13th ITS European Congress, Brainport, the Netherlands, 3-6 June 2019

Paper number ITS-TP1873

Creating effective MaaS systems - Using a systems engineering approach to design an open (e)MaaS architecture

S.P. Haveman[1*], J.R. Reyes García[1], E. Felici[2], G.M. Bonnema[1,3]

1. University of Twente, Faculty of Engineering Technology, Department of Design, Production and Management, Drienerlolaan 5, 7522 NB, Enschede, The Netherlands, s.haveman@utwente.nl

2. Ministry of Infrastructure and Water Management, The Hague, The Netherlands

3. University of South-Eastern Norway, Norwegian Industrial Systems Engineering group, Kongsberg, Norway

Abstract

In order to combat various sustainability challenges in mobility, the concept of Mobility as a Service (MaaS) seems a promising direction. To ensure that MaaS can be embedded into our society, an appropriate foundation on which a thriving MaaS ecosystem can develop itself needs to be established. This paper presents the use of a Systems Engineering approach to design a MaaS system architecture that can provide this foundation. We elaborate on two example cases to demonstrate and evaluate this approach. The first case comprises an overall architectural design for a MaaS system focusing on a functional decomposition, whereas the second case provides a functional design of an Application Programing Interface (API) for communication between MaaS Provider and Transport Operator. In the end, it is concluded that a systems approach can be beneficial to establish an effective MaaS ecosystem.

Keywords:

Mobility as a Service, MaaS API, Systems Engineering

1. Introduction

The concept of Mobility as a Service (MaaS) is developing itself rapidly [1, 2] and has been deployed with various business models and approaches [3]. However, in general it can be said that efforts are fragmented as no overall framework that truly facilitates an open and flexible ecosystem exists yet.

This is not surprising since MaaS is a fledgling concept in a complex market. Also, one of the causes might be that one of the leading definitions of MaaS focuses heavily on the user perspective and thus the front-end of the system is mainly considered. For example, MaaS is defined in [4] as “single application to provide access to mobility, with a single payment channel instead of multiple ticketing and payment operations”. This leads to a common abstraction that “MaaS is an app”, while the concept of MaaS entails much more, as also noted in [4].

Fragmentation in the service offer is a real risk for all stakeholders. This can for example lead to oversupply. Examples are for example to be found in bike sharing where bikes were littering streets and ultimately ended up unused, as seen in [5]. On the other hand, if there are only a few MaaS Providers or Transport Operators active, the negative effects of market monopolies could play a role. For example,

not opening up Public Transport towards every MaaS Provider is a risk here.

The benefits of an open ecosystem have not gone unnoticed. On an international level the MaaS Alliance has been established and addresses issues such as a single market and technology issues [6]. Simultaneously, our research is situated in the European eMaaS (electric Mobility as a Service) project [7], which aims to provide an open architecture for (e)MaaS [8]. On a national level, an example of this from the Netherlands is the MaaS pilots that are being deployed in seven regions [9]. Next to simply kick-starting MaaS, one of the main goals is to pave the way for easy nationwide deployment by creating a transparent design and ways to share data. Therefore, working groups for the development of an API (Application Program Interface) have been established, led by the government.

1.1 Problem Definition

To realize a successful (e)MaaS ecosystem in which parties can freely collaborate, we argue in this paper that a systems perspective must be taken into account. This means to take a problem at hand, regard this problem in its environment and start reasoning from that perspective [10]. For a MaaS system, this requires understanding and a description of the (e)MaaS ecosystem (for a first view see [8]). A common design approach for the development of systems is Systems Engineering [11, 12]. Systems are defined in [11] as "an integrated set of elements, subsystems, or assemblies that accomplish a defined objective. These elements include products (hardware, software, firmware), processes, people, information, techniques, facilities, services, and other support elements". Therefore, a systems approach must support a multitude of ways to reason about, visualize and design systems, as for example described in [10] and more practically in [13].

Therefore, the question that we aim to answer is: in what manner can a systems engineering approach contribute to the development of an effective and resilient MaaS ecosystem? With effective we consider a system that adequately supports stakeholders to reach their goals [14]. Resilience signifies a flexible and future-proof system design that can accommodate various business models and modes of operation.

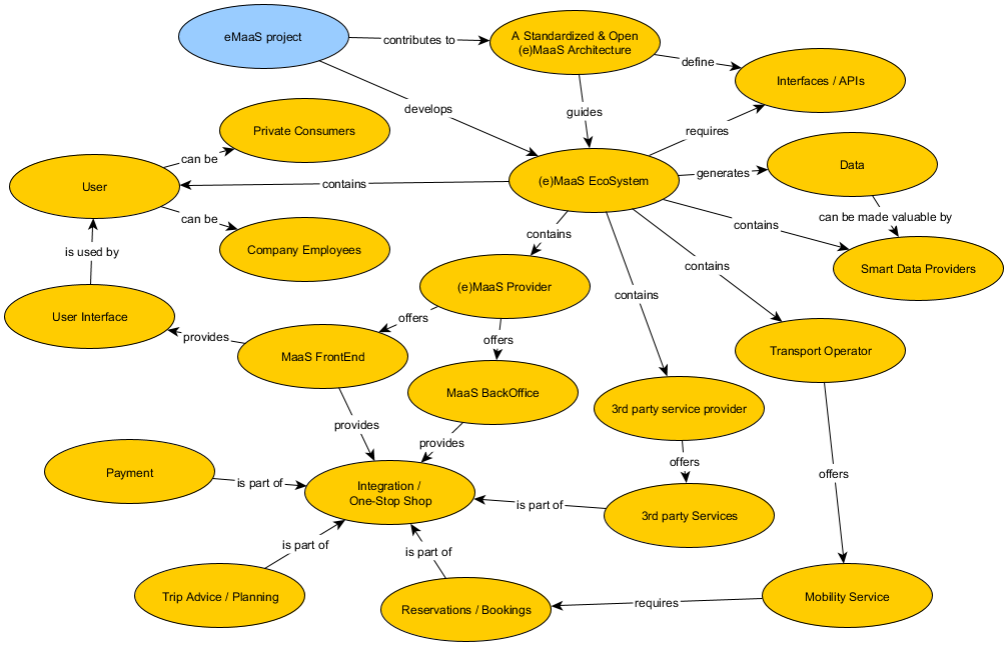

Figure 1: Context Diagram of the eMaaS project [7] and the (e)MaaS ecosystem

1.2 Background of the research

This research is situated in the eMaaS project. The eMaaS project aims to develop a standardized & open (e)MaaS architecture that guides the (e)MaaS ecosystem and supports eco-friendly mobility. A context diagram with the goals and concepts in this project and the envisioned (e)MaaS ecosystem is shown in figure 1. Due to our interest in providing an open architecture, the efforts of a currently established MaaS Provider – Transport Operator API working group led by the Dutch Ministry of Infrastructure and Water Management are also represented in this work.

1.3 Contents

This paper is structured as follows. In section 2, a background into systems engineering and system design approaches is provided and directly related to MaaS. Section 3 & 4 present the outcomes of these design efforts and provide two example cases. Section 5 provides a discussion of the results, as well as a reflection on the applied tools and techniques. Section 6 presents an outlook, in which future research directions are posed.

2. Applying a Systems Approach to MaaS

This section clarifies the term 'a systems approach' and discusses how to apply this to a MaaS context.

2.1 Systems Engineering and Systems Thinking

Systems Engineering (SE) is, according to [11], "… an interdisciplinary approach and means to enable the realization of successful systems…" and a process, perspective and profession [11]. The system architecture is then "the fundamental concepts or properties of a system in its environment…" [15]. In [12], SE is defined as top-down and life-cycle oriented. This means that systems should be regarded in their context and ultimately systems (when engineered) will always be created to achieve a specific objective within that context. To achieve these objectives, a system needs to be able to perform a certain set of functions [16]. These functions can be decomposed and assigned in a structure [17].

The relation between systems engineering and systems thinking is given in [11] as follows: "the SE perspective is based on systems thinking … a perspective that sharpens our awareness of wholes and how the parts within those wholes interrelate". [10] also state that "A person who thinks like a systems engineer can produce excellent results following a variety of different paths". This means that any SE approach should also support switching between viewpoints as a means to enable systems thinking.

At the core of MaaS is the notion that MaaS will deliver an "integrated solution" of different functions. Multiple developments are ongoing by different actors but somehow it needs to grow into a cohesive ecosystem. While this process is not governed through one single entity, separate actors in the MaaS ecosystem can ensure that the value of their efforts is increased by taking into account the overall system perspective. Systems Engineering and System Thinking can support here very well.

2.2 A systems perspective for MaaS Systems

To develop an architecture for MaaS ecosystems, a systems approach could follow the steps below:

1. Clarify the context of the MaaS (eco)system and identify relevant actors and their objectives
2. Make an inventory of the functions that are needed to achieve these objectives
3. Allocate these functions to a certain structure, resulting in a basis for a system architecture

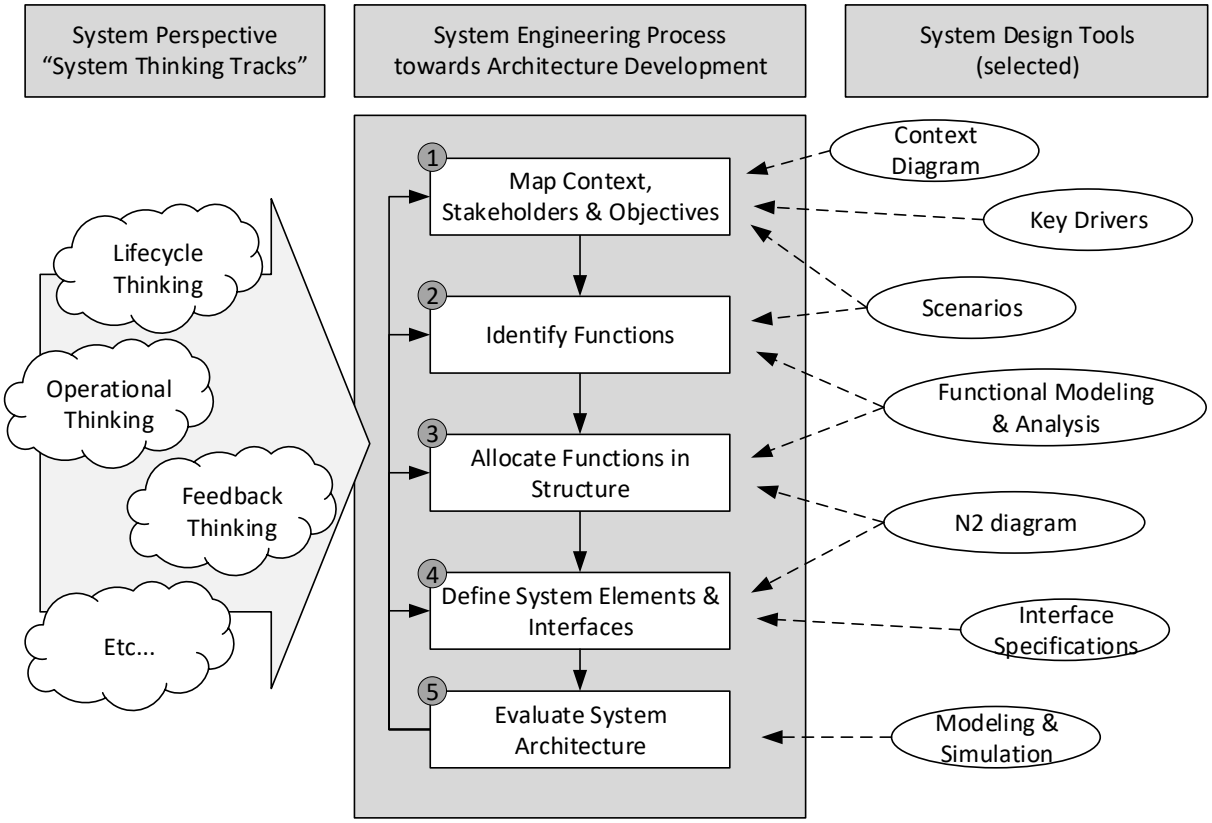

Figure 2: Overview of the systems engineering process, system thinking approach and tools as it can be applied to support development of a MaaS System Architecture

4. Define the elements in this system structure as well as the interfaces between them
5. Evaluate the system architecture to make trade-offs and decisions towards an optimal design

These steps are also shown in Figure 2. Note that this is an iterative process. Figure 2 also visualizes the systems thinking perspective and system design tools and techniques, based on examples given in [13] for the thinking tracks, or in [11–13, 17] for the tools. Examples of applications of system design tools to MaaS systems are:

- Context Diagram; an example of a Context Diagram is already given in Figure 1. A context diagram should outline the context of a system including possibly involved stakeholders. For a MaaS system this could for example include a city of operation. Within this city, the city government could have certain goals and objectives for the overall mobility in a city, which in turn has its effect on the MaaS system. Creation of a context diagram helps to unveil, visualize and act upon these outside influences.
- Scenarios; in order to establish a good reference of how a system will operate, scenarios are a very good way to describe a system in operation, without yet needing to detail the actual systems. These scenarios can provide an excellent basis to derive functions, or reason about the system with various stakeholders. In the case of MaaS, applicable scenarios would be how a user would plan and book a trip, execute a trip or pay for a trip, or how a complete MaaS concept would operate in a city or region and how traffic can be optimized.

- Functional Modelling and Analysis; once (the first) functions are identified, these can be structured and further analysed. Many techniques exist such as Functional Block Diagrams or FAST (Functional Analysis System Technique). An example of a functional block diagram is shown in the first example case, in Figure 3. A FAST diagram technique helps to link higher order functions to lower order functions. For example, a high level function as "improve urban air quality" to a low-level function as "provide sustainability score of proposed itinerary".
- Modelling and simulation, during the development of MaaS systems it can be beneficial to understand the behaviour of the system in more detail. An example for MaaS can be the required number of available transport assets in a certain area of operation, based on an expected user base and their behaviour. Creating models and simulations of these systemic problems allows for better understanding of the effects of the proposed architecture.

The system thinking tracks described in [13] act as a checklist, creativity starter and a means to view the system under design from different perspectives. Examples applied to MaaS are:

- Lifecycle thinking: in the development of MaaS, not only the system in place needs to be considered, but also how to get the system in place and finally how to decommission the system. The MaaS ecosystem in Paris suffered a blow, as after decommissioning the Autolib shared car system, the electric charging infrastructure could not be reused [18]. Lifecycle thinking is also applied in Figure 3, with functions related to establishing the MaaS system.
- Feedback thinking: feedback thinking can be used to consider in advance how to improve the MaaS system over time by ensuring that appropriate data from the system is available to analyse the system. Therefore, attention should be paid to ensuring that the right statistics are available for the appropriate actors in the MaaS system, such as utilization rates, usage patterns, user preferences or planning and booking success rates.

Whereas the previous two lists show several short examples and considerations, the following sections provide two example cases that provide applications of system design tools and system thinking tracks in more depth, to illustrate the systems approach more clearly.

3. Example Case 1: (e)Maas Architecture Design

This section provides an overview of several ongoing design efforts in the eMaaS project towards the development of an architecture for (e)MaaS systems. This section mainly describes step 1 to 3 of the process shown in Figure 2.

To be able to define a first version of the architecture, and particularly to establish a functional overview, various activities were executed. These included an analysis of the relevant systems in place of project partners (various types of mobility & smart data services) through the creation of context diagrams and determination of key drivers [19] for relevant stakeholders, a literature review towards existing architectures and ecosystems [8] and scenario building towards the desired (e)MaaS system.

3.1 Using an N^2 diagram to reason about (e)MaaS Architecture

To establish a functional overview and structure of the system, an iterative approach was used where functions and structure were explored together. A first example is given in table 1 in the form of a N^2

Table 1: N^2 diagram of high level functional concepts in an (e)MaaS System. Functions and the context are shown on the diagonal, whereas the other cells are interfaces between two functions.

User	User Inputs			
System Feedback	**Provide Front-End**	Service Request		
	Data Provision	**Provide Services**	Data Request	
		Processed Data Streams	**Provide Interface Layer**	Data Request
			Data Stream	**Environment**

diagram. A N^2 diagram is a helpful approach to explore the structure of a system. The diagram is a matrix in which the diagonal describes either the systems' functions or elements from its context. If an item is in the same row as a function, these are outputs of the function in that row, whereas the items on the same column are inputs to a function. For example, the function 'provide front-end' has as output a 'service request', which is input for the function 'provide services'. These inputs and outputs describe the interfaces. The N^2 diagram can be expanded with additional (sub)functions and subsequently additional interfaces can be identified. Next, a suitable system structure can be derived.

3.2 Functional View of the (e)MaaS Architecture

Through further iterations on the N^2 diagram (not reported due to space constraints), developed (e)MaaS scenario's as well as the context diagrams and objectives of relevant stakeholders, a comprehensive list of required functions has been identified. Figure 3 shows a functional view which includes several of the identified functions for an (e)MaaS system. The functions are grouped into two main functions, of which the first is to establish & develop the (e)MaaS solution, whereas the second is to operate the (e)MaaS solution. These top-level functions are further specified two levels deeper.

The selection is based on relevancy for the (e)MaaS project partners. A functional overview for

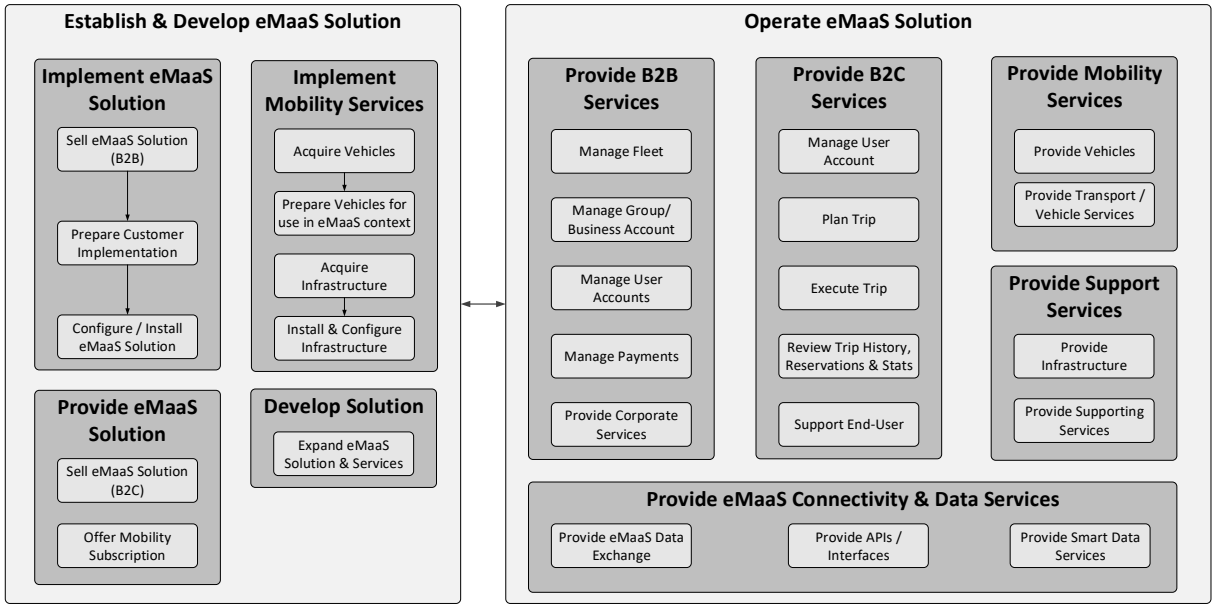

Figure 3: Functional Overview for an (e)MaaS Architecture

(e)MaaS reasoning from the perspective of a Public Transport Operator would look different, though it still would be a view of the same system. To clarify, in principle there is never one single (functional) view of a system that is understandable by, and of value to, all of its stakeholders [20].

3.3 Software Architecture View of the (e)MaaS Architecture

The functional overview and the further iterated N^2 diagram supported the definition of a structure where functions are grouped and interfaces are identified. This grouping of functions leads to the identification of interfaces, and is visualized in a Software Architecture view in Figure 4. Altogether, this view provides a blueprint for the required elements from a software perspective. It could be that all of these elements are put into place by separate actors, or by one single actor. This is not defined at this point. In fact, a definition of which actors have which responsibilities would go against the notion of establishing an open (e)MaaS market where actors can freely join to provide one or more of the requested roles.

4. Example Case 2: Transport Operator – Maas Provider API

This section details the results of applying specific system engineering techniques to a critical component in the eMaaS software architecture, the connection between MaaS Provider and Transport Operator (the "common shared mobility API" in Figure 4). This work has been carried out in the

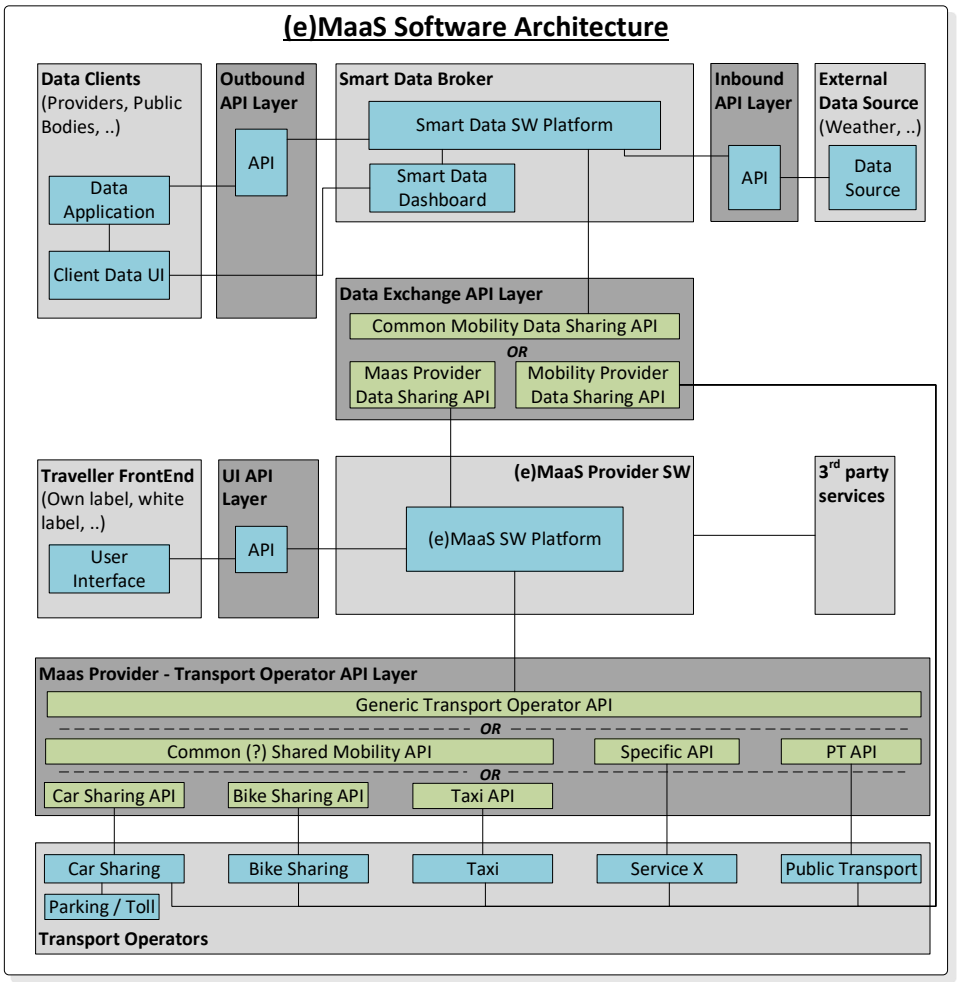

Figure 4: (e)MaaS Software Architecture – Light grey blocks are roles/functions, whereas dark grey blocks are interfaces. For each interface, an API is foreseen to support an open & modular ecosystem

context of a Dutch API working group initiated by the ministry of Infrastructure and Water Management and relates back to step 4 in Figure 2.

4.1 SE approach for an interface

In order to be able to develop complex systems, the system must be decomposed in smaller elements that each perform separate functions. To connect these elements back into one whole again, interfaces are required as well. An API definition is a common way to prescribe a software interface and ensures that code is properly modularized [21]. The main design question is: which functionalities need to be supported in this Transport Operator – Maas Provider API (TO-MP API)? In order to answer this, it must be considered which functions the MaaS Provider and Transport Operator provide themselves, and whether data or knowledge from the other party is required in order to execute this function. To reason about this, the operational concept of the system must be considered.

4.2 Existing MaaS APIs

In the public domain, few examples of MaaS APIs exist. The MaaS Alliance has made available the beginnings of an API [22], currently mainly focused on the booking process between MaaS Provider and Transport Operator. The API description includes an operational view which provides a very useful view on the operational process. However, it also currently lacks a proper functional decomposition in different modules, for example integrating payments in the booking process.

4.3 Functional Design of the TO-MP API

As a first step, the requirements of the most relevant actors, being user, MaaS Provider, and Transport Operator were described in the form of user stories. This overview was cross-referenced with the functional analysis done on the full MaaS system, as described in the first case. Together, this has led to the definition of several main functional and information blocks of the TO-MP API, shown in Figure 5. The TO-MP API functional blocks defined are the following:

- Operator Information: Gives static information on the operator according to the GBFS(+) [23] standard (which is in turn based on the GBFS standard [24])
- Registration: Allows various registration processes of users and sharing of information
- Planning: Gives information about availability, estimated travel time and costs.

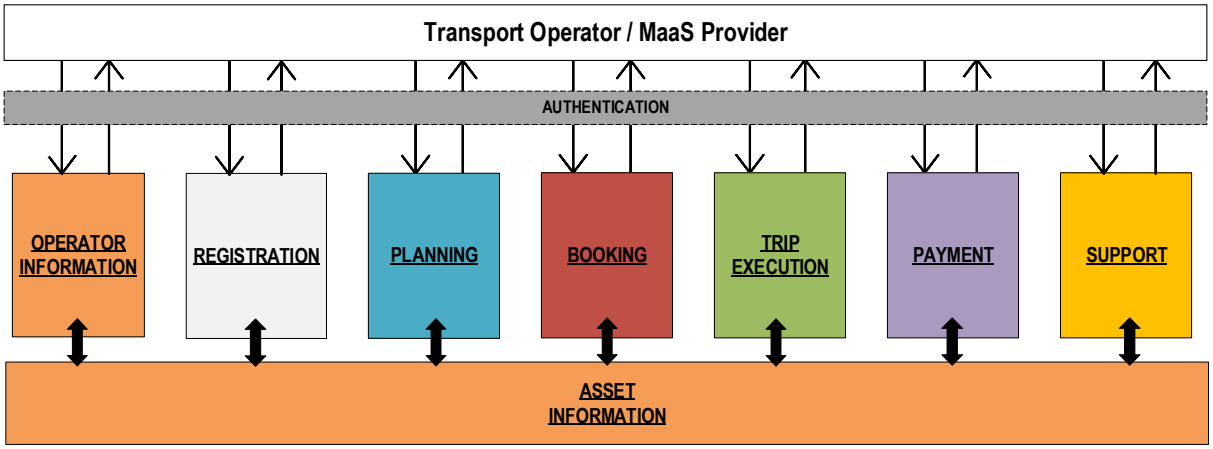

Figure 5: Transport Operator to/from Maas Provider API –Functional Modules.

Note: this is a snapshot of current, ongoing developments in the Dutch TO-MP API working group.

- Booking: Allows booking of a specific asset or service for a specific place, time and date.
- Trip Execution: Allows access to asset and travel during booked period.
- Payment: Allows payment of the service. Supports different business models (i.e. pay-as-you-go or subscription-based).
- Support: Allows a structured process to resolve issues and support users where needed.
- Asset Information: Provides static information used by other modules to supplement API calls with asset information if needed. Assets can be vehicles or infrastructure (e.g. charge points).

The dynamic functional blocks have additional optional modules (not pictured in Figure 5) that might be desired or required depending on scope of the MaaS implementation and Business Models in place.

4.4 Operational Design of the MP-TO API

Next to the functional design, the operational perspective was also considered to build a blueprint for the API design. After all, it must be clear when and where which communication is needed. Clarifying this will also clarify which API calls are required to be included in the final API design. Figure 6 shows an operational view of the API modules 'booking'. Similar views have been developed for the other functional blocks but these were not included due to space constraints. The operational flows are separated in a technical process flow and a user journey. The optional modules are included in the operational flows, except several support procedures as they are hard to locate properly as e.g. a user support question can originate everywhere in an operational flow. This has been incorporated in the design by adding a separate functional block for 'support'.

5. Discussion

This section will address whether a systems engineering approach can contribute to an effective and resilient MaaS ecosystem. In this paper we presented several tools and techniques that are part of a system engineers toolbox. In some cases, these are 'common' techniques that are also in use by other disciplines. However, here they are presented together with the system thinking mindset that allows reasoning about the system in various perspectives. This allows to think beyond the initial application of a MaaS endeavour and consider its place in the overall MaaS ecosystem.

With respect to resilience, the example given with respect to Autolib [18] and its charging infrastructure could have been avoided with a lifecycle thinking perspective. Another example towards resilience is how to support further rollout and expansion of developing MaaS ecosystems. In the Dutch MaaS pilot set-up [9], the ministry of Infrastructure and Water Management foresaw risks of fragmentation and has stimulated the market, under its guidance, to develop interface specifications to ensure modularity and thus flexibility as a basis for further expansion. It remains to be seen how this will develop in practice, but without this effort it would be hard for actors in MaaS systems to build upon each other's services.

With respect to effective MaaS systems, it is our view that systems can only be effective if the objectives of all its stakeholders are considered and addressed. In this work we have presented multiple techniques that can be used to ensure that all of these perspectives are considered. For example, in the presented software architecture in figure 4, the objectives of stakeholders such as governing bodies have been considered, by ensuring that the architecture enables the provision of relevant data to understand and

possibly manage mobility issues through smart city dashboards.

Concluding, we believe that a systems approach is of core importance for the development of MaaS and hope to have demonstrated its potential value through the examples presented in this work.

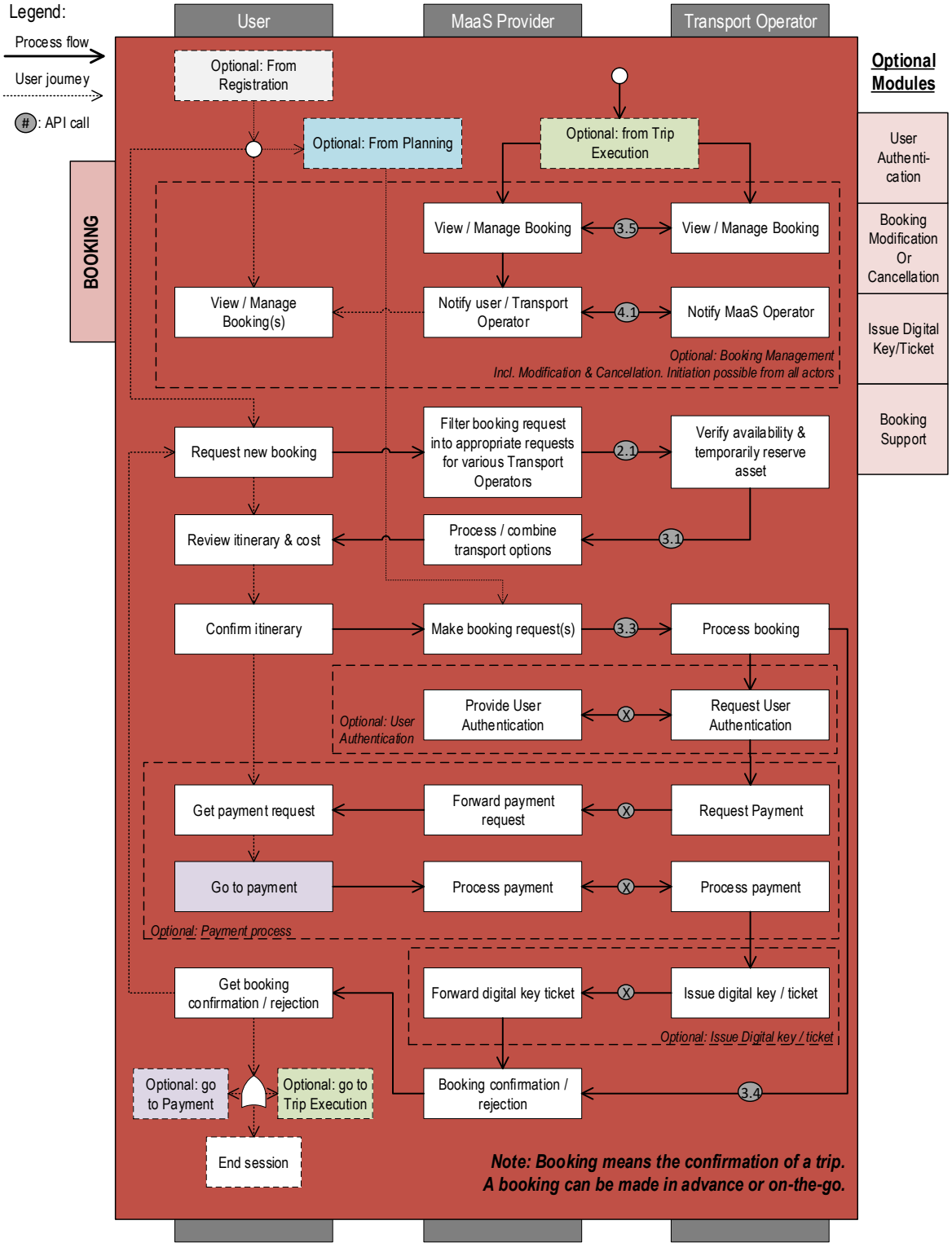

Figure 6: Operational view of the booking module.

Note: this is a snapshot of current, ongoing developments in the Dutch TO-MP API working group

6. Outlook

Within the eMaaS project, we will continue to address the concept of MaaS from a systems perspective and to explore other SE-based approaches such as Systems of Systems [25]. We will also continue to contribute to the development of an (e)MaaS architecture [8] and evaluate in further depth the application of a Systems Engineering approach. Ultimately, the development of this architecture should also support a development roadmap towards an open MaaS ecosystem. Standardization of interfaces through functional designs and API descriptions will be subject of further research, both in the eMaaS project and in various working groups on API standardization for MaaS.

Acknowledgements

The eMaaS project has received funding from the ERA NET COFUND Electric Mobility Europe (EMEurope). Participating project partners are Move About GmbH from Austria, Urban Software Institute GmbH from Germany, the Urban Institute Zrt. from Hungary, Move About AB from Sweden, GoodMoovs (trademark of eMobilityToolBox B.V.) and the University of Twente from the Netherlands.

We would also like to thank collaborators in the Transport Operator – Maas Provider working group led by the Dutch Ministry of Infrastructure and Water Management for their input and feedback on various versions of the API design of which the results presented in this work have benefited heavily.

References

1. Peng, B. (2017). *Car sharing business - Development trends, business model and solutions*.
2. Freese, C., & Schönberg, A. T. (2014). *Shared Mobility - How new businesses are rewriting the rules of the private transportation game*. *Think Act*. Munich, Germany.
3. Jittrapirom, P., Caiati, V., Feneri, A.-M., Ebrahimigharehbaghi, S., González, M. J. A., & Narayan, J. (2017). Mobility as a Service: A Critical Review of Definitions, Assessments of Schemes, and Key Challenges. *Urban Planning*, *2*(2), 13. doi:10.17645/up.v2i2.931
4. Maas Alliance. (2019). What is MaaS? Retrieved January 8, 2019, from https://maas-alliance.eu/homepage/what-is-maas/
5. Taylor, A. (2018). The Bike-Share Oversupply in China: Huge Piles of Abandoned and Broken Bicycles. Retrieved January 11, 2019, from https://www.theatlantic.com/photo/2018/03/bike-share-oversupply-in-china-huge-piles-of-abandoned-and-broken-bicycles/556268/
6. MaaS Alliance. (2017). *Guidelines & Recommendations to create the foundations for a thriving MaaS Ecosystem*. Retrieved from www.maas-alliance.eu
7. electric Mobility as a Service (eMaaS) project. (2018). Retrieved November 15, 2018, from http://www.emaas.eu/
8. Reyes García, J. R., Lenz, G., Haveman, S. P., & Bonnema, G. M. (2019). State of the art of electric Mobility as a Service (eMaaS): ecosystems and system architectures (forthcoming). In *EVS32 Symposium*.
9. Dutch Ministry of Infrastructure and Water Management. (2018). Plan, book and pay for a trip

with one app (in Dutch). Retrieved January 8, 2019, from https://www.rijksoverheid.nl/actueel/nieuws/2018/06/26/met-een-app-een-reis-plannen-boeken-en-betalen

10. Boardman, J., & Sauser, B. (2008). *Systems Thinking: Coping with 21st Century Problems*. (A. B. Badiru, Ed.)*Industrial Innovation Series*. Boca Raton, FL, USA: CRC Press.
11. INCOSE - International Council on Systems Engineering. (2015). *Systems engineering handbook : a guide for system life cycle processes and activities*. (D. D. Walden, G. J. Roedler, K. J. Forsberg, R. D. Hamelin, & T. M. Shortell, Eds.) (4th ed.).
12. Blanchard, B. S., & Fabrycky, W. J. (2011). *Systems Engineering and Analysis*. (Prentice Hall, Ed.)*International Series in Industrial & Systems Engineering* (5th ed.). Upper Saddle River, NJ, USA: Pearson Education International.
13. Bonnema, G. M., Veenvliet, K. T., & Broenink, J. F. (2015). *Systems Design and Engineering: Facilitating Multidisciplinary Development Projects*. Enschede, The Netherlands: CRC Press.
14. Kamargianni, M., & Matyas, M. (2017). *The Business Ecosystem of Mobility-as-a-Service*. *Transportation Research Board (TRB) Annual Meeting*. Washington DC.
15. ISO/IEC/IEEE. (2011). ISO/IEC/IEEE 42010 - Systems and software engineering: Architecture description . New York, NY, USA: Institute of Electrical and Electronics Engineers.
16. Maier, M. W., & Rechtin, E. (2000). *The Art of Systems Architecting* (2nd ed.). CRC Press.
17. Pahl, G., Beitz, W., Feldhusen, J., Grote, K.-H., Wallace, K., & Blessing, L. (2007). *Engineering Design - A Systematic Approach* (3rd ed.). Springer.
18. Lemaire, B. (2018). Autolib terminals can not be used due to lack of software (in French). Retrieved January 8, 2019, from https://www.lemondeinformatique.fr/actualites/lire-les-bornes-autolib-inutilisables-faute-de-logiciel-72700.html
19. Heemels, W., & Somers, L. (2006). The key driver method. *Model-Based Design*, 27–42. Retrieved from http://alexandria.tue.nl/openaccess/Metis226428.pdf
20. Rozanski, N., & Woods, E. (2012). *Software Systems Architecture* (2nd ed.). Upper Saddle River, NJ, USA: Addison Wesley.
21. Bloch, J. (2006). How to design a good API and why it matters. In *OOPSLA '06 - Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications* (pp. 506–507). doi:10.1145/1176617.1176622
22. Maas Alliance. (2019). MaaS Alliance - MaaS API Github Repository. Retrieved January 8, 2019, from https://github.com/maasglobal/maas-tsp-api
23. OpenBikeShare. (2019). General Bikeshare Feed Specification Plus. Retrieved January 8, 2019, from https://github.com/openbikeshare/gbfsplus
24. NABSA. (2019). GBFS - General Bikeshare Feed Specification. Retrieved January 8, 2019, from https://github.com/NABSA/gbfs
25. Boardman, J., & Sauser, B. (2006). System of Systems – the meaning of. In *Proceedings of the 2006 IEEE/SMC International Conference on System of Systems Engineering* (pp. 118–123).