model to identify if they understood what the model was meant to present. Finally, we did a round of structural interviews with them regarding their user experience. The prototype was overall evaluated as fast responding and easy-to-use. We experienced the limitation of missing the other perspectives of IMoG in the prototype. Nonetheless, our prototype demonstrated the large potential of a dedicated tooling approach for any IMoG related project by not bothering the user with cumbersome interactions. Thus, we encourage the reader, other industry partners or committees to try out IMoG and its tooling prototype in their committee.

11. Evaluation

The initial evaluation in the original proposal of IMoG [6] stated two strengths: the appropriate level of abstraction for modeling innovations and the examined ways through the matrix. The examined ways include, for example, a top-down diagonal approach from the Context Level of the Strategy Perspective down to the Component Level of the Structural Perspective or a bottom-up approach from the ideas of the semiconductor suppliers back to the context of the car manufacturers.

The appropriate level of abstraction was confirmed and further underlined by the use cases where we have applied IMoG: IMoG helped us to adequately tackle the innovations. We reconsidered our opinion regarding the mentioned ways through the matrix. Instead of specifying several possible ways through IMoG, we think it is rather appropriate to follow the mentioned process for IMoG presented in section **??**. Furthermore, iterating between the problem space and solution space perspectives similar to the process defined in the twin peaks model [14] is in our opinion the most appropriate approach.

The initial evaluation in the original proposal of IMoG [6] identified three potential limitations based on an academic example of wireless charging: scalability, detailed behavioral models, and bridging to product level models. The application of IMoG to the larger example of the e-scooter sheds more light onto these topics. Firstly, we did not encounter any issues regarding scalability in the use cases modeled here, which indicates that IMoG as such does not introduce unnecessary and unmanageable complexities. Secondly, our use case here confirms the view that the absence of detailed behavioral models is actually a strength: Details are not required and should be left out in abstract innovation modeling. Nonetheless, such detailed models should be possible to be attached to solution blocks whenever needed. Finally, the bridge between an IMoG model to a product level model remains properly solvable: Bridging the gap by referring IMoG's elements, using transformations of IMoG models to established system level development languages or by translating the IMoG model into a development focused framework (see Broy et al. in [3]) with adding the behavioral aspects to the designed framework are the recommended choices.

While applying the use cases we learned two more lessons: Reordering the perspectives into the problem space and solution space made it easier to apply IMoG. This distinction got added to the design principles of IMoG (see section 4.1). Another lesson was, that interpreting abstraction levels as filter functionality is better suited for the modeler than interpreting abstraction levels as a division into diagrams. We examined that the division of an innovation model into several pieces would do more harm regarding its user experience and usefulness than it would help.

12. Closing

This technical document presented the Innovation Modeling Grid in detail. This document is the successor of two publications on IMoG [6, 11] and focuses on presenting all details of the methodology. Beginning with the process and an overview, each perspective was presented in detail. Afterwards the tooling and the evaluation was presented.

Overall, we think that IMoG has great potential to be really useful in committee driven innovation modeling. Next to the applications of the e-scooter example and the application in a project of the GAIA-X family [17], IMoG is currently applied in the "Arbeitskreis Automotive" in a workshop series. This document shows that much is already researched about IMoG, however IMoG still has some missing ends in parts of solution definition and tooling. The model of IMoG can still improve and this improvement is enabled through getting some crucial feedback from applications like the workshop series. Therefore, if one is interested in committee driven innovation modeling, we encourage to take a look at IMoG and tailor it to the needs of their committee.

Part IV.

Appendix

This appendix contains the following parts:

- The glossary of IMoG

Glossary

Definitions from the Literature are abbreviated with the following References.

References

[Architecture Modeling] - http://ses.informatik.uni-oldenburg.de/download/bib/paper/OFFIS-TR2011_ArchitectureModeling.pdf
[Autosar] - https://www.autosar.org/fileadmin/user_upload/standards/foundation/19-11/AUTOSAR_TR_Glossary.pdf
[Bosch] - Design & Use of Software Architectures - Adopting and Evolving a Product-Line Approach.
- https://dl.acm.org/doi/book/10.5555/339362
[Cz98] Czarnecki, Krzysztof. Generative programming-principles and techniques of software engineering based on automated configuration and fragment-based component models. Diss. 1998.
[IEEE1471] - http://www.iso-architecture.org/ieee-1471
[IEEE1471] - https://www.iso.org/obp/ui/#iso:std:iso-iec-ieee:42010:ed-1:v1:en
[INCOSE] - INCOSE Systems Engineering Handbook : A Guide for System Life Cycle Processes and Activities, 2015
[ISO15288] - https://www.iso.org/obp/ui/#iso:std:iso-iec-ieee:15288:ed-1:v1:en
[ISO24765] - https://www.iso.org/obp/ui/#iso:std:iso-iec-ieee:24765:ed-2:v1:en
[ISO26262] - https://www.iso.org/obp/ui/#iso:std:iso:26262:-1:ed-2:v1:en
[IREB] - International Requirements Engineering Board IREB - http://www.compliance-technologies.com/DS/ireb_cpre_glossary_17.pdf
[Gupta19] https://medium.com/@nikhilgupta08/problem-space-vs-solution-space-f970d4ace5c
[Kang90] - Feature-Oriented Domain Analysis (FODA) - Feasibility Study - https://apps.dtic.mil/sti/pdfs/ADA235785.pdf
[Knauf] - https://knaufautomotive.com/de/tier-1-und-oem-automobilteile/
[Mozilla] - https://developer.mozilla.org/en-US/docs/Web/HTML/Element/div
[Pohl] - Software Product Line Engineering - https://link.springer.com/book/10.1007/3-540-28901-1
[SPES] - Model-Based Engineering of Embedded Systems - https://www.springer.com/de/book/9783642346132
[SysML] - https://sysml.org/.res/docs/specs/SysML-v1-Glossary-06-03-04.pdf

Abstraction Level:
(1) An abstraction layer defines a specific level of abstraction and granularity at which the System under Development is examined. The level of granularity of the respective abstraction layer is in turn determined by a structural characteristic that stems from the layer above. Initially, we consider the system as a whole. [SPES]

(2) An abstraction layer or abstraction level is a way of hiding the working details of a subsystem by generalizing away certain aspects. [Inspired by Wikipedia-Abstraction Layer]

Aggregation:
See Decomposition and Aggregation

Alternative:
See Variant

Allocation:
An allocation is an assignment on one element onto another. An element may represent a feature, function, requirement, block, property, a resource, hardware, tasks, software, persons or others. In IMoG, a function to block assignment is a common allocation. [IMoG Definition]

Assignee:
Quality Perspective
The responsible person for any given Block (a Feature, a Requirement or a Component). [IMoG Definition]

Block:
A Block is a visual representation of a functionality, requirement or structure element. [IMoG Definition]

Note, that the differences in the meta models are too big to use the definition from somewhere else like SysML (A modular unit that describes the structure of a system or element. [SysML]). IMoG treats Blocks as visualisation elements only.

Cardinality:
The minimum and maximum number of objects in a relationship. [IREB]

Committee Leader:
The responsible person leading the roadmap committee. [IMoG Definition]

Component:
Structural Perspective

A Component is a Block on the Structure Perspective, that is follows the semantics of either (1), (2) or (3) [IMoG Definition]

(1) non-system level element (3.41) that is logically or technically separable and is comprised of more than one hardware part (3.71) or one or more software units (3.159) [ISO26262]

(2) A Component is a unit of composition with cantractually specified component interfaces and explicit context dependencies only; it can be deployed independently and is subject to composition by third parties.[Pohl]

Note, that Components are restricted to Perspectives in IMoG, such that the context information are essential for the Components definition. The typical interpretation that everything is a component as often found in other model definitions just does not apply for IMoG.

Component Level:
The Component Level describes the components of the system and parts in its atoms and is imposed over all perspectives. [IMoG Definition]

Configuration:
A feature configuration is a chosen set of features that represent an innovation where every choice is resolved. A feature configuration is permitted by a feature model if and only if it does not violate constraints imposed by the model. [Wikipedia - Feature Models]

The IMoG Functional Perspective represents a Feature Model.

Constraint:
Quality Perspective

See Requirement (Constraint)

Context Level:
The Context Level describes the innovation as a whole system embedded in its environment and is imposed over all perspectives. [IMoG Definition]

Corporation Representative:
The responsible person of a corporation to coordinate the corporation internal tasks to produce the needed imputs for the roadmap. [IMoG Definition]

Decomposition and Aggregation:
(1) Decomposition denotes the partitioning of an analysis element or design element (e.g., of a goal, a function, or a logical/technical component) into parts. [SPES]
(2) Abstracting a collection of units into a new unit is called aggregation. For example, school is an aggregation of students, teachers, etc. Refining an aggregation into its constituent units is called decomposition. [Kang90]

Domain Expert:
The domain expert represents a specialist of a particular discipline covering subdomains of development. The domain expert supports the innovation modeling and evaluates its influences and dependencies of certain domain elements on other domain elements. [IMoG Definition]

Feature:
Functional Perspective

(1) A Feature is a noticeable or important part of an innovation. The Feature is an abstraction of functionalities that require system of systems solutions. The Feature functionality is described by the User Stories and decomposed into Functions. [IMoG Definition]
(2) A feature is a logical unit of behavior that is specified by a set of functional and quality requirements [Bosch]
(3) A Feature is an end-user visible characteristic of a system.[Pohl]

Function:
Functional Perspective

A task, action or activity that must be accomplished to achieve a desired outcome.
[AUTOSAR 2012]

Functional Perspective:
In the Functional Perspective the functional needs of the system and their breakdown into sub-functions is modeled. The set of all sub-functions then realize the top-level system functionality. [Architecture Modeling]
The Functional Perspective is part of the problem space and describes the features (like User Needs) and functionalities needed for the innovation without focusing on solutions. For details see the description of the Functional Perspective in its associated file. [IMoG Definition]

Future Availability:
The Future Availability represents the date when the Requirement or any other Block will be relevant. [IMoG Definition]

Generalization:
See Specialization and Generalization

HTML Divs:
The HyperText Markup Language (HTML) Content Division element (Div) element is the generic container for flow content. HTML is primarely used for representing the content of Websites. [Mozilla]

Strategy Perspective

IMoG responsible Model Expert:
The responsible person of creating and maintaining the IMoG model on the command of the committee members. The IMoG responsible Model Expert is also called IMoG Modeler. [IMoG Definition]

Knowledge Perspective:
The Knowledge Perspective targets the creation of a database of reuseable blocks of existing models and a knowledge base by transferring the knowledge of domain experts into an appropriate ontology. The Knowledge Perspective is part of the solution space and is used to refine the model with existing knowledge and constraints. The expert knowledge in the knowledge base may for example contain sensor characteristics and constraints from the road traffic regulations. [IMoG Definition]

Mapping (between Views/between abstraction layers):
A mapping between Views is a relationship between two models representing the Views. A mapping can exist between models of different Viewpoints or between models of the same View, but on adjacent abstraction layers. [SPES]

Model:
A model is an abstract representation of an existing reality or a reality to be created. Every model is created for a specific purpose of use. [SPES]

OEM:
The OEM (Original Equipment Manufacturer) is the manufacturer of the final product, which deals with the market launch of the vehicle. [Knauf]

Perspective:
A Perspective combines Views of different abstraction levels which are related to similar Viewpoints. Perspectives can be used to group and to structure Views of different disciplines in order to cope with the complex task of developinga system. [Architecture Modeling]

Problem Space:
(1) The Problem Space focuses on the description of the problem while ignoring technical informations. [IMoG Definition]

(2) The set of all valid system specifications in a domain (e.g. valid feature combinations) is referred to as the problem space and the set of all concrete systems in the domain is referred as to as the solution space. [Cz98]

(3) A lenghty example based explanation can be found under [Gupta19].

Quality Perspective:
The Quality Perspective represents the link between problem space and solution space and contains constraints and requirements to Features and Functions as well as to structural solutions. For more details see the description of the Quality Perspective in its associated file. [IMoG Definition]

Refinement:
A relationship that represents a fuller specification of something that has already been specified at a certain level of detail. [SysML]

Requirement:
(1) A requirement is:

Quality Perspective

1. A need perceived by a stakeholder.
2. A capability or property that a system shall have.
3. A documented representation of a need, capability or property. [IREB]

(2) A statement that identifies a system, product, or process characteristic or constraint, which is unambiguous, clear, unique, consistent, stand-alone (not grouped), and verifiable, and is deemed necessary for stakeholder acceptability. [INCOSE Sys Eng Handbook V4]

Requirement (Constraint):
A requirement that limits the solution space beyond what is necessary for meeting the given functional requirements and quality requirements. [IREB]

Quality Perspective

Requirement (Functional):
A requirement concerning a result of behavior that shall be provided by a function of a system (or of a component or service). [IREB]

Quality Perspective

Requirement (Non-Functional):
A quality requirement or a constraint. Synonym: Extra-functional requirement [IREB]

Quality Perspective

Requirement (Quality, Performance):
A quality requirement is a requirement that pertains to a quality concern that is not covered by functional requirements.

A Performance requirement is describing a performance characteristic (timing, speed, volume, capacity, throughput...). Performance requirements may be regarded as another category of non-functional requirements. In this glossary, performance requirements are considered to be a sub-category of quality requirements. [IREB]

Quality Perspective

Requirements Engineer:
The requirements engineer creates initial top-level requirements for the innovation and captures them uniformly (formally or in natural language). The requirements engineer leverages the expertise of the domain experts and system architects to uniformly refine the requirements in the system models. [IMoG Definition]

Roadmap Manager:
The roadmap manager monitors the innovation status, reports to top management on the feasibility of the innovation, surveys new technologies from other partners, and updates the roadmap. The roadmap manager investigates trends and innovations. During innovation modeling, the roadmap manager performs the initial tasks and writes the roadmap after consulting with the other domain experts, requirements engineers, and system architects. [IMoG Definition]

Roadmap Manager of the Committee:
The roadmap manager of the committee is responsible for the creation and maintenance of the roadmap. [IMoG Definition]

Robustness:
The degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions. [ISO/IEC/IEEE 24765:2017]

Specialization and Generalization:
(1) Abstracting the commonalities among a collection of units into a new conceptual unit suppressing detailed differences is called generalization. Refining a generalized unit into a unit incorporating details is called specialization. For example, the conceptual entity "employee" is an abstraction of secretaries, managers, technical staffs, etc. [Kang90]

Specification:
(1) A set of requirements for a system or other element. [SysML]

(2) A systematically represented description of the properties of an entity (a system, a device, etc.) that satisfies given criteria. It may be about required properties (requirements specification) or implemented properties (e.g., a technical product specification). [IREP]

Solution Space:
(1) The Solution Space focuses on how to translate the problem into solution designs. [IMoG Definition]

(2) The set of all valid system specifications in a domain (e.g. valid feature combinations) is referred to as the problem space and the set of all concrete systems in the domain is referred as to as the solution space. [Cz98]

(3) A lenghty example based explanation can be found under [Gupta19].

SQL Query:
A database query based on the formal language *Structured Query Language* compliant to the standard ISO/IEC 9075.

Stakeholder:
(1) Individuals, groups, and/or institutions which may be impacted by the system throughout its life cycle, including acquisition, development, production, deployment, operations, support, and disposal. [SysML]

(2) Individual or organization having a right, share, claim, or interest in a system or in its possession of characteristics that meet their needs and expectations [ISO/IEC/IEEE 15288:2015]

Stereotype:
A stereotype defines how an existing Block may be extended, and enables the use of platform or domain specific terminology or notation in place of, or in addition to, the ones used for the extended Block. [IMoG Definition inspired by SysML (replaced 'metaclass' with 'Block')]

Strategy Perspective:
The Strategy Perspective is expected to be the first considered Perspective of IMoG (part of the problem space) and targets the first phase of the creation and the capturing of one innovation. For details see the description of the Strategy Perspective in its associated file. [IMoG Definition]

Structural Perspective:
The Structural Perspective targets the modeling of the solution space by using the Requirements and Functions. It may include environment descriptions, system decompositions including software elements, hardware elements and a mapping between them, Mission Profiles and properties as well as sensor descriptions or abstract technologies with parameters, functions and key performance indicators. For more details see the description of the Structural Perspective in its associated file. [IMoG Definition]

System Architect:
The system architect has the role of an interdisciplinary expert who designs systems by using modeling techniques. The system architect has know-how in the area of software - hardware design. In innovation modeling, the system architect takes on the role of the innovation modeler and its decomposition into subsystems. [IMoG Definition]

System Level:
The System Level describes the system and its parts and is imposed over all perspectives. Decompositions and property consistency are of special importance on the System Level. [IMoG Definition]

Tier 1:
The Tier 1 suppliers develop system solutions that are tailored to the end product without major changes. [Knauf]

Tier 2:
The Tier 2 creates the components needed to be integrated into systems. This includes the production of semiconductors and microcontrollers. [inspired by Knauf]

Quality Perspective

User Need:
A prerequisite identified as necessary for a user, or a set of users, to achieve an intended outcome, implied or stated within a specific context of use. [ISO/IEC/IEEE 24765:2017]

Variant (or Alternative):
A variant is a representation of a variability object (a particular instance of a variable item of the real world or a variable property of such an item) within domain artefacts.[Pohl]

For more information about the IMoG Model Element "Variant" see the section "Model Elements".

Variation Point:
A Variation Point is a representation of a variable item of the real world or a variable property of such an item within domain artefact enriched by contextual information. [Pohl]

For more information about the IMoG Model Element "Variation Point" see the section "Model Elements".

View:
A View is a representation of a whole System under Development from the Perspective of a related set of concerns (based on [IEEE1471]). [SPES]

In IMoG the "System under Development" can be substituted with an innovation.

Viewpoint:
A Viewpoint is a specification of the conventions for constructing and using a View. Viewpoints comprise patterns or templates from which to develop individual Views by establishing the purpose and audience for a View and the techniques for its creation and analysis (based on [IEEE1471]). [SPES]

Bibliography

[1] Iris tooling, `Online: https://genial.uni-ulm.de/dev/`

[2] Breckel, A., Pietron, J., Juhnke, K., Sihler, F., Tichy, M.: A domain-specific language for modeling and analyzing solution spaces for technology roadmapping. Journal of Systems and Software **184**, 111094 (2022)

[3] Broy, M., Gleirscher, M., Merenda, S., Wild, D., Kluge, P., Krenzer, W.: Toward a holistic and standardized automotive architecture description. Computer **42**(12), 98–101 (2009). https://doi.org/10.1109/MC.2009.413

[4] CARIAD: Transforming automotive mobility. CARIAD Homepage, `https://cariad.technology/de/en/company.html`

[5] Czarnecki, K., Eisenecker, U.W.: Generative Programming: Methods, Tools and Applications. ACM Press/Addison-Wesley Publishing Co. (2000)

[6] Fakih, M., Klemp, O., Puch, S., Grüttner, K.: A modeling methodology for collaborative evaluation of future automotive innovations. Software and Systems Modeling **20**(5), 1587–1608 (2021). https://doi.org/10.1007/s10270-021-00864-3

[7] Infineon Technologies AG: Innovation in the automotive industry. Infineon Homepage, `https://www.infineon.com/cms/de/about-infineon/press/press-releases/2022/INFATV202201-039.html`

[8] Systems and software engineering — life cycle processes — requirements engineering — Part 1: Vocabulary. Standard, International Organization for Standardization, Geneva, CH (2018), `https://www.iso.org/standard/72089.html`

[9] Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E., Peterson, A.S.: Feature-oriented domain analysis (FODA) feasibility study. Tech. rep., Carnegie-Mellon (1990)

[10] Klemp, O., Puch, S., Westphal, B.: IMoG - a methodology for modeling future micro-electronic innovations

[11] Klemp, O., Westphal, B., Puch, S.: Imog–a methodology for modeling future micro-electronic innovations. arXiv preprint arXiv:2304.09110 (2023)

[12] Knauf Industries: Tier 1 and oem automotive parts. Homepage, `https://knaufautomotive.com/de/tier-1-und-oem-automobilteile/`

[13] Kuhnert, F., Stürmer, C., Koster, A.: Five trends transforming the automotive industry (2018), `https://www.pwc.at/de/publikationen/branchen-und-wirtschaftsstudien/eascy-five-trends-transforming-the-automotive-industry_2018.pdf`

[14] Nuseibeh, B.: Weaving together requirements and architectures. Computer **34**(3), 115–119 (2001). https://doi.org/10.1109/2.910904

[15] Olsen, D.: The lean product playbook: How to innovate with minimum viable products and rapid customer feedback. John Wiley & Sons (2015)

[16] Robert Bosch GmbH: The software-defined vehicle. Bosch Homepage, `https://www.bosch-mobility-solutions.com/en/mobility-topics/software-defined-vehicle/`

[17] Shakeri, A., Klemp, O., Westphal, B.: Shaping a gaia-x data ecosystem through innovation modeling. In: 31st IEEE international Requirements Engineering Conference, RE 2023 (2023)