FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Transformation of Service Blueprints into the Business Process Model and Notation

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplomingenieur

im Rahmen des Studiums

Business Informatics

eingereicht von

Matthias Winkelhofer, BSc
Matrikelnummer 00951553

an der Fakultät für Informatik
der Technischen Universität Wien
Betreuung: Ao.Univ.Prof. Dipl.-Inf. Dr.-Ing. Jürgen Dorn

Wien, 10. Juni 2018

Matthias Winkelhofer	Jürgen Dorn

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Transformation of Service Blueprints into the Business Process Model and Notation

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplomingenieur

in

Business Informatics

by

Matthias Winkelhofer, BSc
Registration Number 00951553

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Inf. Dr.-Ing. Jürgen Dorn

Vienna, 10th June, 2018

Matthias Winkelhofer	Jürgen Dorn

Erklärung zur Verfassung der Arbeit

Matthias Winkelhofer, BSc
Arnsteingasse 17/17 1150 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 10. Juni 2018

Matthias Winkelhofer

Danksagung

Ich möchte diesen Abschnitt nutzen um all jenen Menschen zu danken die mich auf dem langen Weg zur Fertigstellung meiner Diplomarbeit begleitet und unterstützt haben.

Mein Dank gebührt Herrn Ao.Univ.Prof. Dipl.-Inf. Dr.-Ing. Jürgen Dorn der mir stets mit Rat zur Seite gestanden ist und dessen wertvoller Input maßgeblich zum Ergebnis dieser Arbeit beigetragen hat.

Auch möchte ich mich ganz herzlich bei den fachlichen Ansprechpartnerinnen und Ansprechpartnern für meine Case Study bedanken die es mir ermöglicht haben auch einen praktischen Einblick in das Thema Service Design zu erhalten.

Weiters bedanke ich mich bei meinen Eltern Margaretha und Gerhard sowie meinem Bruder Bernhard die mich mit unermüdlichen Einsatz und Verständnis während meiner Ausbildung unterstützt und gefördert haben.

Zu guter letzt gehört ein besonderer Dank meiner Partnerin Petra die für mich vor allem emotional eine unbezahlbare Stütze war. Ohne dich hätte ich es nicht geschafft.

Kurzfassung

Die vorliegende Diplomarbeit zählt zum Bereich Service Design und Modellierung. Bedingt durch die Relevanz und Bedeutung solcher immaterieller Leistungen für die aktuelle Geschäftswelt und generell unser tägliches Leben, erfordert deren Entwicklung, Analyse, Dokumentation und Kommunikation eine strukturierte Herangehensweise. Service Blueprinting ist in diesem Zusammenhang ein grafisches Konzept welches den Standpunkt des Kunden als zentrales Element betrachtet. Der eher abstrakte Charakter der Methodik und die generellen Anforderungen des modernen Prozessmanagements können jedoch erfordern, dass weitere, detailliertere Modellierungssprachen angewendet werden. Als Antwort auf diese Gegebenheit befasst sich diese Diplomarbeit nun mit der Kombination von Service Blueprinting und der Business Process Model and Notation (BPMN). Die zugrunde liegende Idee besteht darin eine Repräsentation eines Services in Form eines Blueprints systematisch einzulesen und automatisch ein entsprechendes BPMN-Modell zu generieren. Nach der Behandlung der theoretischen Grundlagen bezüglich der involvierten Elemente und Konzepte wurde eine Case Study zum Thema der *emergency admission of a patient at a surgical ward within an Austrian hospital* entworfen und modelliert. Die Entwicklung des theoretischen Transformationskonzepts bildet den nächsten Schritt. Als Vorbedingung ist es jedoch erforderlich ein einheitliches Set an Funktionalitäten für Service Blueprints zu definieren, welches auf Vorschlägen aus der Fachliteratur sowie auf Erkenntnissen aus der Entwicklung der Case Study beruht. Um zu belegen, dass die entworfene Transformation geeignet ist um Prozessmodelle aus Service Blueprints zu gewinnen, ist der finale Teil der Diplomarbeit der Entwicklung eines Prototyps gewidmet. Um in diesem Kontext die IT gestützte Erstellung eines Service Blueprints zu ermöglichen wurde eine individualisierte Library für die Modellierungssoftware *Draw.io* entwickelt. Die Durchführung zahlreicher Tests mit der neuen Implementierung hat gezeigt, dass die geplante Kombination der beiden Modellierungssprachen tatsächlich möglich ist. Durch die unterschiedlichen Abstraktionsebenen beider Seiten sollte das resultierende BPMN-Modell noch zusätzlich erweitert bzw. verfeinert werden um auch alle Erfordernisse abzudecken.

Abstract

The master thesis at hand belongs to the area of service design and modeling. Due to the influence and importance of this intangible phenomenon within the context of today's economy and daily life, there is the need to have a structured way for their development, analysis, documentation, and communication. In this regard, Service Blueprinting is a well known graphical modeling concept with a special focus on the customer's point of view. Because of its more abstract nature, in order to be able to satisfy the demands imposed on modern workflow and process management, it might be necessary to use finer grained and more structured approaches as well. Accordingly, the target of this thesis is to examine the possibility to contribute to the overall topic while using the well-known service modeling concept in combination with the Business Process Model and Notation (BPMN). The basic idea is to use a given service representation illustrated via the more abstract methodology and automatically generate the corresponding process model in BPMN. After the elaboration of the general concepts and principals included within the overall context, a case study targeting the *emergency admission of a patient at a surgical ward within an Austrian hospital* is discussed. Within a next step, the theoretical concept for the conversion of the specific notational elements is created. As a prerequisite, due to the missing standardization of Service Blueprinting, it is necessary to specify a functional tool-set under the consideration of several propositions made within the literature and insights from the practical application. As evidence that the developed transformation can be actually used to generate a valid BPMN model based on a given Service Blueprint, the development of a prototypical implementation of the necessary mechanism builds up the final part of this thesis. For this purpose, a customized library for the modeling environment *Draw.io* is created which enables the usage of the defined notational set for Service Blueprinting. Applying the newly designed functionality that was implemented in *Java* on various test models and service representations that are a result of the mentioned case study, it has been shown that it is, in fact, possible to use BPMN as a complementary representation and automatically generate a corresponding illustration. Due to the different granularity of the two methodologies, the result of the transformation should be subject to further refinements to include all details that are necessary from the organizational point of view.

Contents

CHAPTER 1

Introduction

Services are phenomena with an intangible nature that occur in many aspects throughout our daily life [Bus18]. They are constantly present and used whether we are on our way to work, doing some shopping, contacting the manufacturer of a broken dishwasher, seeking medical assistance, or continuing one's education. Over the past fifty years, the society in industrialized countries has shown an evolution towards a service economy [Sho11]. Before that, the most important economic factors were related to the manufacturing area where the focus lies on the creation of physical and tangible products. This has changed and services are now at the heart of the global economy. Companies like Amazon, Google or Apple were able to use very well developed and unique services and processes to distinguish themselves from their competitors and gain a quite strong position on the respective marked. Another example is the company Uber which provides an innovative approach towards the issue of personal transportation. In fact, the offered service was that well adopted by the potential customers, that a rapid international growth could be achieved in a short amount of time [Dog17]. So it happened that what started out in 2009 in San Francisco is now a service that is available in 632 cities world wide [Ube17]. In contrast to regular taxis, Uber uses a smartphone app to link the customer to the infrastructure. If you are in need of a transportation, the program displays the drivers that are available near your location, information about the cars and how much it will probably cost you. Furthermore, also the payment is performed electronically. Another specialty lies in the fact that everyone who fulfills the requirements listed on the homepage of Uber can register him- or herself to be a driver. As such you do not have a fixed schedule but are free to drive whenever you want. Again the communication with the customer works via the smartphone and no additional equipment like a taximeter is required. Since very critical characteristics are the real-time communication and the ease of use for all participants, the overall idea would probably not be possible without the use of modern information technology. So looking at this example we can see that a very important backbone of the service consists of the applied infrastructure (i.e. the

app infrastructure linking the driver to the smartphone of the customer). However, this observation can also be made when reviewing various other very innovative services that currently reshape the way how we handle issues of our daily life. Take Amazon for instance. Starting with the online sales of books, nowadays you can purchase nearly everything via this platform. The subscription for the scheduled delivery of toilet paper is just one example.

As a conclusion, we can say that along with the economic shift towards the service economy and the impressive number of new technological opportunities that were invented during the last century, the innovation potential that is inherited by the service area is also vastly increasing. As a future outlook, Blake Morgan addresses in a Forbes article the rather popular topic of the Internet of Things (IoT) where a growing number of established daily life technology is linked to the communication potential introduced by the networking capacity [Mor16]. These IoT devices then have the possibility to automatically communicate with external sources and so have the potential to enhance the overall service quality. An example would be a broken washing machine. Currently, the owner has to contact the manufacturer (mostly via telephone) and organize the appropriate steps. In many cases, as a regular customer, probably more detailed technical knowledge is not available and only the symptoms (e.g. rattling noise, leakage of water) can be described to the help desk. So the first step consists of a professional error analysis on sight. Now let us assume that the washing machine has the capabilities to run some self-diagnostics and is linked to the network. Not only could the device gather very detailed information concerning the present issue (e.g. its nature, affected technical module, the exact time of its first occurrence, circumstances of occurrence) on its own and send it to the service center of the manufacturer, but also resolving instructions could be returned automatically. In some cases, it is even thinkable that the machine repairs itself using some implemented maintenance routines (e.g. descaling, increasing the automatic service intervals).

Modern services that are considered as being valuable for all participants (e.g. customer, vendor, supplying company), often rely on information technology as described above. But there is one aspect that is a more basic requirement and at the same time a very critical factor for the success of the service. The orchestration of the service flow and the underlying processes has to provide a smooth transition between the individual steps, meet the customer's needs for a satisfying experience and still remain valuable to the providing organization. So it is probably safe to say that currently, we live in a time where the composition and the quality of business services and the underlying processes are more important than ever and that the results are critical for successful companies [Gui12]. It is not an easy undertaking to come up with offerings that meet the requirements mentioned above. But even if such a service already exists, due to the fast-changing nature of markets and environments, the need for continuous enhancements and improvements is definitely given [Mor16].

To address this issues, and establish and maintain a service as an important unique selling proposition (USP), an organization needs to think about a structured and formalized way

to plan and describe the overall structure as well as the details that contribute to the experience. It shall be possible to support the design and development of new services, the documentation of already existing ones and the analysis of possible weak points and failures [BOM08] [BY05] [SDSB09]. According to Geke Van Dijk, Bas Raijmakers and Luke Kelly there are several tools that can be applied during the lifecycle of a service to support the intended purpose [DRK16]. A methodology that is probably one of the most known and widely adopted approaches targeting this topic is called Service Blueprinting. Originally it was developed in 1984 by the bank executive G. Lynn Shostack who published an article in the Harvard Business Review called "Designing Services That Deliver" [Sho84]. Service Blueprints are developed by scientists in the area of service marketing and over the years different approaches and enhancements have been introduced. Basically, it is a graphical modeling technique that is created while focusing on the customer's perspective of the interaction that should be captured. Despite the fact that basically there is no universal standard (e.g. in contrast to UML), typically some kind of flowchart notation is applied describing the actions and the tangible objects that are subject to the service. A key aspect of the methodology consists of lanes that organize the modeling canvas and classify the process steps according to the entity that is responsible for its execution. Examples are the customer's actions (e.g. calling a restaurant), the actions performed by the organization that the customer is actually able to whiteness (e.g. confirm reservation), the actions of the organization that the customer cannot whiteness (e.g. key-in booking information into the system), as well as other categories like physical evidence (e.g. receipt, employees) or managerial activities. Service Blueprints are often applied in a collaborative way using a cross-functional team including members that ideally represent every participant of the final workflow [DRK16].

As described above, the use of information technology is a key component when looking at the implementation and execution of modern services and the corresponding processes. In order to support this fact and in general the increasing requirements of successful services, a detailed model of the planned undertaking is necessary. To enable the transformation towards the underlying information technology, but also from the point of view of an organizational quality management, it is important to capture the service in a more coherent and standardized way. But due to the abstract nature of Service Blueprints and of course the lack of consistency, the concept does not fulfill this requirement on its own. However, a more standardized and finer grained modeling approach like the Business Process Model and Notation (BPMN) might also not be the best method, since it is rather difficult to use, especially without the required knowledge concerning notations or process modeling in general. The reason for that lies simply within its inherent complexity that is necessary to illustrate a process in all its details. Furthermore, BPMN might not be the ideal solution to pitch an idea, particularly when it comes to a collaborative design process [Bar11].

So, despite the importance of the design and maintenance of services within the context of today's business environment, the level of modeling support is not yet satisfying. A possible solution could be the usage of both methodologies, but since they lack a common

basis, a created Service Blueprint has to be manually translated into BPMN. This is not just time-consuming but tends to be rather ambiguous since there is no clear formula describing the transformation process itself.

To overcome these limitations and aid the creation, documentation, communication, and analysis of services, a (semi) automatic transformation of Service Blueprints into the BPMN could be introduced. Since it is already a characteristic of BPMN to support the translation of the designed models into executable structures, it also should be a goal to extend this automation towards the more abstract level of Service Blueprinting. As a long-term vision, a fully automated transformation could be approached.

Considering a practical application of this constellation, it would, for instance, be possible to first use a Blueprint to draft the service concept. This could happen in a collaborative way as a joint undertaking of persons with very distinct views on the topic at hand, like process or business experts as well as persons responsible for the development of the organizational workflows. Only one member of this group is required to have the modeling skill-set to apply Service Blueprints. Afterward, the automatic transformation generates the corresponding BPMN model. This one can then be applied to add additional details like exceptions, detours or links to other workflows.

Such a conceptualized mapping could overcome the mentioned inconsistencies, make the overall service design process more efficient and, furthermore, constitute the initial step towards the automation of service provision. For this purpose, the idea shall be approached within the following chapters.

At first, the theoretical concepts of services, Service Blueprints, as well as the Business Process Model and Notation, shall be discussed in detail. Due to the fact that during the development of the transformation it is necessary to be able to rely on as much knowledge as possible about the characteristics and mechanisms of both sides, this step is a rather crucial one for the whole undertaking. Especially since Service Blueprints do not have one, well-defined standard, different approaches towards this modeling technique that were introduced during the last decades shall be reviewed. In addition to the method itself, also enhancements and combinations with other concepts like Event-Driven Process Chains (EPC) [MML10] shall be examined. The actual transformation constitutes the heart of this work. To support the creation of the concepts, contribute to a better understanding of the applied methods and simply to show how the automatic combination of Service Blueprints and BPMN could work, a practical case shall be developed along the way. It will be specified using a verbal description as well as the application of the service design methodology. The subsequent chapters will introduce the actual linkage between the two sides of the conversion. To do so, first of all, the differences and commonalities will be analyzed and documented. Using this basis, several approaches towards the solution of the underlying problem will be developed that differ in notation and nature of their functionality. After the description of their inherent characteristic as well as their advantages and disadvantages, a recommendation will be made, pointing out a single set of transformation rules. It will then be used to develop a prototypical implementation of the achieved mapping. When looking at the computerized creation of BPMN, one

can already find rather sophisticated tools that offer a wide range of notations and characteristics included within the standard. The other side which consists of Service Blueprints does not enjoy such a wide-ranging support. Nevertheless, the creation of a whole new application from scratch does not seem the best way to approach this topic. So to target the prototypical implementation, the first step will be dedicated to the analysis and determination of possible solutions involving already existing and established applications. Once an appropriate technical environment has been created, the previously designed conversion rules will then be used to perform the transformation and serve as a proof of concept.

The chosen approach of this work shall represent a contribution to the topic of service creation as well as service maintenance and enhancement. Furthermore, the following research questions will be answered along the way.

- What is the common basis of the concepts Service Blueprinting and BPMN (especially with respect to the linkage between both methodologies)?
- What are the limitations for the creation of a semi-automatic transformation between the two sides?
- Is it necessary to introduce more formal characteristics or restrictions to Service Blueprints to be able to aid the transformation and if yes, what are they on a conceptual basis?
- When considering today's business process modeling, are there any enhancements that could be introduced to Service Blueprinting that might aid the overall development?

CHAPTER 2

Basics and Related Work

This chapter is dedicated to the description of the concepts and characteristics of the basic components that are relevant when considering the transformation of Service Blueprints into the Business Process Model and Notation (BPMN) later on. This includes the definition and nature of services, their development, and management, the modeling technique Service Blueprinting as well as the basics of BPMN. As already described above, this step builds up the foundation for the conceptual development of the transformation that is the main purpose of the overall work. It is necessary to be able to rely on a comprehensive theoretical basis that will positively contribute to the achieved results when looking at the combination of the underlying concepts. The final component consists within the side-by-side comparison between the two methodologies and can be considered as the first step towards the desired transformation.

2.1 What is a Service

The term *service* is very well known and probably everyone has somehow an understanding of the phenomenon that is described while using this word. But as it is often the case with such concepts, when asked what it is exactly, which characteristics are responsible for the classification and where it ends and another begins, this issue is not so clear anymore. However, since it is important for this work to have a clear and well-defined perception of the basic idea, the following paragraphs will try to highlight the nature and variants of services.

During the introduction of this thesis, the importance of services for our daily life as well as for the economic systems in modern industrialized countries was emphasized. Already when looking at the mentioned examples of Amazon and Uber, one can observe some very basic but essential characteristics. First of all, a service is basically not static or solid [BOM08]. It rather inherits dynamic properties and consists of several processes. A classic example would be the treatment in a barbershop where a shave or haircut is performed in exchange for a specified amount of money. It is obvious that the respective treatment is not something that exists physically at any time. It is a procedure that is carried out with the intent of a specific result, the agreed cut.

This finding entails another aspect that is a direct consequence. Every transition and every process consumes an amount of time [FLM04], for example, the time that is needed to wash someone's car. The duration that is required to come to a result may be pre-defined and fixed or be varying from case to case. This depends for once if the performance heavily depends on human interaction, which is, of course, harder to schedule and standardize than mechanical interaction [Sha10] [SDSB09]. Furthermore the nature of the service has to be considered. For example, in general, the trimming of one's beard is not linked to a specific time frame but to the style and the treatments. In contrast, when booking a massage one typically registers for a time slot (e.g. 30 minutes). Additionally to the fact that services consume time, the importance and value of this phenomenon may be different due to the type. Using the examples above, when it comes to a massage, the amount of time one receives the treatment can be considered as a direct indicator for the overall value (i.e. a 50 minutes treatment is perceived more valuable than 30 minutes). Looking at a classical haircut this is not the case since it is a rather outcome-oriented process. But this does not necessarily have to be the case. For example, it is thinkable that someone tries to increase the customer experience during such an offering to a level where the actual time of execution gains in importance and the value to the customer is more a combination of the outcome and the experience. So it is possible to create services that distinguish themselves from comparable ones while using an orchestration of its elements that is tailored to the specific customer's needs. This is also a reason why this area is so important for successful companies.

The next characteristic that was already implied above and can be observed when looking at the provided examples, is the fact that a service always inherits some kind of interaction between multiple sides [SS16] [ZBG06] [LW11]. The number of participants may vary

dramatically depending on the type and the exact nature of the used processes. Within the example of the barbershop, we have the salon as the organization and the customer as acting parties. The communication and the service provision is mostly executed in real time and in a very personal way, typically using phone calls during the booking phase and personal, on-site execution of the value generation. In contrast, the Amazon marketplace is an example where three parties are involved in the sales process. The retailer itself acts as a broker linking an external supplier indirectly to the customer using their online platform. The supplier initially registers him- or herself at the platform of Amazon submits the goods they want to sell and add additional information like the preferred shipping method (e.g. shipment via Amazon). At this point, when the customer browses the catalog on the homepage, the products of the external supplier are listed as well and can be purchased [Ama17]. From the customer's perspective, the overall process does not seem too different from products sold solely by Amazon itself. Only some additional information or confirmation messages show that the marketplace was used. Looking at the channels, one can see that there is no personal interaction between the three participants and everything is handled while using the internet's networking capabilities and automated processes including the payment.

As stated in the first paragraphs of this chapter, services have dynamic characteristics. Of course, in many cases, they are linked to physical elements that are necessary for the underlying processes or the resulting experience. In the barbershop, for instance, we have the salon itself, the chairs, the equipment, but also a receipt at the end and, of course, the new cut. In the area of Service Blueprinting, but also service design in general, these elements are often called physical tangibles [Sho84]. They are a very important aspect since they are a physical representation of the service, can be seen and used as evidence of the service performance and contribute to the customer experience. The later one could be especially relevant for a more traditional customer basis where expenses are often justified with physical possessions. Younger generations may not be that fixated on ownership and property and modern concepts like sharing and crowdsourcing gain in importance [Art16]. Another characteristic linked to tangible elements within the context of services is the ability to create an emotional attachment to rather simple objects. An example would be a souvenir of a beautiful holiday that reminds the customer of this experience. Even if it has no value on its own, the context implies an emotional and sentimental value [Art00]. Due to these effects, the used tangibles, their quality, the association to the service at hand, their volume and the circumstances during the transition phase need to be very well planned and orchestrated. For instance, when considering an educational diploma, it does make a great difference for the students and their relatives if the document is simply handed over quietly in the office of the deanery by the secretary or at a festive diploma ceremony. In fact, in this case, the circumstances are that important to most of the students, that they pay an additional amount of money only to take part in this supplementary process. The physical certificate can be seen as a tangible object of the educational service. It is of course evidence for the completion of the studies, but it is not the degree itself.

But tangibles have also the potential to negatively influence the customer's experience, for example, if the objects are not perceived as complementary to the service or feature a significant deviation of quality in comparison to the main performance [BOM08]. Furthermore, the provider of the service has to consider the possibility that one's offering does not necessarily result in a pleasant situation for the targeted audience. This could happen in cases where something went wrong during the execution (i.e. error, failure) or simply due to the nature of the service, as it is the case with the treatment at a dentist. If so it is probably not the best idea to introduce some tangible evidence that is meant to create emotional associations towards this negative experience. Another aspect within this context concerns the fact that basically, it is not possible to compensate or even justify services that inherit a generally bad design. As emphasized above, tangible objects are a very important ingredient when it comes to the service orchestration, customer satisfaction and the identity of the performance. They can be seen as an element of proof and build up trust in ones offer, but they do not have the capability to even out shortcomings of the overall concepts since, despite the fact that tangibles are a powerful complement, in general, they are not at the center of the value composition.

In contrast to the physical components that were described above, services itself are of an intangible nature [Bus18]. As a result, or for a reason, one cannot touch the processes, they do not have stock levels and cannot be stored [Bha17]. There is no possibility to duplicate a specific instance without repeating all the steps. Furthermore, it is not an easy undertaking to measure a service or compare multiple ones [SS16]. It has to be noted that in general there is not one correct way how such an offering should look like, how the interactions are handled, what the tangible objects should be or to which extend the customer shall participate at the service. All these points strongly depend on the purpose of the undertaking, the targeted customers, the operational and social environment, as well as the providing organization and its image. So it is not only important to know your audience but also to know yourself and the environment both of you are in.

The last lines point out another interesting and rather unique characteristic of services. The fact that the customer him or herself is a very important part of the concept and is, in contrast to other areas, an active participant that has the potential to positively or negatively influence the experience, the performance, and the outcome. Let's take one's education as an example. Basically, the student is the customer and without him or her the execution of the offering would not be possible since an essential part of the value generation is missing. Additionally, the student can influence the efficiency of the provided lectures. If he or she is not interested in learning and does not show the effort that would be necessary to reach the intended level of knowledge, even the best teachers and educational system do not stand a chance. The degree of customer involvement depends of course on the nature of the service. In an article with the title *An Explication of three Service Business Process Modeling Approaches* Yahya Kazemzadeh, Simon K. Milton, and Lester W. Johnson chose the example of a pizza restaurant to explain this issue in more detail [KMJ14]. They state that customer involvement basically is necessary to achieve the intended customized experience. The higher the involvement

the bigger the customization. Using their example, for the restaurant it would be possible to include the customer already during the assembling process of the pizza and so tailor the product to their individual needs. This could potentially increase the customer satisfaction depending of course on their initial expectations. The authors furthermore state that there are twenty-five different ways to handle the two steps *Assemble pizza* and *Cook pizza* within the overall process.

In general the degree of customer involvement is the major catalyzer for the service complexity and, as a result, has the potential to reduce the efficiency of the processes [KMJ14]. According to Christopher Lovelock and Jochen Wirtz, the ultimate form of customer involvement can be found in services that are designed to give the target audience almost complete control over the execution of the service steps and concentrate more on the provision of the necessary environment and infrastructure [LW11]. The use of these so-called self-service technologies (SST) are more and more important and especially when it comes to administrative tasks like airport check-ins, banking or government agency issues (e.g. online tax declaration, e-government) this approach has great potential and already made an impact in our daily lives. But these designs require a very good system to support the execution of the individual steps. When it comes to information technology people want it to be functional and they tend to get frustrated really fast if there is an error (system or user inflicted) and are forced to contact the supplier to find a workaround for their request [LW11].

As already implied above, the customer involvement is tightly linked to the critical sections and processes that are part of the service orchestration [Sha10]. The reason for this lies in the nature of human beings and in dealing with such. In contrast to the interaction with machines like electrical systems, each human being is unique in one's values, attitudes, beliefs, fears, prejudices, and behaviors. Since the target of a service is, to at least not inflict a negative experience, to some degree the characteristics of the customers need to be taken under consideration and every person could require a different treatment. Additionally, human communication may be subject to misunderstandings and is potentially error prone [Woo97]. To be able to meet the specific customer needs and to support a positive overall experience, these critical points must be meticulously targeted during the design phases of the service. However, during the execution of the individual process steps, human interaction requires time and flexibility, which results in the fact that it remains potential critical for the efficiency of the service. But adequate planning is a basic requirement to deal with possible errors and to come up with customer friendly alternatives that minimize the occurring negative impacts. In general, one can say that the way how and to which extend the service orchestration and the underlying process flow is designed under consideration of potential failure points, detours, and compensations, is a sign of service quality.

The description of services their definitions, characteristics, and specialties could be examined on several hundreds of pages and even then there would still be some new observations that could be taken into account. Since the purpose of this chapter was to emphasize the subject at hand and to investigate its nature that needs to be considered

during the targeted transformation, only a small introduction into this economic and social phenomenon has been provided. The upcoming part shall focus on the question how the literature proposes to approach the topic of service development and management.

2.2 Designing Services

The need for planning and designing of services and the documentation of their characteristics, to be able to reevaluate the performance within the context of a changing future environment has already been addressed during the previous chapters. Over the last decades, along with the growing relevance of the service area for the economy and the success of companies on the respective markets, authors from the academic area as well as from the different areas of daily economics (e.g. strategists, marketers, designers) have targeted the issue of the initial development and ongoing enhancement and management of services. Due to their dynamic characteristics and the strong interconnection to the various parties during the execution phases, the design process has the potential to critically influence the outcome of the introduction of a new offering and can be directly associated to the failure or success of such an undertaking. Additionally, when looking at the focus of this work, that consists within the creation of a transformation from a service modeling approach towards the more process-oriented world of BPMN and is intended to support the development and management of services, the issue at hand and the consideration of the approaches introduced in the literature, can be seen as equally important as the definition of services and their characteristics. To approach this point, after a small historical remark, the nature of a development process, the contained steps as well as some tool-sets and methodologies that could provide support, will be described.

As stated in the introductory chapter, the dawn of the service economy dates back approximately fifty years. Over the time not only there was an evolution when it comes to the nature of services, but also when looking on the way how their development is approached by the providing organizations. In an essay, Lucy Kimbell describes the topic of service design from a marketing perspective which is an interrelated area that also concentrates on the costumer's requirements rather than the organization itself [Kim16]. She points out that with the broad availability of all products after the boom of the world economy (i.e. the economy was no longer fixated on the scarcity of products), the focus of the companies shifted towards understanding the human needs and how to use this knowledge to distinguish themselves from the competition. As a result of this situation, the customers have gained structural power and the ability to influence the environment since the companies abandoned an approach where they dictated how and what the audiences had to buy. New ways needed to be developed how to approach end-user driven products and, especially since the 1970s, services.

Probably one of the most important insights concerning this topic that was captured by the literature is the statement that a service process, that is tailored to the needs of its target audience and inherits a sustainable over time development, needs to be approached in an interdisciplinary way [BOM08]. That means that during the process of

designing and developing a new service or analyzing and enhancing an already existing one, ideally multiple persons, with different professions and points of views, shall participate. According to Geke Van Dijk, the participant of this joint undertaking should be a so-called T-shaped individual. This term describes that each team member should provide a broad general understanding of the whole topic and a very detailed knowledge when it comes to his or her specialty [Dij16]. But it is not sufficient to only include marketers or designers. One essential point is to gather information from all parties that interact with the service later on. To do so, besides persons from different design and strategical areas, also representatives from the customer's side as well as service personnel and employees who are responsible for the operational business later on (also called secondary users [Mie16]) should be integrated during the process.

At this point, one could probably ask the question how the service development team should exactly look like besides the integration of customer representatives as well as front- and backstage service staff and which organizational fields are responsible for the development. According to Marc Stickdorn and Jakob Schneider, service designers can be found within the areas of product design, graphic design, interaction design, social design, strategic management, operational management and design ethnography [SS16]. The expertise that is actually required when it comes to a specific case, of course strongly depends on the nature and the characteristics of the service at hand. But since services are multidisciplinary and have the potential to link a multitude of organizational and social aspects, some knowledge about the areas mentioned by Stickdorn and Schneider will be necessary to consider and optimize all essential aspects of the intended offering.

However, due to the various dependencies and influences to other areas, service design on its own cannot be considered as a stand-alone discipline [SS16]. When looking at the definition of the process and its characteristics, Stickdorn summarizes five properties that need to be taken into account and fulfilled when approaching the development steps [Sti16a].

User-centricity is the first principle and emphasizes the importance of the people that interact and work with the services during its execution. As mentioned before, of course, the main actor at this point is the customer, but also the secondary users (i.e. the service staff) have to be considered. Due to this demand, several approaches in this area (e.g. also Service Blueprinting) try to view the planned offering from the customer's point of view. This way critical aspects and weak-points are detected more easily. Most of the time design processes that are efficient for the own organization are not a big issue since economic theory suggests various performance indicators that can be used. But when it comes to the needs of end-users and their experience during the service, the evaluation can be quite complicated.

Co-creativity describes the next principle and can help to overcome this issue. It is important to understand, that depending on the service topic, there can be various customer types with very different values and needs. User representatives should participate during the development process which will result in a more tailored and satisfying service. But also in general, there is the necessity for the development process to be an interdisciplinary

one as already described above. Besides the widened area of expertise, this contributes to the fact that coming up with a new service and specifying its details, is also an issue of creativity and innovation. A collaborative approach can positively influence the quality of the final solution. In addition, according to Stickdorn, a co-operational approach promotes a less error-prone execution phase and a strong customer integration facilitates increased loyalty later on.

Sequencing is about the dynamic nature of services. They consist of multiple processes that can be further divided into individual process steps which are then carried out during the execution phase. Since the goal is to plan the offer as well as possible and to apply optimizations and fine-tuning where ever it is possible, during the development and the maintenance of services, these steps are identified and addressed individually. This does not only concern the interaction points front stage, but also supportive and managerial activities. However, one thing that should not be neglected is the fact that in addition to its components, a service is also perceived as an entity on its own. Therefore the overall orchestration is important as well.

Evidencing means that within the context of a service, even if it runs basically in the background and remains mostly unnoticed by the customers, he or she should be aware of the performance. Otherwise, there could be some surprises when it comes to the payment of such. According to Stickdorn, some services are intentionally designed to be subtle and unnoticed (e.g. housekeeping in a hotel). But even so, it is necessary to give some evidence to the customer. The extent and the chosen method, of course, depends on the nature of the offer itself. Such pieces of evidence have the potential to add value to the performance and create an emotional association. In that case, it is possible that the customer's experience is prolonged even after the end of the actual execution phase. In addition, tangible objects of services can be, and probably often are, used to support the users during the process and especially when it comes to critical and more complicated steps. One of the most important aspects of this area is the fact that the customer does only accept the provided pieces of evidence if they are consistent with the characteristics of the context. Only then it can be used to achieve the advantages described above.

Holistic describes the way how a service should be considered during the design and management process. It emphasizes the viewpoint that it is important to address as many aspects as possible and evaluate them under consideration of the customer's experiences and feedback. It is probably necessary to repeat specific development steps with a special focus on certain points to facilitate the best outcome of the undertaking. One issue at this point that needs to be addressed, is the implementations of fallbacks and detours in case of an error. Only if these are already considered during the planning phase, it can be guaranteed that the service remains valuable to the customer.

Due to the inherent feedback loops and the demands of the co-creativity, the service design process has nonlinear properties. However, it would be incorrect to assume that this makes it impossible to address this issue while using a structured development approach. In fact when considering the creation of a new service (or the documentation and analysis of an already existing one) the initial step should be the outlining of the iterative design

process itself. Of course, it is necessary to take the service into consideration and so to come up with a more tailored resolution towards the problems and goals at hand [Sti16b].

According to Stickdorn, on a meta level, a very simple structure can consist of the steps exploration, creation, reflection and implementation in an iterative way [Sti16b].

The first one, **exploration** is about the overall situation, the problem that should be addressed and its environment. This includes reviewing the own organization and its viewpoint concerning the main issues. But of course it is necessary to focus on the customer while developing a new service and so to satisfy this demand, also this side of the value exchange has to be considered at this point. This step is finalized with the visualization of the results from the analytical processes and is all about the identification of the fundamental problem that should be solved. All the subsequent steps only make sense and have a chance to result in a well-designed service that is highly valuable and desirable to all participants, if the goal and the underlying shortcomings are identified and described in detail.

The next step, **creation** is as its name implies strongly focused on the generation of comprehensive approaches targeting a solution of the problems that were identified before. At this point, it is very important to apply the property of co-creativity to include all the different values and demands of the individual stakeholders within the context of the resulting service. Due to its creative and innovative nature, such a design process is not a straight road without any detours or dead-ends. In contrast, it is of an exploratory nature and therefore includes making errors. The important thing is to learn from the mistakes and to incorporate the new information during the next iteration and so approximate the final, efficient solution.

To do so, the subsequent part is called **reflection** and is of course tightly linked to the creational aspect. During the whole design process, these two steps are performed iteratively to increase the quality of the desired solution. The perceptions and the individual service concepts that where a result of the preceding phase are tested and evaluated. The generated insights are then passed on to the next iteration. According to Stickdorn a major aspect at this point is connected to the intangible nature of services. To conduct the necessary extensive evaluations, the generated concepts need to be analyzed under somewhat realistic circumstances. It can be quite complex to create a plausible prototype, that can be used to simulate the service steps during their execution phase and include the concerned stakeholders, to be able to extract valuable feedback information. Especially the service interaction points between the individual participants should be somehow applied at this stage. In any case, similar to development processes in the area of software engineering, the sooner an error can be identified and countermeasures applied, the lower are the costs. A point of failure only results in minimal expenses during the early stages of the design process, but when it comes to later stages, where most of the process details are already fixed and communicated, it can cause a serious development issue and delay the project's schedule.

Probably after performing several iterations, it is time to **implement** the now satisfying

service concept. This final step focuses on integrating the new offering into the operational area of the organization. It is very important to include the employees that are responsible for performing the value-adding process steps at this point. In general the aspects of change management are very important to ensure an efficient transition towards the new service.

Considering the demand that this topic shall be approached in a structured manner, several methods and tools might be necessary to support the efforts of the responsible project team. Within the literature different types and ways how to apply them can be found. Due to its nature, the selection of the applied methodologies should be made under consideration of the characteristics of the service, since not all tools are equally suitable for the current problem. Another aspect that needs to be taken into account when thinking about the toolset is of course the present know-how of the team members as well as the culture of the organization [DRK16].

However, looking at the steps as described above, the chosen methods also may differ depending on the current project phase [DRK16]. It might be necessary to include multiple methodologies considering one project phase, experiment with their applicability and discard it if it proofs that it does not contribute to the progress of the development.

Looking at the stage of exploration, the use of so-called Customer Journey Maps could be a possible way to go. It is based on a structured visual modeling approach and is used to start-up an idea and gather initial information. As it is quite common in the domain of service design, it is all about the customer's experiences. To retrieve the vital insights, interviews could be performed. But also the active participation of the end-users could be a viable approach that enables the team to harvest the experiences at first hand. Once more it is recommended to try to integrate the customers during the design of the individual project steps. To do so, the initial focus should lay on the identification and linkage of the various touchpoints throughout the journey. Afterwards, it is possible to go into more detail when it comes to specific critical steps. The visualization, as shown on an example in Figure 2.1, displays the process flow on a meta level and should be understandable by all service stakeholders. Target is to capture the customer's interactions, encounters, and tasks under consideration of his or her emotional state during the overall journey.

When it comes to the iterative stages of creation and reflection as described by Stickdorn, a method is needed that tries to incorporate more details of the service and provide a basis for generating necessary feedback information. Using so-called Storyboards is a possibility to capture not only the properties of the offer itself but also of its environment and context. Once more it consists in a graphical visualization and focuses mainly on the events that are directly perceived by the customer. This could be a comic strip, picturing all important steps and situations of the service. But also photographs of actual or staged situations could be used which probably makes it easier to create an image of real-life events. The main target is to trigger discussions afterward and so generate valuable insights into weak points, possible failures, critical sections, but also into opportunities and potential enhancements.

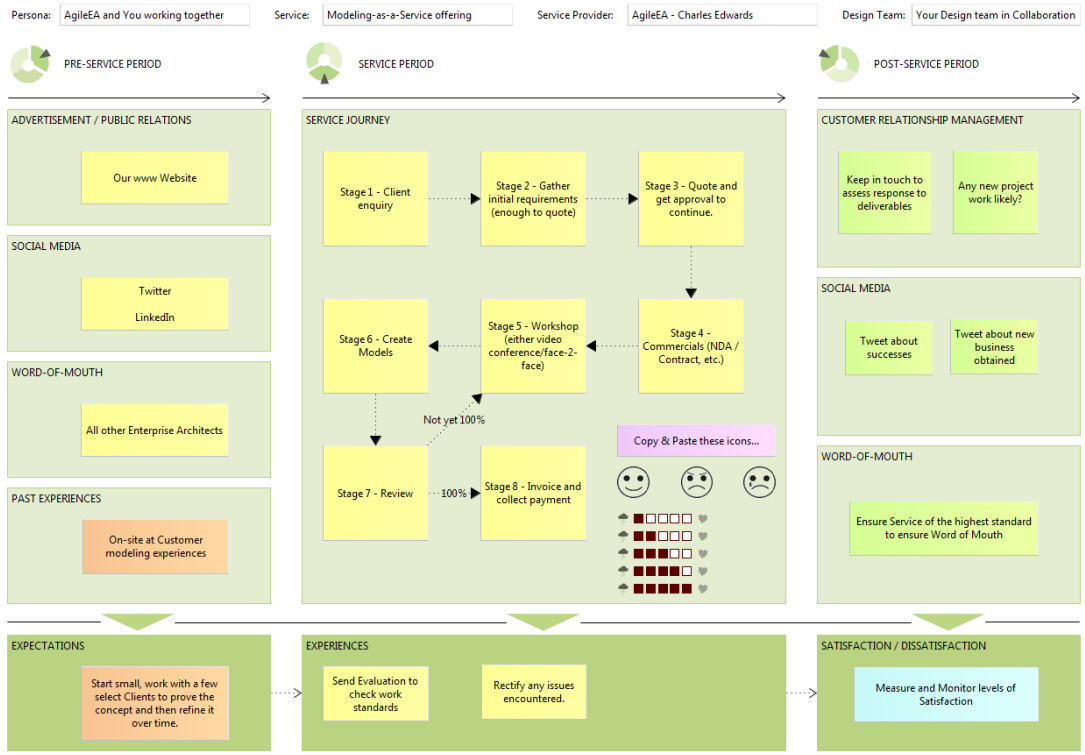


Figure 2.1: Customer Journey Map Example [SS]

Considering the last step, the implementation, the well known and widely adopted Business Model Canvas could be a part of the supporting set during this stage. It is a method to generally document business models. Due to its ability to use it in a collaborative way (e.g. sticky notes) it is also applicable within the domain of service design and can be used to clarify the concept of the intended offer. Basically, it is a fixed table containing sections for key partners, key activities, key resources, cost structure, value proposition, customer relationships, channels, revenue streams and customer segments [BCOR11]. The target is to use the provided categories to specify the planned service or document an already existing one.

As mentioned above, of course, there is a multitude of different methodologies available to aid during the specific development steps. For example, another tool that could be applied within the context of the last phase is Service Blueprinting. Due to its importance for the topic of this work, it shall be described in detail in the next chapter.

2.3 Service Blueprints

Service Blueprinting is a methodology to support teams and organizations when it comes to the invention of new services, the documentation and evaluation of already existing ones and in general the communication of the underlying ideas and concepts. Looking at the goal of this work, this tool is the first component of the intended transformation between Service Blueprinting and the Business Process Model and Notation (BPMN). Hence, due to its importance, the upcoming chapters will explain the characteristics in detail. To do so, after some historical remarks explaining the origin of the methodology, the design process of such models shall be discussed. Afterwards, the concepts applied in Blueprints, as well as their graphical notation, have to be examined. Furthermore, the benefits but also the shortcomings and weak-points are considered to emphasize the applicability of the modeling language. The last section is focused on enhancements and combinations with other methodologies that were suggested within the literature.

2.3.1 Historical Remarks

The origin of the methodology can be traced back to the year 1983. The bank executive G. Lynn Shostack published an article in the January 1984 issue of the *Harvard Business Review* with the title *Designing Services That Deliver* [Sho84]. At the very beginning of this text, Shostack describes a situation that is valid until this day. It is all about a lack of service quality as it can be observed while experiencing various services throughout one's daily life. She notes that although many shortcomings originate from human factors, a very critical aspect consists within the underdevelopment of more structured approaches towards the design of new service offerings (especially in the 1980s). Furthermore the author criticizes that even if there are some kind quality controls in place, they only focus on specific components of the overall concept and do not provide a more holistic approach. In general, organizations often engage in trial and error approaches to set up a new service, which may inherit some functional capabilities but do not result in an offering that does satisfy the actual needs of the customers. Additionally when it comes to the sustainability of already established services, without a more structured approach it is easy to miss the need for future developments and enhancements. So what once was a great idea could soon be obsolete. And even if there is the strive towards the creation of an improved concept, without such a support during these critical phases within the service lifecycle, it may be very hard to pin down the important aspects and align the ideas with the actual requirements of the customers. In her article, Shostack describes a service as a larger concept that consists of a multitude of individual components (e.g. process steps, tangibles). It has to be considered in a more comprehensive way to satisfy the need for a structured development of services that is worthy of the economic importance of this topic.

Since visual modeling tools like simple flowcharting do not provide the customer-centric focus that is necessary and neglect the special characteristics of services (e.g. intangible nature, the importance of interaction points), the author proposed the development of

Figure 2.2: Portrait of G. Lynn Shostack [Pri]

a so-called Blueprint to overcome these issues. But, due to its precision, it should still consist of a graphical approach. One of the most important requirements at this point, Shostack mentions the demand to create a model of the intended offering from the point of view of the end-user. Only then it is possible to identify weak-points and critical sections that are responsible for potential dissatisfaction which may result in a failure of the concept at hand. Since service design is also tightly linked to innovation and has the potential to generate USPs that build up important strengths on the market, the intended method has to support the creativity during the development process. In addition, Shostack acknowledged the role and importance of tangible pieces of evidence for the designed service and therefore the necessity to include this phenomenon into a potential modeling approach. Another characteristic consists in the observation that only a part of the individual steps during the execution phase is visible to the customer [Sho84].

As a result, the author proposes a graphical modeling technique and tries to incorporate the aspects as described above. It consists in the use of a basic flowcharting notation in combination with a fixed lattice separating the canvas in an area for the actions that are actually visible to the customer and those that are not. This arrangement is called *line of visibility* and is represented as a horizontal line. Furthermore, the concept of physical pieces of evidence has been integrated and the components itself, as well as the linkage to the corresponding activities, are visualized as graphical symbols. To capture the dynamic nature of services and the possible uncertainty that is inherent by the individual processes, especially at the execution phases where people are directly involved, also fail-points, detours, and loops can be illustrated.

Although it seems that Shostack had very specific ideas and suggestions when it comes to Service Blueprints, it was not intended as a fixed methodology or standard. Even the usage of the flowchart notation as shown in the article can be understood as a suggestion.

The way how it actually is applied may depend strongly on the specific case, the intent or the circumstances of its creation.

So it happened that along with the popularity of this tool-set, over the last decades and even within the more recent years, several authors have proposed enhancements, distinct graphical notations and in general different approaches towards the concept of Service Blueprints. One example is the developments of J. Kingman-Brundage who wrote several articles and papers about this topic and named it service mapping [KB89] [KB91] [KB93] [KB95]. Furthermore authors like Marry Jo Bitner, Christopher Lovelock, Jochen Wirtz and many more wrote papers, articles, books and other contributions introduction their own viewpoints and concepts towards this methodology.

2.3.2 Creating a Service Blueprint

Considering the whole process of service design from the very first idea till the implementation and maintenance of the components during the service execution, Service Blueprints may be used when it comes to several tasks along the way. As mentioned during the introduction of this methodology, it basically can be used for new services, but also for the enhancement of already existing ones. Due to its nature as a more or less precise graphical representation, depending on the used notation and constraints, it is suitable to communicate and document the intended service processes and underlying ideas [BY05].

When looking at the development phases as explained by Stickdorn that were described in a previous chapter, the method Service Blueprinting can be assigned to the phase of implementation [Sti16b]. Within this context Geke Van Dijk, Bas Raijmakers and Luke Kelly specify that the tool-set is typically applied in a collaborative way to create a detailed layout of the planned offering which also has the potential to build up a shared responsibility among the whole team [DRK16]. As main benefits, they mention the possibility to capture all important actors, activities, pieces of evidence and the ability to identify weak-points and critical sections.

The question at this point is how a Service Blueprint is actually created and what are the steps that need to be executed to do so. The collaborative nature while using a cross-functional team was once more mentioned above. Although the actual creation of the graphical representation may be carried out by one individual person using a computerized modeling tool or simply some sticky notes on a flip-chart, the stakeholders connected to the offering as well as domain experts need to be integrated to be able to rely on a more complete view on the topic at hand. Blueprinting itself emerged out of the need for a more structured and comprehensive way to develop and implement services. But it is not enough to introduce such a method to a design process and neglect the way how the model is actually created. Therefore not only the graphical representation needs to be considered but the steps that lead towards this result as well [Sho84] [ZBG06] [DRK16].

To address this issue, G. Lynn Shostack outlined a possible strategy in her initial article from 1984 for designing a Service Blueprint [Sho84]:

1. **Identify Processes** - As a first step the processes that shall be combined to create the service, have to be identified and captured using a graphical representation. The extent of this step depends on the degree of complexity of the intended offering and the steps that are necessary to carry it out. So the created models range from very simple and small to quite comprehensive ones. In general, specific aspects should be examined in more detail if they are of a more critical nature.

2. **Isolating Fail Points** - The customer's satisfaction with ones service strongly depends on the occurrence of fail-points and the way how they are handled by the members of the organization. Since services comprise multiple individual steps during their execution and are often related to several (human) actors, it is impossible to create a scenario where nothing goes wrong. To build up a proposition that is unique and valuable to the end user, it is necessary to know its weaknesses and address them accordingly. So the second step is about using the material at hand and to identify these critical sections. Afterwards the designers have to think about some ways to treat them.

3. **Establishing Time Frame** - An important aspect of services is their dynamic nature which is a logical conclusion of their composition. Similar to all procedures, this means that a specific amount of time has to be consumed to generate the desired output. Due to the relation to the costs incurred (e.g. wages of service staff) and its importance for the customer satisfaction (e.g. waiting for a doctors appointment) it is necessary to take this phenomenon already during the design phases into account. Hence the target of this third step is the determination of a standard execution time that is realistic as well as desirable by all participants. When looking at the previous point, this standard time may not hold in certain cases. Since these situations have the potential to negatively influence the perceived satisfaction, it is equally important to specify time frames for alternative processes flows and compensations.

4. **Analyzing Profitability** - The final step is tightly linked to the previous one (i.e. time as cost factor) and addresses the profitability of the service. In general the creation of a Service Blueprint has its focus on the customer's viewpoint and the satisfaction of his or her needs and requirements. At this final stage the idea that fulfills these demands shall now be examined according to their sustainability from the organization's point of view.

Another approach towards a design process for Service Blueprints has been suggested more recently by Valarie A. Zeithaml, Mary Jo Bitner, and Dwayne D. Gremler and consists of the following steps [ZBG06]:

1. Specify the intended offering and the corresponding processes
2. Specify the characteristics and the nature of the customers
3. Design the interaction points between the different participants of the service

4. Design the organizational front-stage actions in relation to the customer
5. Design the back-stage actions (e.g. support, managerial) and connect them to the front-stage
6. Identify and capture tangible objects and there connection to the individual process steps

Although basically, the two outlines share the same idea, when looking at the points above one can observe several distinctions. First of all the customer focus of the overall approach has been additionally increased within the suggestion of Valarie A. Zeithaml, Mary Jo Bitner, and Dwayne D. Gremler. This is represented by the second step which should answer the question who the customers actually are. Since it is crucial to understand the specific needs of all customers (normally there is more than one and they can be very distinct from each other) the additional highlighting of the end-users seems quite appropriate. The next difference lies within the treatments of the interaction points. Over the years there has been the insight that the activities and phases where the customer directly interacts with the organization (e.g. service staff, self-service technologies) have the highest potential to result in problems, delays, errors and, as a consequence, dissatisfaction. As a countermeasure, it is necessary to plan and design these interaction points very carefully. As a final observation, it can be noted that while Shostack suggested a rather general outline, the second listing is more directly related to the technique Service Blueprinting and its concepts, which will be explained in more detail in the upcoming paragraphs.

2.3.3 The Concepts

Within the context of this work, various literature sources have been reviewed under consideration of their distinct ideas. To explain the detailed concepts of Service Blueprints, the following paragraphs shall constitute a summary of the most important characteristics that have been observed during this examination. It shall be noted that the intended transformation towards the Business Process Model an Notation (BPMN) was taken into account and a special focus lies on concepts that may potentially contribute during the development of the necessary mapping later on.

Service Blueprinting is a tool-set that aims to capture all actions that are necessary for the value generation and occur during the execution of the service (e.g. customer's activities) or are related to its more general context (e.g. preparations, managerial activities of the organization). To do so, a sequence of actions is illustrated using an approach which is more or less similar to flow-charting and represents their dependencies and logical order. Furthermore, the elements are assigned to different categories which state the nature of their occurrence (e.g. customer or organizational activity) and inherent characteristics (e.g. visible to the customer or not, customer induced or independent). Using the applied classification, the interaction, as well as the communication between the distinct participants of the service, are displayed [KMJ14].

Considering the graphical character of the modeling technique, probably the most abstract components are the canvas and the applied notation. While the structure of the former one is mostly fixed after the start of the modeling process, the latter is used dynamically to capture one's ideas. Despite the fact that this technique suggests specific components and approaches, the elements are not compliant with a fixed standard as it is the case with other modeling languages like UML [LW11] [BOM08]. So it is mostly up to the author or the person who actually applies the method to choose what the exact components will be.

The modeling area of a Blueprint is separated using multiple horizontal lines. According to Mary Jo Bitner, Amy L. Ostrom and Felicia N. Morgan these are the *line of interaction*, the *line of visibility* and the *line of internal interaction* [BOM08]. On top of these, another separated area is used to capture the physical pieces of evidence (i.e. tangible objects). Furthermore, Sabine Fließ and Michael Kleinaltenkamp reference the additional separators *line of order penetration* and *line of implementation* which were also mentioned by Jane Kingman-Brundage [FK04] [KB95]. Fließ and Kleinaltenkamp also stated that basically when looking at a Service Blueprint, 2 dimensions can be observed. The horizontal one defines the sequence of the actions (e.g. customer actions, actions of the service staff) and the vertical one their area and nature [FK04]. The line separation as described above is used to create a fixed lattice and introduces this second dimension. The following points shall explain the concepts behind each area in more detail and keep the intended sequence of elements on the modeling canvas as proposed within the corresponding literature (see also Figure 2.3).

- **Physical Evidence** - Top area containing all relevant tangible objects (not actions) that might be linked to the service provision and are perceived by the customer. Due to their ability to influence the service flow as well as the end-users evaluation of the value of the service, this aspect of Service Blueprinting is quite an important one and should not be neglected during the development process [BOM08]. The horizontal axis can be used to capture the sequential order of their occurrence and associate them to the performed actions that define the context within the orchestration.
- **Line of Interaction** - This line constitutes the first actual separation between the different categorization of possible activities. It introduces the differentiation between the actions performed by the customer (above the line) and the ones performed by the offering organization (below the line). Directly beneath, actions are located that are actually perceived by the end-user as being performed as a contribution to the processes targeting the value generation. For example this includes activities executed by the front-desk staff of a bank or a hotel, but also the observable operations of a mechanical self service technology (e.g. e-governmental systems).
- **Line of Visibility** - The *line of visibility* splits the whole modeling canvas into two big areas. All activities that are located above are directly perceived by the

Figure 2.3: Empty Service Blueprint Canvas (created with Realtime Board [Rea])

customer (e.g. the customer's activities, actions performed at the service desk) and the ones beneath are not (e.g. passing on a room service order to the kitchen, preparing the food). Due to their nature, these areas are called front-stage and back-stage [LW11].

- **Line of Internal Interaction** - The back-stage area is then additionally divided using the *line of internal interaction*. It distinguishes between actions that are a direct response to the specific service case at hand (i.e. a specific execution of the service process; e.g. customer A wishes to have his beard trimmed) and activities that inherit a supportive nature and are necessary to perform the service in the first place [LW11].

- **Line of Order Penetration** - The support processes can then be further separated when applying the *line of order penetration* [FK04]. All actions located above are linked to the customer-oriented value generation (i.e. customer induced actions like preparations; e.g. sharpening the razors, restocking shaving soap), while

actives below have more general characteristics and are therefore independent of the customer [BY05].

- **Line of Implementation** - The last line that shall be explained in more detail is called the *line of implementation*. It finally introduces a distinction between the support activities as described above and management actions that are not specific to the service execution [MML10]. This area is also called the management zone and typically includes all managerial activities like planning, human resources, controlling, etc [BY05].

At this point, it shall be noted that due to the fact that Service Blueprinting does not follow a fixed standard, the selection and application of the desired lanes are basically up to the creator of the model and can vary depending on the know-how of the project team as well as the target of the undertaking. The lines as described above represent a selection of the most common ones in the literature. But of course, there are alternatives, respectively additional concepts as well. For example, Christopher Lovelock and Jochen Wirtz suggest the usage of the so-called *Line of Internal IT Interaction* to be able to illustrate the tight linkage of modern service processes to elements of the information technology. It is located at the very bottom of the canvas and creates an area that encapsulates IT elements (e.g. customer records within a database) [LW11].

Considering the horizontal line separation, one can observe that the modeling canvas is now divided into distinct areas that inherit specific characteristics as described above. When it comes to the identification and illustration of the actions that are necessary for the service at hand, the designers have to assign these actions according to their fit to the introduced classification. As a result, this vertical dimension expresses the categorization of the specific actors (i.e. customer, front-stage, back-stage, ...) and their actions [KMJ15]. The horizontal orientation on the other hand finally represents the sequence and the dependencies of the activities within their corresponding area.

Additionally, it seems rather obvious that when it comes to the assessment of the individual actions that are performed during the service execution, there needs to be some kind of interaction between the customer, the organization and the organizational agents among themselves. Furthermore, during the explanations targeting the characteristics of services that were approached within a previous chapter, the importance of the introduced interdependencies, the corresponding exchange of information and the resulting sequence of actions was emphasized and marked as a critical aspect of the overall issue. To satisfy this observation, of course, there is the need for Service Blueprints to capture these interactions which, as a result, happen between the various areas of the canvas. Following the outlined concept, the horizontal lines act as points of interaction between the different categories of service participants (e.g. customer, front-stage staff, support staff) and can be seen as interfaces [KMJ14]. Milton and Johnson refer to this interaction between the areas and therefore between the different actors that participate in the service process, as the communication flow [MJ12].

To be able to specify the intended offering in more detail and to contribute to its success, further concepts are the illustration of weak- and failpoints. These elements are of importance due to the fact that when it comes to services, the experience and its inherent, perceived value are potentially negatively influenced by a single aspect that does not work as expected and result in dissatisfaction on the side of the customer. In their book *Services Marketing: People, Technology, Strategy* Christopher Lovelock and Jochen Wirtz use graphical markers to indicate positions within the sequence of activities that may be subject to extensive waiting times (in comparison to the previously specified standard time) and possible failpoints [LW11]. Depending on the chosen detail and extent of the applied flow-chart notation, countermeasures and detours may be also captured within the Blueprint.

As stated before, there is no predefined way how to capture one's ideas within a Service Blueprint and how to illustrate them. Nevertheless, the applied notation, especially of the sequence of the activities, restricts the detail and the possibilities of the resulting models. Furthermore, the complexity may also be influenced. Since the main goal of this work is to use an existing model and transform it into BPMN the different possibilities and elements of Service Blueprinting are of great importance. To continue to address this topic, the upcoming paragraphs shall focus on the graphical representations of the model and outline different suggestions of the corresponding literature.

2.3.4 Graphical Notations

Before one starts to apply Service Blueprinting on a specific case, a convention has to be made, defining which graphical elements and representations are used to capture the targeted service and its components. If this step is neglected before beginning with the actual design process, the result may not be consistent within itself or in comparison to other models that were already created in the past. As a consequence, the Blueprint may lack quality and fail the intended purpose. In general, the used notation of a modeling language has a vast impact on the concepts and cases that can be illustrated using the chosen tool-set as well as the intended level of detail and complexity. To be able to create a very comprehensive abstraction of reality that is comparable with other models of the same language and be understood by people who possess the knowledge concerning the used methodology, a multitude of distinct mechanisms have to be applied following an agreed convention regulating and expressing their symbolic functionality and capabilities. Due to this necessity, popular modeling languages, especially ones targeting a more technical area of expertise or the representation of detailed process flows, often inherit a fixed set of graphical notations and concepts that are regulated and developed via a centralized standard (e.g. UML).

The term Service Blueprinting, on the other hand, describes a more general approach towards the design of services and does not possess a unified convention. It suggests the use of concepts, as explained in the previous chapter, but is not limited to those and especially does not restrict the use of graphical notations [Sho84] [ZBG06] [DRK16]. For the intended scope of this work it is necessary to examine the concepts and illustrations

as proposed within the literature to be able to rely on them later on when it comes to the actual mapping and transformation to the Business Process Model and Notation. To target this issue, the following paragraphs will describe some suggestions and display their graphical representations.

When looking at the basic concepts of the methodology as explained within the previous chapter, they can be summarized in the following list [MJ12] [KMJ15] [BOM08]. To transfer their basic intentions into the model, a proper graphical representation has to be applied.

- Modeling Canvas
- Line Separation
- Communication Flow
- Actions
- Action Flow
- Actor Categories
- Physical Evidence
- Additional Concepts

The first point describes the ground area of the final model and does not require a specific notation. However, the initial aspect that has to be taken care of during the design process, is the application of the horizontal separation of the canvas, since this part of the model should be fixed until the illustration of the actual action flows. For this purpose, typically solid, dotted or dashed lines are used. Furthermore, it is also possible to apply a combination or doubled lines to emphasize the significance of the separation between adjacent areas [MML10]. An important declaration is the naming of the corresponding type of separator (e.g. line of interaction, line of visibility) which is commonly noted beneath or above the corresponding line. As a result, a basic setup as shown in Figure 2.3 is created.

As a next step, the actions, the corresponding action flow, and the communication flow carried out by the customer and the members of the providing organization, have to be added to the model. At this point, the notation that was suggested within the various literature sources varies to a great degree. In her fundamental article introducing the basic concept of Service Blueprinting, Shostack uses simple quads to display the various activities and solid arrows representing the action and the communication flow [Sho84]. The constellation as shown in Figure 2.4 is probably one of the most basic but, at the same time, comprehensible ways to capture such a process flow. Especially for people with an elementary understanding of modeling issues.

But depending on the intended purpose of the undertaking, a more complex or even a simpler conceptual notation could be applied. One example for the latter one is shown

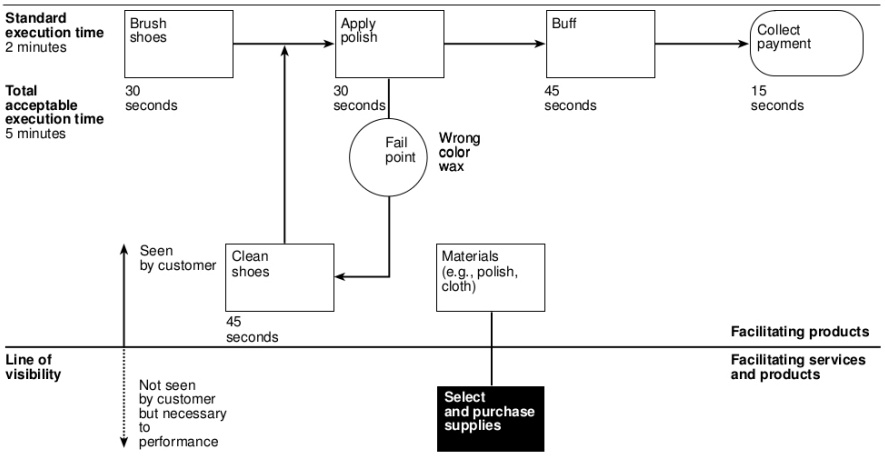


Figure 2.4: Shostack's Service Blueprint for a corner shoeshine [Sho84]

in Figure 2.5 which is an example introduced by Bitner, Ostrom, and Morgan in their corresponding paper [BOM08]. In this model, the action flow within a specific area is not represented explicitly using lines connecting the activities. It is illustrated in a more simple way using only a single dimension of operations which do not inherit a more complex routing like alternative paths or loops. The sequence of actions within an area is derived from the horizontal orientation of the Blueprint. The communication flow, on the other hand, cannot rely on such an implicit way of representation. Since the vertical dimension of a Service Blueprint is used to display the different areas [BOM08], it cannot be used for the illustration of sequential dependencies. For this reason, Bitner et al. used solid arrows to indicate this concept. The direction as shown in the diagram indicates the source and the target of a specific communication process.

Another simple approach that is in a way similar to the last one, is used by Christopher Lovelock and Jochen Wirtz. Within their book, they also rely on the horizontal dimension of the Blueprint to indicate the straightforward sequence of the actions within their corresponding area [LW11]. To indicate the communication that occurs along the interfacing segmentation, filled triangles are applied to the top and/or bottom side of the activities. Again the direction illustrates the source and the target of the flow. This approach is even simpler as the one applied by Bitner et al., since even for this part of the model, traditional arrows are not applied. Only when it comes to the representation of the IT infrastructure, additional lines are added to the diagram. As a result, the model looks very organized and comprehensible. But naturally, this way of displaying a service has some strong limitations and does not work for every purpose. If it is necessary to display more complex sequences or show a more detailed service process, the chosen notation may have to be extended as well. As it is probably with all graphical representations, to a degree there might be a trade-off between the possibilities and the comprehensibility of the created model.

An observation that has been made during the review of the literature is that of course

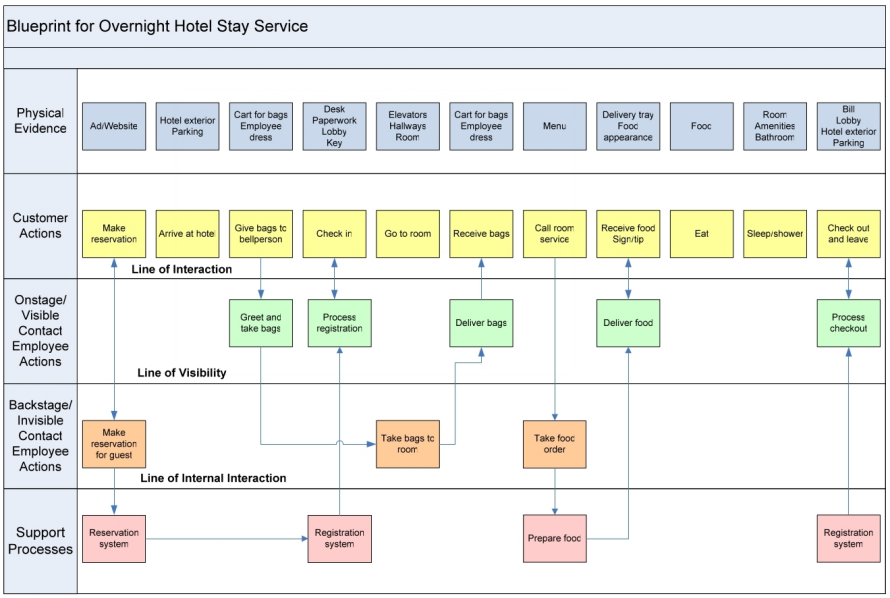


Figure 2.5: Service Blueprint for an overnight hotel stay [BOM08]

depending on the intended purpose of the displayed model, a simpler or more complex notation has been chosen. As an example, again the work of Christopher Lovelock and Jochen Wirtz can be mentioned. Along with their explanations targeting the general concepts of Service Blueprints, they also addressed the topic of self-service technology (SST). During their examinations they illustrated the processes of an e-banking service using this modeling methodology as shown in Figure 2.6 [LW11]. At this point, it can be seen that for the more complex use-case they applied a notation that is similar to a classical flow-chart including start- and endpoints of the process as well as alternative paths (i.e. customer decisions). Additionally, the authors used markers to indicate potential waiting and fail points.

Another example of a more complex approach concerning the notation can be found when looking at the article of Arash Shahin which was published in the year 2010 in the Journal of Management Research [Sha10]. The author examined the method Service Blueprinting to target critical service processes and also included a case study describing a four-star hotel. During his examinations concerning the basic concepts and notational representations, he came up with the suggestions as shown in Figure 2.7. Like other authors, he recommends the use of start and end nodes with the intention to strictly specify the initial and the final actions that are executed during one iteration. Looking at the sequence itself, besides the common quads indicating the activities, he also lists other graphical concepts that can be used within the specific areas. The first one on the list is the *Decision Point* and enables the designer to incorporate alternative paths. Additionally, the *Input/Output* node is used for actions where objects carrying information vital for the service execution are exchanged between the agents. The last one on the

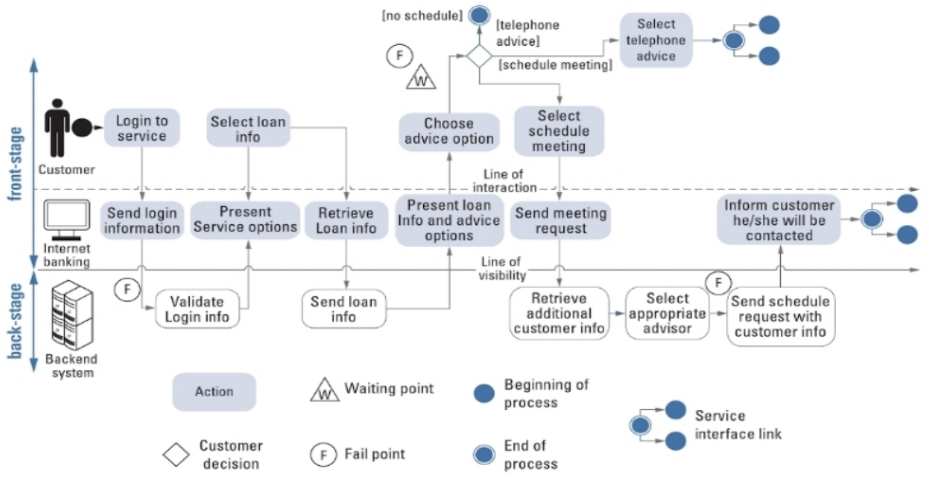


Figure 2.6: Service Blueprint for a self-service internet-delivered banking process [LW11]

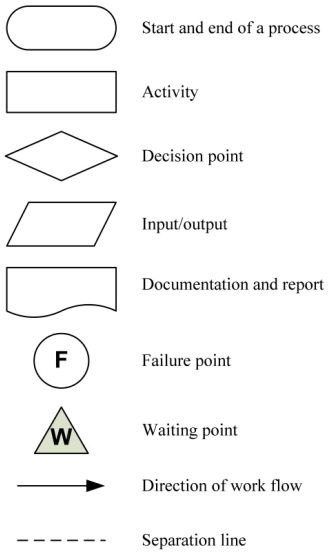


Figure 2.7: Suggestion for the notation of Service Blueprinting by Arash Shahin [Sha10]

list is called *Documentation and Report* and indicates activities that need to be carried out during the process to either capture required data for the organization or pass on information to other instances. In addition to these concepts, once more the use of failure and waiting points are suggested.

Despite the extent and complexity of the chosen notational concepts, every Service Blueprint illustrates the action and communication flow between the various parties of the service execution [Sho84] [KMJ15] [BOM08]. The participants or agents inherit specific roles and functions (e.g. customer, front-desk staff) within the overall process

orchestration. To be able to assign the actions corresponding to their affiliation to the individual actors, the areas as a result of the line separation are used [KMJ15]. Their description is typically indicated on one side of the canvas [BOM08] [Lov14].

Once the service process itself is properly represented by the designed Blueprint, it is time to add the physical pieces of evidence that occur during the execution phase. Looking again at the literature at hand, one can observe that tangible elements are typically displayed as simple quads or similar objects assigned to their own area located on top of the model canvas [Sho84] [BOM08]. When it comes to the more simpler representations of process flows, the pieces of evidence are simply vertically aligned with the actions that define the context of its occurrence [LW11] [KMJ15]. But when it comes down to more complex orchestrations probably the use of additional graphical links using lines could be applied [Sho84].

At this point, the core concepts common within the literature including their notation were discussed. Of course, when the purpose of the modeling project demands additional mechanisms, the basic methodology of Service Blueprinting may be extended as well as their graphical representation. The upcoming chapter shall summarize the potential benefits as well as the shortcomings that might occur while applying this tool-set.

2.3.5 Benefits and Shortcomings

During the previous sections of this chapter, the nature of the modeling technique Service Blueprinting including the most common concepts and corresponding graphical notations was discussed. This next part is dedicated to emphasize the reason why this methodology is used in the first place. To do so the benefits for one's organization while applying the tool-set shall be examined. Additionally, there is also the need to have a look at the shortcomings and weak-points that, after all, limit the applicability of the approach in its basic constellation and may be opportunities for future improvements. These observations shall conclude the section and form the bridge to possible enhancements and combination of the observed concepts.

To begin with, it should be noted that Service Blueprinting is considered as one of the most widely adopted development approaches when it comes to the area of Service Design [KMJ14]. Besides the fact that it is rather simple to apply in its basic versions, one of the most important reasons for its popularity is the strong customer focus of the overall approach [Sho84] [KMJ14] [BOM08]. As already explained, the whole service orchestration is designed and captured while taking the customer's point of view of the value generation. This characteristic encourages the persons involved in the model's creation to abstract from their traditional and organizational point of view and observe the intended process and its context from the end-users perception.

In general Service Blueprints help the organization when it comes to the gathering of initial ideas and the development while adding all necessary activities and operations as well as the linked physical pieces of evidence that are perceived during the execution phase of the service [BOM08]. Afterwards, the results of the design process may be

used as artifacts subject to the documentation of the service. Furthermore, due to the graphical nature of the methodology it also can serve the communication of the ideas and constellation of activities [BY05].

When it comes to already existing services, the methodology can be used to analyze the nature and characteristics of the offering and provide the designers with valuable insights into the processes as they are executed in real life [SDSB09]. Again the customer's point of view is a critical aspect of the model's value. Once a service is put in place and applied over a longer period of time, it is often necessary to perform minor changes and adapt the mechanisms to new circumstances and requirements (e.g. new law concerning documentation, special demands of important customers) [Lov14]. In reality, these amendments are often implemented without reviewing the whole service offering since such an undertaking would result in extensive costs. But this also may promote situations where a once well-designed process inherits characteristics that are not satisfying to the customer anymore and therefore negatively influence the perceived value. To get the undertaking back on track, Service Blueprinting can be applied to map the current situation from the end-users perspective, identify the weak-points and critical sections in general [BOM08].

Using this analysis as a starting point, Blueprinting can also be applied to determine opportunities and therefore help to enhance the service at hand [Lov14]. Depending on the nature of the detected flaws, new concepts, mechanisms, and technologies may be applied to not only even out the negative aspects that were observed, but furthermore, increase the perceived value and create innovative process constellations. Since one of the most important things, when it comes to maintaining the organizations market position, is to come up with unique ways of approaching the underlying topic, foreseeing changes within the customer's demands or at least timely align the chosen methods with these requirements, this characteristic of Service Blueprints surely contributes to its success.

Typically, in comparison to other process-oriented areas, services design itself is not well supported by structured modeling approaches [DRK16]. Due to the significance of services for the success of a modern company, it is important to construct and conserve the planned offering in a more structured and comprehensive way [Sho84]. The fact that Blueprinting is a graphical representation of the concerned processes, makes it suitable for this task. The use of a modeling approach consisting of a flow-chart notation enables the designer to capture the whole execution phase of the service. Exactly this characteristic, along with the fact that this kind of methodology is widely adopted throughout various domains, makes the Blueprint to an important artifact when it comes to the communication of one's intentions. The freedom that is inherited by the approach due to the lack of a fixed standardization contributes to its applicability within more creative contexts.

The next advantage that shall be noted at this point was already addressed within the chapter concerning the design process of a service and goes back to the 5 principles mentioned by Stickdorn [Sti16a]. It describes the fact that service design and therefore also Blueprinting strives towards a more complete illustration of the whole situation. Not

only the knowledge of one domain, but rather of all areas of expertise that are directly involved or at least influence the value generation later on have to be integrated. Once more the property that the customer's point of view is a key aspect of the methodology, contributes to the fulfillment of this demand since after all the end-user's expectations are vital to the success. Of course, to be able to meet this requirement, the knowledge of the various participants and areas have to be gathered and integrated into the model. Service Blueprinting supports this aspect because it allows its design process to be carried out in a collaborative way [Sti16a]. An example would be the application using a set of workshops where all the parties representing the specific groups within the domain have the possibility to influence the final outcome. This way the used methodology also facilitates the creativity of the new service offerings and its design process, which allows Service Blueprinting to be regarded as a creativity method [BOM08].

The last benefit that shall be noted at this point concerns the complexity of the result of the model itself. As stated in the sections discussing the concepts and their graphical notation, the representation of the service offering can be conducted with a varying level of detail. It is possible to use a rather simple illustration for more general purposes like the communication of the underlying ideas towards other parties. On the other hand, the methodology enables the designer to increase the complexity, add additional functionalities and so create a more holistic model. Furthermore, this distinct levels of abstractions also entail the possibility to use the more generalized illustration as a kind of overview and then apply a drill-down to display the sub-processes in more detail [Lov14]. This way the benefits as mentioned above are maintained to a great deal and, at the same time, quite complex service offerings can be created, documented and analyzed.

But of course no modeling approach is perfect and so Service Blueprints also inherit some disadvantages and shortcomings that shall be discussed in the next paragraphs.

First of all, probably one of the most significant negative aspects that is of a particular importance for this work and which was already mentioned on several occasions within the previous sections, is the fact that for Service Blueprinting a standardization does not exist [LW11] [BOM08]. As a result, at the beginning of each design process, there has to be an agreement concerning the applied concepts and their graphical notation. But of course, this influences the comparability and compatibility with Blueprints created by other designers and within a different context.

The nature of the method as explained also induces another disadvantage. Although the tool-set supports the application of varying levels of detail, especially when it comes to the combination of services with very critical processes within an organization, the concepts, and possibilities that can be included in a Blueprint may not be sufficient to illustrate all necessary aspects like special cases, detours, delays, exceptions, or interfaces to other parties [MML10]. After all, to display a process with a high complexity, a more comprehensive approach like BPMN might be required.

Furthermore when looking at the actual creation of the model itself, one can see that in comparison to other techniques like UML, up until now Blueprinting lacks a broad tool

support for a computerized incorporation of services. This concerns the illustration of the process orchestration itself as well as the automatic transformation of the concepts into other compatible languages. To target this observation, within the context of this work, a first step shall be created and a matching towards BPMN, that entails the automatic transformation will be examined and discussed.

Considering the benefits and shortcomings that are mentioned above, over the past years, several authors have suggested possible extensions and combinations with concepts of other modeling areas. The following chapter shall provide a very brief introduction of some of them.

2.3.6 Enhancements and Combinations

Service Blueprinting is a modeling technique that is applicable to services from very different domains. It provides basic concepts and guidelines to create representations of the underlying processes which may inherit a varying degree of abstraction. When considering the more skeptical literature, one of the most significant flaws of this methodology is that the common concepts and notations which are similar to traditional flowcharting approaches may not be sufficient to capture all necessary details and mechanisms[AF15]. This shortcoming in mind, several authors have suggested extensions and combinations with concepts known from other modeling techniques and areas. Besides enhancements that are directly aligned with the classical elements of a Blueprint, like the *Line of Internal IT Interaction* as discussed by Christopher Lovelock and Jochen Wirtz [LW11], more extensive constellations where also addressed. This short section shall list some of the more interesting ones that were encountered during the literature research of this work.

Service Blueprints and Event Driven Process Chains

The first candidate that shall be presented at this point is the combination of Service Blueprints with Event Driven Process Chains (EPC). It was mentioned and examined as a possibility within a paper of Jochen Meis, Philipp Menschner, and Jan Marco Leimeister[MML10].

EPC was developed in 1992 at the University of Saarland under the lead of August-Wilhelm Scheer in cooperation with SAP and contributes to the ARIS concept (Architektur Integrierter Informationssysteme) [ST05]. It is a tool-set designed to enable the designers to illustrate business processes and the corresponding workflows [Com17]. To do so the modeling language uses the two main elements *Activity* and *Event*, which, when connected to a diagram, form an alternate succession [Bau10]. Within a sequence, an *Event* is directly followed by an *Activity* that is performed to respond to the *Event*. Afterwards, as a result, a new *Event* occurs, which is then once more followed by an *Activity*. Such an element can always have at most one incoming and one outgoing arrow, connecting it to the next adjacent object of the opposing nature. To display different flows, logical operators (e.g. AND, OR) can be used to split and join the distinct paths.

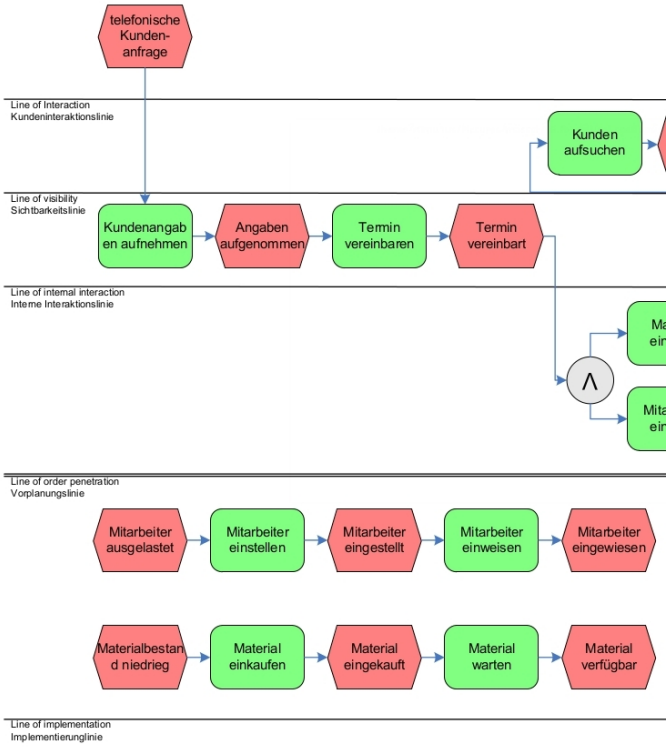


Figure 2.8: Example for the combination of Event Driven Process Chains and Service Blueprinting [MML10]

To overcome the notational vagueness of Service Blueprinting and introduce a comprehensive way to model all sorts of process flows, Jochen Meis, Philipp Menschner and Jan Marco Leimeister investigated the possibility to apply the methodology EPC as a flow-chart notation to illustrate the action sequences in the distinct areas of the Blueprint [MML10]. Within such a diagram as shown in Figure 2.8, the flow of the actions as well as the communication via the interfacing line separators (e.g. *Line of Visibility*, *Line of Interaction*) is captured using the connecting arrows of the Event Driven Process Chain.

As a result of their examinations, the authors came to the conclusion, that the application of this combination supports the ability of Service Blueprints to represent service offerings and their underlying processes. Due to the fixed notation and the strict convention concerning the applicability of its functions, the diagram is less ambiguous and inherits a better comparability. But due to the nature of EPC, that every sequence is displayed while using the alternate succession of its elements, such a model of a real-world scenario tends to get very extensive and confusing. Furthermore, the authors noted the lack of support for displaying computerized processes (in their case support within the area of real estates) [MML10].

As a final remark, they suggested looking for a more suitable process modeling language to combine with Service Blueprinting, which leads to the next constellation that shall be mentioned.

Business Service Blueprint Modeling

The next method within this context is called Business Service Blueprint Modeling (BSBM) and was also suggested by Jochen Meis, Philipp Menschner, and Jan Marco Leimeister [MML10]. It basically combines the methodology Service Blueprinting with the Business Process Model and Notation (BPMN) to create a tool-set that is suitable for the area of modern service design with a strong focus on processes that posses a strong dependency towards the information technology (e.g. self-service technology).

BPMN is probably one of the most widely spread and adopted modeling techniques within the process area. It constitutes a standardized graphical language that is available in its version 2.0 since 2011 [OMG11]. To support designers from various domains, the approach offers a comprehensive catalog of different concepts and the corresponding notations [Ber15]. To illustrate a process, activities or groupings of activities are connected using directed edges representing the sequence flow. Different gateways (e.g. parallel gateway, inclusive gateway), events (e.g. error, messaging, timer), data objects and other elements enable the designer to describe even very complex orchestrations. Furthermore, BPMN provides two grouping concepts called pools and swimlanes which impose specific restrictions and functionalities. Due to the importance of this language for the overall purpose of this work, it will be discussed in more detail in an own chapter later on.

In their paper Jochen Meis, Philipp Menschner and Jan Marco Leimeister discuss the possibilities to combine BPMN and Service Blueprinting. The basic idea is similar to the combination with EPC and so BPMN shall be used as flow-chart notation illustrating the action flow within their associated areas [MML10]. The authors point out three different ways how to use the BPMN grouping mechanisms (i.e. pools and swimlanes) for the representations of the line separation from Service Blueprinting. The first one uses pools to display the areas for the customer and the organization and the subordinated lanes for the specific lines of the Blueprint as shown in Figure 2.9 (e.g. line of internal interaction). The second alternative only relies on the application of pools and every horizontal separator constitutes an own BPMN pool. The last suggested variation is to use only one pool which contains the whole diagram and capture each line using swimlanes. Due to the different functionalities of the used BPMN concepts, each alternative results in a distinct approach towards this topic. After their description, the authors examined the specific characteristics and applied a set of criteria to rate the variants.

As a conclusion, Jochen Meis, Philipp Menschner, and Jan Marco Leimeister stated that probably no alternative is suited for all kinds of services that could be modeled. Only the third approach does not provide any significant benefit for a specific area [MML10].

According to the authors, Business Service Blueprint Modeling inherits the benefits from Service Blueprinting as well as from the Business Process Model and Notation. Further-

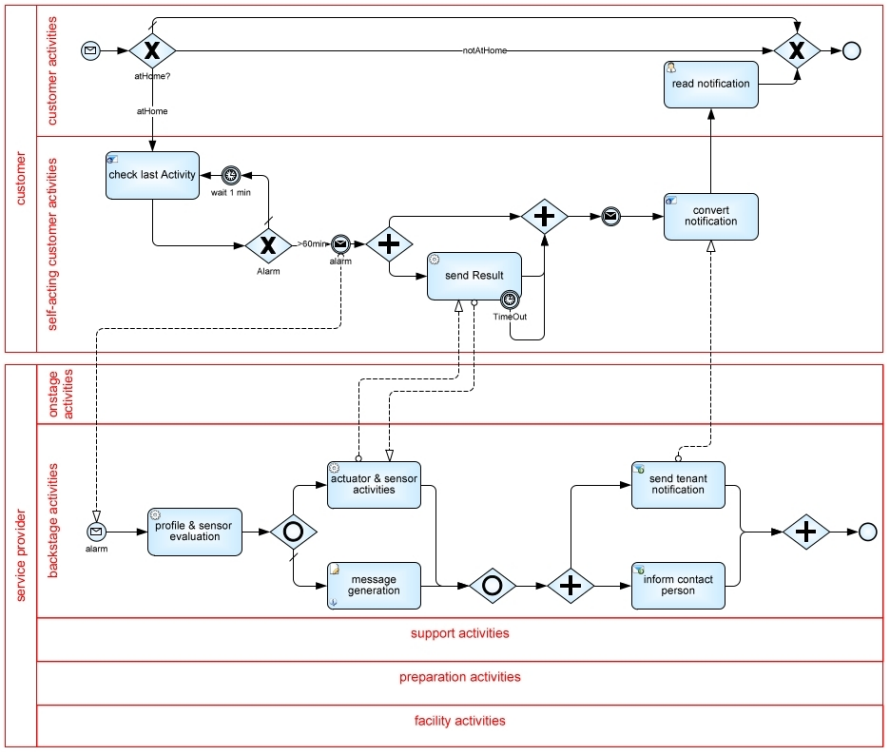


Figure 2.9: Combination of BPMN and Service Blueprinting using one pool for the customer and one for the organization [MML10]

more, due to the capabilities of BPMN, this new modeling approach has the potential to contribute to a possible automatic processing of the modeled services [MML10].

The downside, on the other hand, might be introduced by the complexity of BPMN itself. A big advantage of Service Blueprints is the applicability due to the simple concepts and the enabled straightforward notation. Now if BPMN is used for this flow-chart part, the advanced concepts and the associated restrictions impose the requirement that the designer has to have solid knowledge of this modeling language. But also other stakeholders who should interpret the model at the end, need to have at least a basic understanding of BPMN itself.

Service Blueprints and the Failure Modes and Effects Analysis

Another combination with Service Blueprinting, that is of a different nature than the approaches mentioned before, was suggested by Pao-Tiao Chuang in the year 2007 [Chu07]. His starting point was to target possible errors within offerings of an organization. When it comes to services in general, each orchestration has potential failures and weak-points that influence the customer's experience and so the perceived value. With an increasing number of touch points and interactions with the end-users, the numbers

of errors that might occur and negatively influence the processes are also increasing. This results in a higher complexity of the intended service [KMJ14]. To counteract this observation and maintain and ensure the quality, every organization strives towards the minimization of these negative influences. For this purpose, probably the best approach is to integrate the consideration of critical sections concerning the workflows, interactions and communications already during the initial design steps, but also when it comes to the later phases of the project [Sha10]. As already mentioned during the explanations of the nature of Service Blueprinting, the methodology has the potential to support the creators of a new service to analyze the intended action and communication flows and detect possible weak-points. Pao-Tiao Chuang now intends to use this characteristic and combine it with the concepts of the so-called Failure Modes and Effects Analysis (FMEA) to extend the potential effectiveness of dealing with these issues.

FMEA was originally developed for the suppliers of the US American military. The breakthrough of the methodology can be traced back to the 1960s when the NASA applied it within the context of their Apollo missions. Later on, especially industrial and automotive companies starting with Ford have adopted the concepts to ensure and improve the quality of their work. For this purpose, FMEA proposes mainly two distinct groups of approaches. The first one focuses on the design of the products (e.g. cars) and the second one on the characteristics of the organizational processes. When it comes to services, the second category is more applicable. Looking at the development processes, FMEA tries to address possible failures already at the very beginning of the concepts. It follows the premise that the costs of occurring errors increase rapidly along the lifecycle of the project and so the detection and prevention should be the main priority [BK].

The method itself consists within the application of a form sheet helping to answer questions about where errors might occur, what the extent and the impact could be, what the consequences are and why this could happen. This questionnaire should then be answered by a group of experts [BK]. Furthermore, Pao-Tiao Chuang summarizes the development process using the following points [Chu07]:

1. Identify potential failure modes
2. Evaluate their causes and impacts
3. Prioritize them
4. Develop appropriate counter measures

The combination as intended by the author uses Service Blueprinting to describe the intended service and its underlying processes. This way the potential fail- and weak-points shall be identified and brought to the awareness of the design team. Within a next step, the Failure Modes and Effect Analysis is applied to evaluate these critical sections and develop suitable measurements for the prevention and correction of errors [Chu07].

So in contrast to the combinations with the Event Driven Process Chains or the Business Process Model and Notation as explained before, this concept is of a more additive nature

and does not have an impact on the creation and design of the Blueprint itself. Therefore an additional combination with an integrative approach is still possible.

Gantt Charts for Service Blueprinting

Another very interesting approach using a combination of another modeling technique was proposed by Sabine Fließ, Britta Lasshof, and Monika Meckel at the University of Hagen [FLM04]. Their starting point was to focus on time as a critical factor for service processes and to develop a representation that incorporates time as a quantity into the displayed concepts. As a result of their paper, they suggested the application of Gantt charts as notation illustrating the sequence flows within the corresponding areas of the Service Blueprints.

Gantt diagrams are a very popular method to model time oriented sequences of activities. They are often applied when it comes to project management or other organizational planning and controlling tasks. When looking at the modeling canvas, typically the different procedures that consume time during their execution phases are located on the vertical dimension. The horizontal axis represents the quantified time itself. For each activity, a quad is placed on the canvas. Its vertical positioning is, of course, depending on its affiliation with the process context. It is common that operations that are executed initially are also displayed first (i.e. in the upper left corner) within the diagram which so imposes a sorting. The horizontal alignment is following the point of time when the activity starts and the length of the graphical element represents the duration of its processing. The resulting subsequent illustration is also applied to display the dependencies between the specific tasks and their inherent duration [Wil02].

Sabine Fließ, Britta Lasshof, and Monika Meckel used the concepts as described above and integrated them into a Service Blueprint [FLM04]. Within this context, the time-consuming tasks that are part of the Gantt chart represent the necessary activities for the service and are located within the corresponding areas of the Blueprint. Again the horizontal dimension represents the time aspect of the model and now can be used to explicitly display the duration of the actions. The sequence and communication flow of the model are displayed using arrows as it is common for this kind of model. Within their conclusion, the authors note that the applicability of this approach strongly depends on the nature of the service and the purpose of the Blueprint.

Probably the main advantage of this combination lies within the fact that a critical and therefore very important aspect of this overall topic, the time, is explicitly displayed within the diagram. It is not only possible to illustrate the flows and the dependencies between the distinct activities but also to include the required duration of the tasks. A possible downside lies within the fact, that to be able to represent quite complex and comprehensive processes, specific functions like alternative paths, loops or exceptions need to be included. The application of a fixed timeline on the horizontal axis might not be compatible with these requirements [MML10].

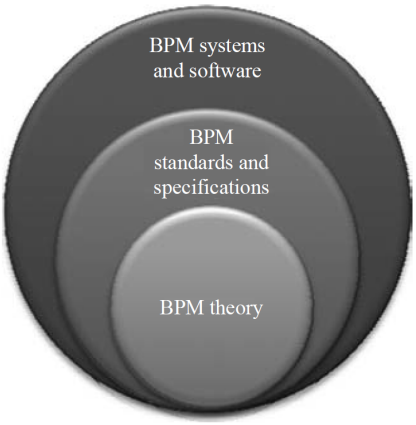


Figure 2.10: Correlation between the aspects of Business Process Management [KLL09]

2.4 Business Process Model and Notation

This next chapter is dedicated to the other side of the intended transformation that shall be introduced and discussed in the context of this work. The Business Process Model and Notation (BPMN) is a well known and widely adopted modeling approach when it comes to the creation and design of processes. It offers a comprehensive catalog of concepts that can be used to illustrate very complex structures and workflows. To approach this methodology, first of all, the general topic of Business Process Management is discussed which should provide some insights into the motivation of BPMN. Afterwards, the historical background, the concepts, the used notation as well as the benefits and shortcomings will be addressed.

2.4.1 Business Process Management

To grasp the definition and nature of the modeling language BPMN, it might be a good idea to briefly address the field that can be considered as superordinated. It is called Business Process Management[1] and according to the Gabler dictionary of economics, it targets the creation and design of business processes with a special focus on efficiency and its integration into the organization and its culture [Spr14]. It is not a specific technology on its own, but rather a whole management discipline [HPN08].

The necessity for the organizations to focus their efforts on their business processes with the intent to enhance the performance and value within the context of the overall goals is a result of the global economic development during the more recent history [KLL09]. The fast-changing nature of the customer's demands, the integration of new technology into the process chains and especially the participation in a global market with many potential competitors, are only some of the possible reasons. The breakthrough of the information

[1] In the literature often abbreviated as BPM. But due to possible confusions with Business Process Modeling, it is not used within the context of this work.

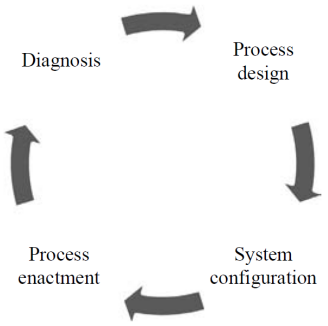


Figure 2.11: The Business Process Management Lifecycle [KLL09]

technology has enabled the development of various methods and approaches to target these new requirements. The results are often summarized with the term Business Process Management Systems. Figure 2.10 represents an illustration by Ko et al. targeting the correlation between Business Process Management theory, the corresponding standards and specifications as well as the systems [KLL09].

During their introductory paragraphs, to define the overall field, the authors refer to the Business Process Management Lifecycle which includes the steps *Process design*, *System configuration*, *Process enactment* and *Diagnosis* [KLL09] [vdAtHW03] as shown in Figure 2.11. So the discipline starts with the design of the intended processes, using mostly graphical standards, but also the implementation and analysis of the underlying systems.

In accordance with these phases, a categorization of the standards that can be found throughout the literature was suggested [KLL09].

- Graphical standards
- Execution standards
- Interchange standards
- Diagnosis standards

Of course, the first aspect is especially important for the context of this work, since it describes methodologies to capture, design and evaluate processes within an organization and even across its borders. Along with other standards like the Unified Modeling Language (UML), the Business Process Model and Notation (BPMN) is also associated to this group an is considered to be one of the most expressive and applicable ones [KLL09].

2.4.2 Historical Remarks

Within the following paragraphs, a short summary concerning the historical origin and the developments of BPMN shall be given.

The need for a structured approach towards the design of the organizational processes as a result of the economic changes that occurred during the last decades was emphasized already within the previous chapter. Along with demands targeting the comprehensiveness and compatibility of the intended approach, another important aspect is considered to be the possibility to apply it to the communication of the concepts and ideas between the various stakeholders that are directly or indirectly connected to the process [MJ12].

To address these new requirements, already existing modeling techniques, as well as methodologies specifically created and dedicated to the area of business processes, were applied. For many of those their origin that lies within the information technology proved to be an issue when it comes to the integration into a more cross-functional environment. Well known and widely adopted languages like UML (more precisely UML activity diagrams) could not establish themselves because they were considered to be too strongly related to the IT area and not comprehensible for business users of other domains [Flo15].

To meet the expectations of the various participants of the design process and still be able to fulfill the demands imposed by the problems that were highlighted before, in the year 2004 the Business Process Management Initiative used different aspects and concepts of several modeling approaches and introduced the Business Process Modeling Notation. In 2005 this organization was merged with the Object Management Group (e.g. also managing UML) which since then is responsible for the maintenance and development of the standard [BPM12].

Over the years the methodology was very well adopted within the academic area and by various companies (e.g. IBM and SAP) [MJ12]. Many authors reviewed the proposed concepts and the possibilities of their application. But of course no approach is perfect and so several shortcomings and disadvantages were highlighted along the way. As a countermeasure, the second version of BPMN was released in 2011 by the Object Management Group. Among various other improvements, also the name was changed to Business Process Model and Notation [OMG11].

BPMN 2.0 was created with the intent to enhance the methodology's ability to be interpreted by the various stakeholders. Furthermore, the new version should support formal execution semantics out of the box and provide a standardized exchange format for its models. Although new and more comprehensive functionalities were added to the tool-set, models that were created while using the first version are still compatible. Among others, the possibility to illustrate processes that stretch over a network of multiple, distinct organizations and completely new ways to display task orchestrations were added [Whi12]. Especially this last addition was meant to cover the requirements that are a result of modern business relationships and organizational constellations.

This second version of BPMN including the proposed enhancements and extensions targeting the design of the process flows have to be considered when looking at the transformation from Service Blueprints to BPMN. To have a more detailed understanding of the available concepts, the following chapter shall discuss some of the most important

ones.

2.4.3 The Concept and Graphical Notation

Similar to the topic of Service Blueprinting, it is also necessary to consider the concepts and possibilities that are proposed within the context of the Business Process Model and Notation. Probably one of the major distinctions between the two methodologies is the fact that, in contrast to the service-oriented language, BPMN constitutes a fixed, standardized tool-set. This means that potential enhancements and additional mechanisms are either only relevant for the specific circumstances of its application (e.g. within an organization) or need to be approved, integrated and released by the governing organization, the Object Management Group. Because of this characteristic of BPMN its concepts and the corresponding notations will not be discussed within separate sections, but in direct correlation in the following paragraphs. Since the standardization specifies a strict graphical counterpart that illustrates a certain mechanism or aspect of the modeling language, a separated examination should not be necessary.

To do so, after a short overview of the general aspects and nature of BPMN, the used concepts shall be explained in more detail. At this point, it should be noted that due to the extent of the methodology, this chapter will focus on elements, mechanisms, and functionalities of the language that are considered as being relevant for the intended transformation of this work. The Business Process Model and Notation within its current version 2.0 that was released in 2011 will serve as a foundation for this purpose.

The basic idea of the modeling language can be traced back to the token flows introduced within the context of Petri Nets [Hav05]. For BPMN, these were of course enhanced to a great degree and embedded within a very comprehensive catalog of elements and mechanisms. The core of BPMN is constituted by the techniques targeting the illustration of processes inheriting a varying degree of complexity. As already mentioned before, with version 2.0, it is also possible to design orchestrations including multiple organizations. The elements embedded within the resulting diagrams reach from simple tasks executed by human or mechanical actors to digital artifacts. Also a multitude of other functionalities that allow the creation of complex process flows are included within the standard. Another aspect that was introduced in 2011 as a member of the BPMN family is the new diagram type Choreographies. These provide a different notational approach than the Collaboration Diagrams and strongly focus on the sequence of interactions [Whi12]. Within the literature, also a distinction between a core set of functionalities and an extended one has been made [KMJ14]. While the first one group's elements and mechanisms that should allow the methodology to be applied by regular users and business analysts, the second one has its focus on the applicability within a more technical area that requires a greater level of complexity. Another very important aspect of BPMN, that was already a part of the earlier releases of the modeling language is its ability to be transformed towards execution standards like the Business Process Execution Language (BPEL) [Whi05]. The fundamental intention is to provide an automatic generation of other structured representations of the designed processes. Although the compatibility

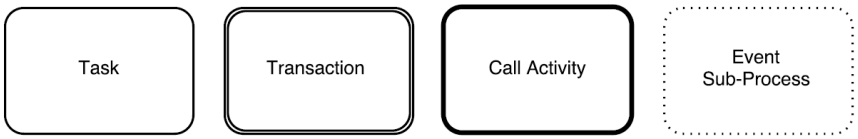


Figure 2.12: BPMN Activities [Dra]

with BPEL has been criticized in the past [ODA+09], with BPMN 2.0 this aspect was also reviewed and improved. As a technical enhancement, each model is stored within a well-specified XML structure that enables the processing after its initial creation.

The next paragraphs will discuss the elements and mechanisms of BPMN Collaboration Diagrams in more detail. This type of illustration comprises the counterpart of Service Blueprints when it comes to the transformation intended by this work.

To begin with, for the purpose of designing complex processes, BPMN offers the superordinated components *Activities*, *Events*, *Gateways*, *Data Objects*, *Swimlanes* and *Pools*. These elements and their corresponding subcategories (if present) are combined and connected to quite complex process structures [Ber15].

Activities

As its name implies, the first group *Activities* contains elements that describe the execution of specific tasks (e.g. call the restaurant) that are displayed while using quads with rounded corners. BPMN 2.0 specifies the subcategories *Tasks*, *Transactions*, *Event Sub-Processes* and *Call Activities*. These elements inherit different functionalities and are graphically distinguished using their type of border as shown in Figure 2.12. The *Task* for example, is probably one of the most basic and at the same time important concepts. In its simple form, it depicts a job (i.e. a unit of work) that is executed when the corresponding path of the process flow is selected [Ber15] [OMG11].

Each *Activity* element can be augmented while using a so-called *Activity Marker*. It describes a set of six different graphical symbols that are placed within the bottom of an activity and indicate a changed behavior of the corresponding element. It is possible to apply multiple *Activity Markers* at the same time. An example would be the *Sub-Process Marker* that indicates that the element is an aggregated representation of a more detailed process flow (i.e. consisting of several smaller steps) and may be illustrated using a drill-down approach. Another example is the *Loop Marker* that adds a recurring characteristic to the *Activity* (i.e. it may be triggered more than once) [OMG11].

Similar to the previous concept, it is possible to indicate so-called *Task Types*. Their notation consists of a set of small symbols that are positioned within the upper left corner of an element. In contrast to the *Activity Marker*, these concepts do not change the execution characteristics of the element but add some information about its nature. An example could be the declaration as an *User Task*, specifying that the job is executed by

Figure 2.13: Example for a compensatory BPMN Service Task [Dra]

a human user, or a *Send Task* which says that at this point a message of some kind is transmitted [OMG11]. Figure 2.13 illustrates a *Task* element that is marked as being a compensation (marker at the bottom) for the case that the reservation at a restaurant has to be canceled. Furthermore, it is indicated in the upper left corner that it is designed as a *Service Task* which means that a kind of automatic process is used to perform this task (e.g. cancellation via computerized workflow).

Events

The next major concept category that shall be explained at this point are *Events*. In contrast to the *Activities* as described above, an element of this group is not something that is performed, but rather something that occurs during the execution of a process. As stated in the official documentation of the modeling language, such elements are triggered by a certain aspect or have an impact on the subsequent flow and impose a reaction [OMG11]. Typical representatives of this category are *Start* and *End Events*. But BPMN in its version 2.0 also suggest more complex elements like *Timers*, *Messageing Events*, *Compensation Events*, *Error Events* and more (for a complete list please refer to the official documentation or the summarizing BPMN Poster [Ber15]).

Furthermore, BPMN differentiates these elements according to their position within the overall diagram. Accordingly, an *Event* can occur at the beginning (i.e. start), at the end or during the process flow (i.e. intermediate). Graphically, these concepts are represented as circles that may contain a symbol indication their specific category (e.g. Message, Timer, Error). Simple *Start* and *End Events* are illustrated as an empty circle. The positioning within the diagram (i.e. start, intermediate, or end) is marked using different line types for the border of the circle as shown in Figure 2.14 and the coloring of the contained symbol specifies if it is of a catching or throwing nature [OMG11].

Not every event category can be applied to every chronological position within the diagram. So, for example, it is not possible to have a *Cancel Event* at the very beginning of the process flow. This fact is a result of the functional differences between these three groups. *Start Events* constitute the first step within the diagram and do not have incoming sequence flows. The overall process is only executed if this event is triggered. The complete opposite, of course, are the *End Events*. They are meant to conclude the flow and do not have outgoing connections towards further diagram parts. *Intermediate Events* on the other hand do not indicate the start or end of the process and should, therefore, have an incoming and outgoing sequence flow. But due to the general nature of

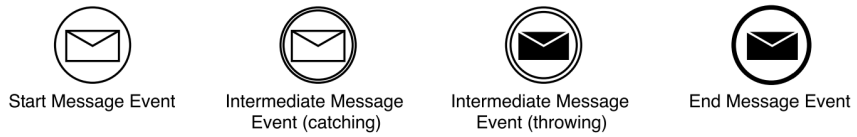


Figure 2.14: BPMN Message Events [Dra]

an event, also this type inherits the ability to influence the course of actions. BPMN 2.0 further specifies the concepts of *Start* and *Intermediate Events* according to their semantic nature and introduces several sub-elements. For example, there is a distinction between an intermediate event that catches a message an one that transmits it (i.e. throwing event). A complete list of the available elements including detailed descriptions about their nature and applicability can be found within the official documentation [OMG11].

As a final remark targeting the concepts described above it shall be noted, that some specific *Events* (e.g. *Message*, *Error*) may not be placed directly on the modeling canvas, but on the border of an *Activity*. This enables the designer to specify if the *Event* may occur during the execution of the corresponding *Activity* or afterwards [Gre13].

Gateways

Taking the elements described above and connecting them with a corresponding flow notation would already result in a representation of a valid process. But due to missing control objects that enable the illustration of more complex sequences like the parallel execution or alternative paths, it would be a rather simple one. Of course, BPMN provides quite comprehensive possibilities to capture even very enhanced process flows.

Within this context, these elements are called *Gateways* and further propagate the token flow mechanisms known from Petri Nets. Similar to this traditional concept, there is the possibility to split, merge and redirect tokens to introduce more variability to the designed processes. BPMN distinguishes seven different types of *Gateways* where each of them is displayed as a diamond-shaped object as known from other modeling languages. The specific type is once more indicated while placing certain symbols at the center of the shape as shown in Figure 2.15 [OMG11].

The first one, the *Exclusive Gateway* is rather straightforward and constitutes an alternative path. When a token that is directed along the process flow, arrives at this point, only one direction can be taken. So no duplication of the token or parallel execution is performed. Each outgoing path of the *Exclusive Gateway* is annotated using a distinct condition. Optionally it is possible to indicate a default path (specific graphical flow notation) that is selected if none of the other conditions is fulfilled [OMG11].

In contrast to the previous one, the *Inclusive Gateway* is treated functionally similar to the logical disjunction. This means that now for each path condition that is true, the token is duplicated and passed on. As a result, it is possible that only one, multiple

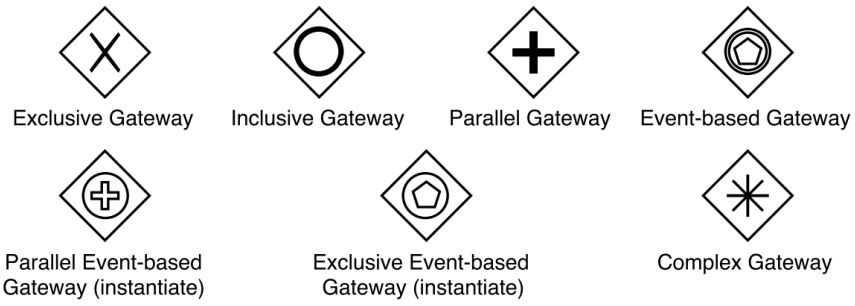


Figure 2.15: BPMN Gateways [Dra]

or all subsequent flow alternatives are executed. An optional default path will only be triggered if no other condition is true [OMG11].

The last member of the rather traditional routing mechanisms is the *Parallel Gateway*. It corresponds to the logical conjunction and describes a parallel execution of all subsequent, connected process flows. To do so, the corresponding token that enters the *Gateway* is duplicated and passed on. If the element is used to synchronize and therefore merge multiple incoming paths, it waits until a token arrives on each of them. Only if this is the case, a single token is passed on to the further sequence flow [OMG11].

In addition to the basic catalog that was presented within the previous lines, BPMN offers some more complex routing mechanisms that enable the designer to illustrate a multitude of situations. The first that shall be mentioned at this point is called *Event-Based Gateway* and provides a quite different functionality. It is a point of separation where the selection of the outgoing path is not performed by checking any assigned conditions but rather by the occurrence of specific events. Therefore each directly subsequent element of the *Gateway* has to be of an event type. Now if a token arrives at the *Event-Based Gateway* it waits until one of the following events occurs. If this is the case, the token is passed on without any duplication. This means that the remaining paths and events are no longer valid. Another version of this routing mechanism is called *Parallel Event-Based Gateway* and prevents this characteristic. It enables multiple events to be triggered and for each one, a new token is passed on. The assumption, in this case, is that during the overall execution of the process, all these events will occur. These *Gateways* can also be used to instantiate a process [OMG11].

The last element in this category is the *Complex Gateway*. It is used to address requirements concerning routing functionalities that are not covered by the other elements [Ber15] and enables the application of a very enhanced synchronization mechanism. To do so the details and circumstances regulating the incoming paths are characterized using an expression. For the control of the outgoing flow of tokens, conditions are annotated resulting in a behavior similar to the *Inclusive Gateway* [OMG11].

Figure 2.16: BPMN Pool including Swimlanes [Dra]

Pools and Swimlanes

The next concept that shall be discussed in this chapter are the grouping mechanisms *Pools* and *Swimlanes*. Again these concepts are very well known from the domain of other modeling languages and inherit a kind of similar functionality.

In general, the need to have such organizational elements arises out of the situation that for real-life processes, it is typically the case that there are multiple participants involved. These may be whole organizations (e.g. companies), single departments, human agents or automatic entities (e.g. systems). Additionally, the designer could introduce a grouping according to functional affiliations between the individual tasks and so encapsulate elements with a more theoretical coherence [OMG11].

For this purpose, BPMN offers the use of so-called *Pools* and *Swimlanes*. Considering the dependencies, the first one can be seen as the superordinated element that may contain a segmentation. When it comes to the graphical notation, *Pools* are displayed as quads that are potentially partitioned using separating lines (i.e. *Swimlanes*) as shown in Figure 2.16. The name of each level is typically annotated on the left side [OMG11].

The way how the designer applies these two concepts within the diagram to correspond to the purpose is basically up to him- or herself. Furthermore, this may depend on the specific use-case, the intention of the undertaking and the focus of the model (e.g. general illustration of the workflow or detailed sequence of the single tasks for automation). So correlating on the situation, one and the same process may be approached in very different ways.

At this point, it is important to note that despite their graphical commonalities, *Pools* and *Swimlanes* inherit different functional characteristics and therefore influence the overall process in different ways. Looking at the superordinated element, it is basically meant to represent a specific participant of the model [OMG11]. Therefore it encapsulates a whole process including its *Start* and *End Events*. This means that there is no sequence flow that passes the border of a *Pool*. But of course, there needs to be the possibility for the various parties to interact with each other. At this point BPMN offers the so-called message flow, that is intended to symbolize the exchange of information between these

actors. It is displayed as a dashed arrow and is either connected to the border of the grouping or to a flow object (e.g. sending task) contained within. BPMN does not support the flow of messages between elements of the same *Pool*. Furthermore, it is not necessary for the grouping to contain any process modeling elements. In this case, the corresponding participant is illustrated as a black box where only the exchanged information can be observed or is relevant for the intention of the BPMN model [OMG11].

Swimlanes on the other hand, traditionally do not represent whole parties but rather functional coherent process elements. So it is possible to capture for example all operations that are executed by a specific department within an organization. In contrast to *Pools*, this concept does not encapsulate a whole process but merely a part of it. Therefore the sequence flow may expand over multiple *Swimlanes* [OMG11].

When it comes to the modeling process itself, it is up to the designer to decide which aspect should be considered as an own participant and therefore illustrated as a *Pool* and which not. So for the example of an organization, it is possible to have each department as an own entity and only use the message flow to link them together or decide to have them within separate *Swimlanes*. Again, the chosen constellation should correspond to the intent of the overall undertaking and the focus of the model itself [OMG11].

Data Objects

As already mentioned in the introductory paragraphs targeting the modeling language BMPN, one of its initial intentions was to create an approach that enables the cooperation between business and technical users. To meet this requirement and keep the aspects of modern organizations in mind, the language had also to address the characteristics of computer aided processes.

One concept that is meant to approach this issue is the application of so-called *Data Objects*. Within this context the standard supports the use of a single object, a *Collection Data Objects*, a *Data Input* and a *Data Output* as shown in Figure 2.17. Furthermore, it is also possible to illustrate the occurrence of complete *Data Stores*. These elements can be connected to the process tasks via *Data Associations* which are displayed using dotted arrows [OMG11].

Considering the complete tool-set of the Business Process Model and Notation in its version 2.0, the explanations provided above only cover the general aspects of the language. It is easily possible to provide a more detailed discussion and extend it over several hundreds of pages. When looking at the official documentation of this standard which has 538 pages, this was probably already done to a great degree. But for the basic understanding of the concepts that are necessary for the intended transformation of Service Blueprints, the explanations as provided above should suffice.

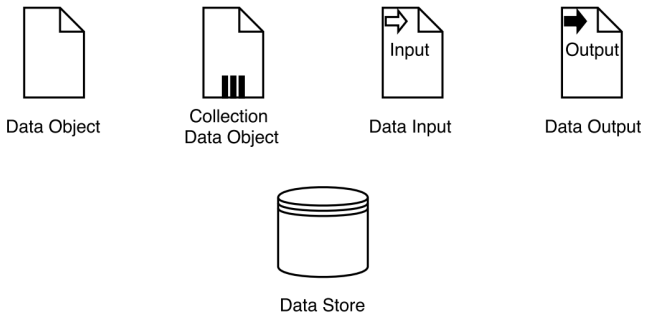


Figure 2.17: BPMN Data Elements [Dra]

2.4.4 Benefits and Shortcomings

The previous chapter had its focus on the concepts and their graphical representations that are suggested by the official standard. All these elements and approaches enable the designer to create quite complex and comprehensive process models. The Business Process Model and Notation itself is very well known and adopted by the economy as well as the academic area. But there is still the question, why this is the case in the first place. Looking at the characteristics of the modeling language, the benefits and the shortcomings that can be observed might offer some possible explanations. Furthermore, when considering the intention of this thesis, it is important to be aware of the positive and negative aspects of BPMN to be able to grasp its role in the transformation later on. Therefore the upcoming paragraphs shall emphasize and discuss these advantages and disadvantages.

The first point that shall be mentioned is rather obvious and is a result of the popularity of the language itself. The motivation to use BPMN for ones process modeling tasks instead of another approach, is quite high also simply because of the fact that many people and organizations are already using it [Pol13]. This makes it easier to find information how to apply the given concepts and capture certain real-world situations. Furthermore when talking to other parties or members of the own organization, the possibility that the created model can be interpreted by the counterpart is considerably high.

In general, the ability of a BPMN model to be used as a medium to transmit ideas and concepts is an important aspect of this language [KLL09]. Although quite profound knowledge about the inherent functionalities and mechanisms is required to capture a complex real-world process in more detail, it is also possible to apply only a smaller subset of the provided tool-set to create an abstract view and so communicate the overall idea [Jua16]. These simpler models might be far from complete but depending on the case at hand and the purpose of the created model, it could be sufficient to graphically underpin a planned undertaking. Additionally, for the reader, it might even be sufficient to only have a basic understanding of flow-charting in general and not BPMN as a specific case.

But these last arguments for the use of BPMN would not be that relevant for this discussion without considering the circumstances of the development and enhancement of this language. As already mentioned a couple of times, BPMN is published and maintained as a standardized modeling approach. This means that it is controlled via a centralized organization, the Object Management Group [OMG11]. Therefore, if the given constraints towards the used concepts and their graphical notations are not violated during the design process of a specific model, it can be passed on to other parties that have knowledge about these concepts on their own, without the need for additional explanations of this case. Without this unified approach, it would not be possible to create comprehensive models of real-world processes that can be interpreted by the respective community.

Another benefit that can be observed is related to the circumstances how the process models can be created. Over the time a respectable number of software developers have started producing their own tools that enable the computer-aided design of BPMN [And12] [Wik18]. Up until now these applications do not only provide the graphical modeling capabilities, but also support the functionalities and the imposed restrictions that are described in the official documentation. Mechanisms like *Cawemo* which enable a collaborative development process were also introduced [Cam17].

One interesting aspect that is linked to the point mentioned in the last paragraph has emerged with the BPMN version 2.0 published in 2011. Since then the modeling language proposes a standardized XML meta structure which describes the models created by the users [OMG11]. This means that, if supported by the chosen tool, every model that was designed using the software generates automatically the corresponding XML structure. It can then be used to import the model into another application or to process it in other ways (e.g. automation, transformation). This aspect also highly correlates with the ability of BPMN to positively influence the communication between several parties.

In general, it is probably safe to say that BPMN is a powerful modeling approach for processes. Especial when looking at the catalog of the version 2.0 one can find a very comprehensive set of concepts that can be used to design complex workflows and constellations. This also induces the versatility of the language which makes it possible to apply it on various domains and areas. A characteristic that is especially interesting at this point is the support of computer-aided processes. As discussed in the chapter covering the concepts and notations, the designer has the possibility to connect data-oriented objects including their flow directly to the diagram [OMG11]. Furthermore, it is possible to indicate if a given task is executed by a human, manually or automatically via specific service.

The last aspect that shall be mentioned at this point concerns the ability to reuse a given model and to transform it towards another language, whether it also belongs to a modeling domain or serves another purpose (e.g. the Business Process Execution Language (BPEL)) [Whi05] [YS10]. Already before version 2.0, this was an intention of BPMN. Since 2011 however, due to the unified background structure in XML, the next step towards this possibility has been made. At this point it should be highlighted, that

such a transformation or automation is not without challenges or problems [ODA+09]. But within this area, BPMN provides a solid foundation for such undertakings.

But of course, this modeling approach also inherits some shortcomings that have a negative impact on its applicability.

As mentioned before, BPMN provides a very comprehensive catalog of concepts which are captured and described in its official documentation that is 538 pages long. Under normal circumstances, it is not necessary to internalize every aspect that is stated in this manual. But nevertheless, the extent of this tool-set and the inherent possibilities may induce some barriers to begin with [Flo15]. This means that to be able to use this methodology and apply it under consideration of the imposed restrictions and functionalities of its elements, profound knowledge concerning the language is necessary.

Furthermore, this aspect also results in the criticism that a correct and, to a degree, complete BPMN model of a real-world situation can get extremely complex and confusing very fast. Of course, this is especially the case since its version 2.0.

The next aspect that was highlighted in the literature concerning this topic, is the ability of the users (i.e. the designers) to apply the concepts and methods without the support of a corresponding software. Due to the precise definition of the graphical notation and the fact that each tiny symbol and even the filling of the shapes and the line type of its borders have very specific meanings and functionalities, it is very hard and inconvenient to use the language to create diagrams by hand. If such an approach is preferred, another tool-set like UML activity diagrams may be more fitting for the task [WvdAD+06].

In addition to this criticism, it was also emphasized, that this very comprehensive but strict definition of its elements may also negatively influence the freedom of the designer [KLL09]. BPMN itself is not that flexible when it comes to the application of its elements targeting aspects and situations in a way that it was not intended for. In combination with the lack of usability regarding modeling approaches without software support, BPMN is probably not suitable for cases that require a certain degree of creativity.

The last remark concerning the shortcomings within this context targets the transformation and automation capabilities of BPMN models that was also mentioned before. In fact, this is not so much a disadvantage but nevertheless may impose an issue when using this modeling language for such an undertaking. Despite the theoretic possibilities that can be achieved especially while utilizing the standardized XML background structure of version 2.0, in contrast to the expectations that one might get while reviewing the suggestions made within the corresponding literature, such a processing of BPMN models can be ambiguous and quite complex [ODA+09]. The respective developers have to consider a multitude of characteristics and aspects to be able to create a transformation that supports and correctly interprets the very comprehensive set of BPMN concepts.

2.5 Comparison of Service Blueprints and BPMN

This section is dedicated to the comparison of the modeling languages Service Blueprinting and the Business Process Model and Notation. As they constitute the two sides during the intended transformation, their concepts need to be considered in context to each other with a focus on similarities and aspects that may impose a gap when it comes to their applicability. The findings and insights that will be gathered in the upcoming paragraphs shall constitute the starting point of the planed matching.

2.5.1 The Comparison within the Literature

During the literature research at the beginning of this thesis, especially two publications of Lester W. Johnson and Simon K. Milton, respectively Yahya Kazemzadeh, Simon K. Milton, and Lester W. Johnson were found to be very interesting and helpful regarding this topic [MJ12] [KMJ15]. Both articles focus on the comparison of the mentioned modeling approaches with the intent to highlight conceptual overlappings and the identification of elements that do not have a comparable counterpart on the other side.

As a reason why the authors took on this topic in the first place, they state that the design of business processes inherits a very high importance in the daily business. Looking at this area, BPMN and Service Blueprinting are very popular modeling techniques. As also described in previous chapters, both of them provide different concepts and notations which are a result of the diverse demands of their respective field of application. While BPMN has its focus on the design of detailed process orchestrations from the organization's point of view, as its name implies, Service Blueprinting has the intention to illustrate a service offering emphasizing the customer's experience during the execution phase. Now, of course, due to its nature, services themselves are basically a composition of at least one process and therefore may also be represented using BPMN. In fact, because of its increasing importance for the success of a company which is accompanied by a high complexity, it is also necessary to take the detailed sequence flows under consideration. As a conclusion, the authors argue that to be able to apply both modeling approaches during the development of one's service offerings, not only an understanding of each methodology on its own has to be present, but also the knowledge about the correlation between those two [KMJ15].

Now to take care of this demand, within their article *Service Blueprinting and Business Process Modeling Notation (BPMN): A Conceptual Comparison* written by Yahya Kazemzadeh, Simon K. Milton, and Lester W. Johnson, they suggest and apply the method of conceptual evaluation to compare each set of concepts introduced by the modeling languages [KMJ15]. Since they do not assume a specific direction for a transformation as it is the case for this work, for the matching two major issues need to be considered. At first, the authors address the question of which concepts of Service Blueprinting can be semantically reflected while using BPMN. Afterward the same is done for the other way around. It can be compared to the usage of a left respectively right outer join in SQL, where always one side of the linkage is represented with its

full data set and for the other one only those lines are shown that find a corresponding match [KMJ15].

In detail Kazemzadeh, Milton and Johnson list 5 steps that need to be executed to be able to come to a conclusion [KMJ15].

1. Examine concepts of Service Blueprinting
2. Examine concepts of the Business Process Model and Notation
3. Match the concepts of Service Blueprinting against BPMN
4. Match the concepts of BPMN against Service Blueprinting
5. Summarize results and derive insights

The first two points form the necessary starting point and concentrate on the assessment of the individual concepts available within each modeling approach separately. As a result, a fundamental understanding about the given tool-sets as well as their semantics and applicability should be acquired. Of course, this is a requisite for the upcoming comparisons [KMJ15].

Afterward, as explained above, in point three and four each side is matched with the other one. To do so, for example in step three, for each semantic element that is part of a predefined version of Service Blueprinting (necessary due to the missing standardization) a possible counterpart shall be identified. This can lead to three different results. First of all, it is possible that a concept cannot be represented by the other modeling language. If this is the case, a gap has already been detected. The best case, on the other hand, is that a corresponding element which is semantically aligned is also present on the opposite side of the equation. If so, it can be used to illustrate the same situations. But of course it is not everything black or white and so a third category may be needed. It applies to cases where a specific concept is only partially covered by the other modeling approach. At this point, it shall be noted that for a full or partial coverage of a concept it is not necessary that this is done by one specific semantic element of the counterpart. There is also the possibility that this is true for a combination of several distinct ones. Within this context, it is important that during the examinations concerning a potential match, it is not enough to determine if such a counterpart can be found, but also to which degree it is corresponding to the target. However if it comes to a situation that an aspect can be covered with multiple different options, the one that requires a smaller set of components should be preferred [KMJ15].

After the compassion of both sides is completed, the final step consists of the collection and representation of the discovered results. At this point, not only the insights are presented which are a direct result of the concept wise examinations, but also logical inferences [KMJ15].

To address the intended matching as discussed above in a more theoretic manner, for a specific concept Kazemzadeh, Milton and Johnson apply the notation $<ai>$ and for a

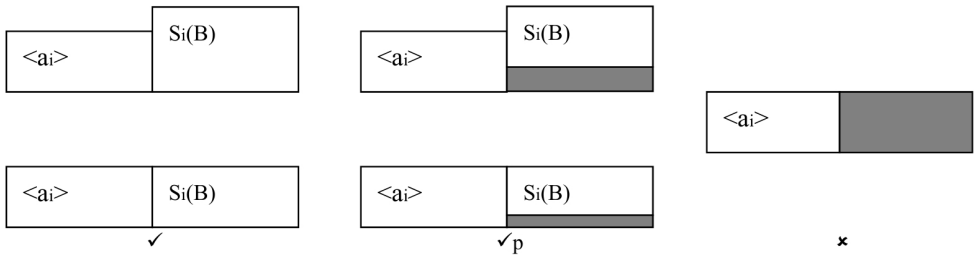


Figure 2.18: Coverage of concept *<ai>* by supportive concept *Si(B)* [KMJ15]

combination of several aspects *<ai+...+aj>*. Furthermore, they use *Si(B)* to indicate elements of a modeling language that can be utilized to cover a specific concept of the methodology under observation. Within their paper, the authors also try to visualize their approach as shown in Figure 2.18. In accordance with the classification of the coverage as explained above, the elements on the left side represent a full, the one in the middle a partial and the right side is without any coverage [KMJ15].

Using this kind of formalized approach, the authors examined each concept form every side.

2.5.2 Comparison of Concepts

This section is intended to give a summary of the findings from Kazemzadeh, Milton, and Johnson and to provide a deeper understanding of the correlation of the individual concepts of both modeling languages. Along the way, the provided findings are discussed and comments about the applicability for the context at hand are added. Furthermore, the upcoming paragraphs constitute the first opportunity to derive ideas targeting extensions and enhancements of the concepts. These insights should aid the development process of the transformation later on.

Coverage of Service Blueprinting

In a first step, the authors focus on the more customer-oriented modeling technique Service Blueprinting and discuss for each semantic element if a matching counterpart can be found within the catalog of the Business Process Model and Notation.

But to follow this approach, initially, the methodology itself and its contained concepts have to be specified. Again, this is especially important for Blueprints since there is a lack of standardization and therefore no predefined set is available. In comparison to the version that was applied within the chapter targeting the Case Study, during their paper Kazemzadeh, Milton and Johnson use a simpler notational representation of a service that is similar to the one suggested by Mary Jo Bitner, Amy L. Ostrom, and Felicia N. Morgan in their corresponding paper (see also Figure 2.5) [BOM08]. Nevertheless, the most important semantic elements are covered using this basis. Examples for missing aspects would be the indication of waiting and failure points.

Concepts of service blueprinting	Degree of overlap	Supportive concepts of BPMN
<Action>	✓	*<Activity>*
<Actor Categories>	✓	*<Pool* plus *Lane>*
<Action Flow>	✓	*<Sequence Flow* plus *Message Flow>*
<Communication Flow>	✓	*<Sequence Flow* plus *Message Flow>*
<Line of Interaction>	✓	*<Pool* plus *Lane>*
<Line of Visibility>	✓	*<Pool* plus *Lane>*
<Line of Internal Interaction>	✓	*<Pool* plus *Lane>*
<Line of Order Penetration>	×	
<Line of Implementation>	✓	*<Pool* plus *Lane>*
<Props and Physical Evidence>	×	

Figure 2.19: Coverage of Service Blueprinting by BPMN [KMJ15]

As a starting point for their discussions, the authors list the concepts that were identified as being a component of the method Service Blueprinting as shown within the left side of Figure 2.19 [KMJ15]. For a detailed explanation of these elements, please have a look at the chapter targeting the nature and characteristics of the modeling approach.

The findings of the step by step comparison performed by Kazemzadeh, Milton and Johnson can be seen in Figure 2.19. In general, due to the rather distinct purposes and intentions behind the two methodologies, one can say that Blueprinting, under consideration of the tool-set as represented above, is probably the more abstract one. BPMN in version 2.0, on the other hand, offers a very complex and diverse catalog of possibilities. Therefore it is not surprising, that the majority of semantic components can be illustrated while applying this more comprehensive approach [KMJ15].

Looking at the element *<Action>*, which constitutes a specific time-consuming operation that is executed by a participant of the service in accordance with the corresponding affiliation determined by the separating lines. While Blueprinting in the form as shown above only suggests one concept at this point, BPMN offers several different variants of the type *<Activity>* that inherit different semantic characteristics and can easily be applied [OMG11].

In addition to the coverage of this core component of Service Blueprinting, the logical order that is necessary to display the intentions behind the service, the *<Action Flow>* respectively the *<Communication Flow>* can also be covered while applying BPMN. Considering the set of concepts of the more abstract modeling approach, as discussed in the article at hand, only a very simple sequence flow is illustrated without the usage of any alternative or parallel paths. While comparing this version with the very comprehensive process modeling language, it is not surprising that the authors came to the conclusion that this point can be rated as fully covered [KMJ15]. Nevertheless within the literature also more complex routing mechanisms including decision points were introduced to Service Blueprinting and also applied during the creation of the previous case study. But even these can easily be covered while applying the BPMN mechanisms like the different *<Gateways>*. Therefore the assumption that the *<Action Flow>*, as well as

the *<Communication Flow>*, can be covered by the *<Sequence Flow + Message Flow>* of BPMN still holds.

The next concept that was listed in Figure 2.19 are the *<Actor Categories>*. They are used to distinguished the different participants that can be observed while considering a specific service. In contrast to other modeling languages, within Service Blueprinting these categories, are of a more general nature and more or less fixed. Throughout the literature different suggestions have been made, but the most common ones are customers, front-stage actors, back-stage actors, support, managerial actors and IT systems [LW11] [ZBG06]. Within the model itself, the lane separation is applied dividing the canvas into the corresponding areas. Coming back to the comparison as done by Kazemzadeh, Milton, and Johnson, targeting this aspect of the methodology, they suggest to use BPMN's *<Pools>* and *<Lanes>* since they offer similar organizational possibilities. According to the authors, this way a full coverage can be achieved.

A very interesting issue that probably also needs to be discussed in more detail when it comes to the design of the actual transformation later on, concerns the topic of the different separating lines. As also explained within the chapter concerning the nature of Service Blueprinting, a main aspect of the methodology consists in the organization of the actions and the corresponding sequence flows while applying separators. Each line implies a specific semantic meaning for the areas that are created between them. BPMN on the other side offers the two distinct notations *<Pool>* and *<Lane>* to structure the illustrated process. The authors now suggest to use a combination of these elements to cover the separating lines [KMJ15]. However, due to the more generic nature of the BPMN elements, there are various ways how this can be accomplished. One exception at his point is the concept *<Line of Order Penetration>*. Within the context of a service, this conceptual element is used to distinguish between actions that contribute to the customer-oriented value generation and those that do not [BY05]. Kazemzadeh, Milton, and Johnson argue that BPMN simply does not offer an adequate possibility to address this issue [KMJ15]. The reason for this could be a result of the purpose of the modeling approach. Due to its process orientation without the customer-centric viewpoint that is inherited by Service Blueprinting, up until now this was probably not necessary.

The second concept where a corresponding supportive counterpart could not be identified by the authors as shown in Figure 2.19, concerns the tangible objects of a service process [KMJ15]. Within the customer-oriented modeling approach, this is an important component due to the impact it might have on the overall value of the offering and the customer's experience. This aspect was also discussed in more detail during the chapter targeting services in general. Once more this shortcoming on the side of the process-centric methodology can probably be traced back to its purpose. Although there do exist notational elements in the language to capture more static parts of a process, these are designed to illustrate aspects of information technology and therefore do not correspond well to the physical nature that would be required for this context.

When looking at the methodology that was applied during the case study of this work, one can see that the concepts *<Decision Points>*, *<Failure Points>* and *<Waiting*

Points> were not part of the notational set suggested by Kazemzadeh, Milton, and Johnson. Nevertheless, they tend to be considered as one of the more popular extensions throughout the literature and therefore should be taken into account during the side by side comparison [LW11].

The first element <*Decision Points*> is subject to the logical order of the different <*Actions*> and was briefly addressed before. It enables the designer to integrate alternative paths into the Blueprint. Looking at BPMN, the quite variable <*Gateways*> are more than suitable to cover this aspect.

The <*Failure Point*> is the next semantic element that was not covered by Kazemzadeh, Milton and Johnson and is meant to indicate positions within the overall service sequence flow, where a failure might occur that possibly influences the overall execution phase and has a negative impact on the customer's experience and perception of the offering's quality [ZBG06]. Looking at the general concepts that are available within BPMN, due to the characteristics of the <*Failure Point*> in Service Blueprinting as described above, it should match the type <*Event*> of the process-oriented modeling language. While reviewing the corresponding catalog, the so-called *Error Events* can be found. They are applied to a diagram to indicate that the occurrence of a potential error might start or end a specific process, or terminate a certain activity [OMG11]. Therefore this BPMN element should be sufficient to capture the described additional aspect of Service Blueprinting as represented within the literature and the case study of this work. However, there might still be an issue while directly comparing the two sides with each other. Once more it has to be noted that, due to its purpose and nature, in contrast to BPMN, Service Blueprinting is not supposed to illustrate the underlying sequence flow with all its details. This can also be assessed when looking at the treatment of errors. While the service-oriented language settles for the indication of the specific situation within the diagram, in BPMN the designer has also to specify what happens if the situation occurs. Without illustrating the impact of the failure and possible resolving sub-processes, the diagram would not be complete. Therefore the process modeling language is obligated to capture the same situation with a greater level of detail. While this is not an issue for the matching at this point, it will definitely be a topic that needs to be specifically addressed when it comes to the design of the transformation itself.

The final conceptual element within this context is the <*Waiting Point*>. In a Service Blueprint it is used to specify positions contained in the process flow that are subject to potential extensive waiting times [LW11]. Of course, this always has to be assessed under the consideration of the nature of the specific task. If the action does require a long period of time to finish in general, it does not necessarily need to be marked with a waiting point. Only if there is the possibility that it results in a delay that exceeds the acceptable limitations and as a result has a negative impact on the customer's service experience, the corresponding point should be indicated while using this label. Looking for a match on the side of BPMN, one might be tempted to choose an element of type event, quite similar to the issue of the failure points. Like this marker, it is also intended to illustrate implications for the customer which are not explicitly addressed within the

Concepts of BPMN	Degree of overlap	Supportive concepts of service blueprinting
<Activity>	✓	*<Action>*
<Event>	×	
<Pool>	✓p	*<Actor Categories>*
<Lane>	✓p	*<Actor Categories>*
<Sequence Flow>	✓	*<Action Flow* plus *Communication Flow>*
<Message Flow>	✓	*<Action Flow* plus *Communication Flow>*
<Message>	×	
<Gateway>	×	
<Data Object>	×	
<Text Annotation>	×	
<Association Flow>	×	
<Group>	×	

Figure 2.20: Coverage of BPMN by Service Blueprinting [KMJ15]

context of BPMN. However, there is a difference when it comes to the impact on the sequence flow itself. While a failure obviously needs to be directly handled by the process somehow, the occurrence of waiting time has not. Therefore this aspect does not inherit a process relevant characteristic and can be considered as merely informative for the designers. Due to the missing customer orientation of BPMN, despite the important role within Service Blueprinting, a possible counterpart cannot be identified.

Coverage of the Business Process Model and Notation

As mentioned above, within their paper Kazemzadeh, Milton and Johnson also discuss which elements that are part of the official standard of BPMN in its version 2.0 can be covered while applying components of Service Blueprinting.

Again, before starting with the actual matching process, the concepts that shall be addressed later on have to be specified. But in contrast to the service-oriented language, for BPMN this is an easier task due to the presence of the well defined and official standard. The authors summary of its content can be seen on the left side of Figure 2.20 [KMJ15].

As one can see, while looking at this direction of the comparison, the resulting degree of overall coverage as displayed at this point is very different to the previous section. However, this is probably not really surprising considering the distinct characteristics of both model languages. While Service Blueprinting tries to illustrate the general sequence flow and the most important aspects of the offering with a special focus on the customer's experience, BPMN, on the other hand, has the intention to describe a very detailed representation of a process orchestration. Therefore the latter methodology naturally needs to provide a wider variety of semantic components to capture quite comprehensive situations.

Nevertheless, there are some components of BPMN that can be covered while applying

Service Blueprints. The first example in Figure 2.20 is the *<Activity>*. It is the core element in the process-oriented language to illustrate time-consuming operations executed manually or automatically. It may be atomic or a composition of a finer-grained sequence of further activities. The latter case is also called sub-process [OMG11]. According to Kazemzadeh, Milton, and Johnson, because of the more general nature of the service counterpart *<Action>*, also this varying level of abstraction can be captured. Therefore they argue towards a complete coverage of the concept [KMJ15].

However, in version 2.0 of BPMN, there are also additional indicators which can be applied to a task element to introduce additional semantic meaning to an activity. As explained during the chapter targeting the concepts of BPMN, they can either influence the execution behavior (i.e. activity markers) or specify the nature of the activities [Ber15]. Examples would be the loop marker or the send task. Considering the simpler concept of Service Blueprinting, it seems not possible to match this semantic variety. Therefore the full coverage as discussed by the authors is only valid for the simple task and the sub-process.

The other two BPMN concepts that are subject to a full coverage as shown in Figure 2.20 are the *<Sequence Flow>* and the *<Message Flow>*. The first concept represents the logical order of flow elements (i.e. activities, events, gateways) within a specific BPMN pool. The second one captures the flow of messages between different pools. Looking at Service Blueprinting, the sequence of the contained elements is introduced while applying the *<Action Flow>* and the *<Communication Flow>*. But it is not possible to match these concepts separately, due to the distinct ways how the process orchestrations are displayed by the two modeling approaches. To overcome this issue, a combination is applied which enables the coverage of the BPMN concepts [KMJ15].

In contrast to the *<Message Flow>*, its conceptual content the element *<Message>*, that is transmitted along the communication between the different *<Pools>*, is not covered by Service Blueprinting [KMJ15].

A very important control mechanism concerning the process flow within BPMN is the *<Gateway>*. The standard of the Object Management Group (OMG) offers a set of different versions that inherit specific routing semantics like the parallel gateway or the exclusive gateway [OMG11]. Only when utilizing several of these options, it is possible for the designer to capture very complex processes with a comprehensive level of detail. Blueprinting, on the other hand, does not support such mechanisms. The very traditional notational set as applied within the paper of Kazemzadeh, Milton and Johnson refrains from any control object and so only allows the illustration of one possible sequence of actions. This is also the reason why an explicit representation of the *<Action Flow>* using lines and arrows is not necessary. In accordance with this observation, the authors have marked this issue as not being covered by Service Blueprinting [KMJ15].

More complex sets presented within the literature and also the one that was applied during the case study concerning the emergency admission, often suggest the use of so-called decision points. Of course, these require an explicit representation of the flow

but enable the designers to display more complex services while introducing alternative paths. This way the exclusive gateway of BPMN can also be covered. Furthermore, in Service Blueprinting sometimes parallel sequences are illustrated using multiple in- and outgoing paths while directly connecting the actions [Sha10]. By adding this concept, additionally, the parallel gateway can be matched. But even with these extensions of the base set of semantic elements, the overall topic *<Gateway>* cannot be fully covered.

At this point, one aspect that is part of the BPMN sequence flow is still missing. Within the official standard, an *<Event>* marks a position in the process where a specific situation occurs and has an impact on the execution. The three main categories are start-, intermediate- and end-event determining the position within the diagram and impose a certain semantic behavior (e.g. end-event is terminating). Additionally, it is possible that an event is designed to throw or catch the corresponding information. Finally within BPMN version 2.0 there are a number of different types describing the characteristics of events which may occur in a real-world process (e.g. *Error*, *Timer*, *Message*) [OMG11].

Looking at the more traditional set, there is no corresponding counterpart on the side of Service Blueprinting [KMJ15]. However, as explained during the matching of BPMN against the service-oriented language, when the concept *<Failure Point>* is added to the applied notations, for this previous matching direction the BPMN element *Error Event* can be used to cover this aspect. But vice versa there are some complications imposed by the semantic characteristics of the BPMN element. As mentioned above, an *<Event>* naturally has an impact on the flow within the diagram. As a result, the model has to provide additional paths to handle the occurring events. But a *<Failure Point>* as applied within the context of Blueprinting, on the other hand, does not result in such a model behavior. Therefore also the extended notational set does not even partially cover the BPMN concept *<Event>*.

The next elements that shall be discussed at this point are *<Pools>* and *<Lanes>*. In BPMN they are used to organize the process flows and the grouping of cohesive tasks. While members of the first category encapsulate whole, concluded processes that are linked with each other using the *<Message Flow>*, the latter one constitutes compartmentalization within such a *<Pool>* [OMG11]. In general, they inherit a more generic nature and it is up to the designer to decide which aspect of the intended process should be captured while using these semantics (e.g. different groups within an organization, different organizations). According to the authors of the article at hand, this characteristic imposes a difficulty while looking for a coverage on the side of Service Blueprints. Although it is basically possible to cover a BPMN segmentation while utilizing the *<Actor Categories>*, this is only possible if *<Pools>* and *<Lanes>* are designed to match the Blueprinting categories in the first place (e.g. customers, front-stage agents, back-stage agents). But due to the versatility of the concepts of the process language, only a partial matching can be identified [KMJ15].

When looking at Figure 2.20 provided by Kazemzadeh, Milton, and Johnson, the concepts that were not discussed up until now are the *<Data Object>*, *<Text Annotation>*, *<Association Flow>* and *<Grouping>*. According to the authors, all these elements share

the same fate and a coverage (partial or full) is not possible while applying the concepts of Service Blueprinting [KMJ15]. This statement does also hold when using the extended set as applied during the case study. These elements are not a part of the actual sequence flow within the process and therefore do not have any impact. Of course, they share the same context and are associated to specific points or elements, but for the execution itself, they do not impose any constraints.

Conclusion

After the application of the method of conceptual evaluation as done by Kazemzadeh, Milton, and Johnson and discussed within this section, it is probably safe to say, that the Business Process Model and Notation is more capable to cover the concepts as introduced within Service Blueprinting in contrast to the other way around. The reason for this observation can be found within the fact that BPMN (especially in version 2.0) simply offers a wider variety of concepts that also inherit more generic characteristics (e.g. pools and lanes) which allow them to be applied on very different process situations and under distinct intentions. Service Blueprinting, on the other hand, is meant to create a graphical illustration of a service offering under the consideration of the customer's experience. This very strict definition results within the limited notational variety throughout the literature. The fact that the language is often applied within cross-functional teams that also include business experts that do not possess profound modeling knowledge, supports the usage of a simpler set of concepts.

Nevertheless, there is a gap while using BPMN to match the concepts of Service Blueprinting. This can probably be traced back to the diverging purposes of the languages. As mentioned above, while the service-oriented approach is utilized using the customer's point of view, BPMN has the intention to model processes considering the providing organization including a special focus on information technology.

The insights that are a result of this conceptual comparison, provide the first possible points of combination between the two domains in respect of the intended matching. Within the upcoming chapters, this knowledge shall be used as a basis and further developed towards possible transformations.

CHAPTER 3

The Case Study

A core aspect of this thesis consists within the examination of the applicability of the intended transformation of Service Blueprints towards the Business Process Model and Notation. Considering this goal and the scientific questions that were listed during the introductory chapter, it is simply not enough to only review the characteristics and issues on a merely theoretical basis. For the modeling area, as well as many other domains, it is true that within this context, the connection to real-life challenges and the ability to address those essentially contributes to the quality of an approach.

So to be able to target the transformation according to these demands, at this point a case study shall be introduced. It will be used later on as a basis to create the matching of the concepts provided by the two modeling languages. Furthermore, in the end, the designed model shall serve as a proof of concept while applying the prototypical implementation.

For this purpose the process of the *emergency admission of a patient at a surgical ward within an Austrian hospital* will be documented. Accordingly, the next chapter will focus on the definition of the case and the way how the necessary information was gathered. Afterward, the case itself shall be described in more detail. As a conclusion of this chapter and basis for the upcoming tasks, a Service Blueprint illustrating the service process shall be created.

3.1 Collecting the Information

While looking for a suitable case, due to the nature of the starting point of the transformation, it was clear that a service-oriented process is needed to accurately examine the corresponding aspects of the modeling techniques. More precisely it should be a service flow between various agents including the direct interaction with the customer.

As classical examples, within the literature often the processes of restaurants or hotels are used, since most people can relate to such situations and so it is easier to understand the cases in their basic form [LW11] [Lov14] [BOM08]. Nevertheless, for this work, the intention was to capture a more specific service, that also provides a certain degree of complexity. To meet this requirement, after some considerations, the decision has been made to illustrate the *emergency admission of a patient at a surgical ward within an Austrian hospital*.

The background knowledge which is required to properly illustrate the case with all its facets was acquired in cooperation with members of the organizational team that executes the process. To do so, several discussions have been held where at first more abstract and later on, quite detailed information was gathered. After the initial session, the retrieved data was used to draft the first rough Service Blueprint of the process. This model was then applied as the basis for the subsequent talks and enhanced in a collaborative approach. The final result can be seen in Appendix A.

To give the reader a better understanding of the observed service, the upcoming section shall describe the individual steps that are necessary to process a patient within a surgical ward when an urgent situation is determined.

3.2 The Case Description

The case itself describes a process in a hospital where a critical situation is detected that requires a surgical response within a very limited time frame. In contrast to a planned surgical procedure, the medical, as well as the administrative process steps, have to be orchestrated under consideration of the emergency at hand. This means that it is simply not possible to allow the overall sequence of actions to inherit dependencies and characteristics that may result in longer waiting periods. Everything has to be aligned and executed in a manner that allows the organization (i.e. the hospital) to treat the patient in accordance with the urgency of the medical problem.

To begin with, it is necessary to explicitly define the start and the end of the process. For the purpose of this work, the initial step consists of the situation that for a certain patient a critical and urgent medical problem is observed. The concluding end of the sequence flow, on the other hand, is marked by the transfer of the person in need of medical attention to the actual surgical process. So the target of the intended model is to capture the emergency admission of a patient on a surgical ward, which includes the preparation of the procedure, the prior medical treatment, and the administrative processing.

Looking at the overall service process, one can determine the following agents that actively participate during the execution phase.

- Patient
- Nursing staff
- Surgeon
- Internist
- Ward physician
- Anesthetist
- Lab staff
- Transportation and administrative staff
- X-ray team

As described above, the assessment of a critical and acute medical problem that needs to be treated in a surgical way defines the starting point of the sequence flow. At this point, the patient is already present at a medical facility that performs the initial examinations. It is not possible to enter the surgical ward directly without passing through at least the ambulance of a hospital. In very extreme and rare cases, theoretically, a patient could be directly forwarded by a medical doctor who runs his or her own medical practice.

Independent of the nature of the transferring institution, if a surgeon is not present or directly available during the preliminary examinations, an external doctor has to be consulted and informed about the case at hand. After a corresponding review of the problem, it has to be determined if the patient takes blood-thinning medication (i.e. hemodilution) which could lead to a life-threatening situation during the surgical intervention (i.e. danger of bleeding out). If so, the ongoing process differs from the nature of the compound. If it is not possible to antagonize its effect with another medication, the surgery has to be postponed and the patient is moved to the general ward. Otherwise, the surgical ward and the theatre are being notified about the new case. If it is necessary that another surgeon is in charge of the task, later on, he or she is also notified.

Now it is time to physically transfer the patient to the surgical ward. The medical report that was the result of the preliminary examinations is passed on as well. At this point, the process flow starts to split up since, due to the urgency of the situation, multiple tasks have to be executed in parallel. Overall a general distinction between a path involving the doctors and one for the nursing staff can be made.

For a better understanding of the process, the actions of the latter one will be described first and when arriving at the concluding synchronization point, the first one shall be explained as well.

After the arrival at the ward, the first step for the nursing staff is to physically assign a bed and settle the patient in. As it is common for such medical facilities, due to its critical characteristics, every patient is connected to monitors which are used to continuously track his or her current situation (e.g. oxygen saturation, blood pressure, heart rate). Along with the data that is gathered this way, also the condition that is expressed verbally is used as a basis for the next steps. Another very important aspect within this context is the determination of possible allergies. In fact, this inherits such a critical nature that each member of the medical personnel that is involved (doctors and nursing staff) asks the patient individually to minimize failures as a result of miscommunication or misunderstandings. Afterward, the nursing staff has to decide if and which immediate medical actions have to be performed to stabilize the patient until everything is ready for the surgery. If there is the need for some treatment (e.g. patient is in pain), the next step consists of the execution of the necessary steps. To administer the medication, an intravenous cannula is used. Sometimes, if the patient gets forwarded by another hospital ward, it is already present and can be used. Otherwise, the staff has to insert a new one.

After receiving the initial medical treatment, the patient has to be registered in the hospital's administrative systems. If it is the case that he or she was transferred by another ward of the same institution, some of these steps may be omitted due to the fact that the basic data is already present in the infrastructure. If not, the first step is to enter the patient's e-card. The ERP system extracts the person's core data and creates a new entry, which then must be completed manually. As a result, the patient's information is automatically transmitted to the ward's organization system, where the bed assignment is recorded. Afterward, a bar code corresponding to the individual can be created. It is used to identify all upcoming documents or artifacts that will be part of the further process steps (e.g. medical records, labeling of blood samples). As a final task concerning this administrative work, the patient's data and the already performed treatment have to be entered in the wards documentation system.

At this point, the process flow continues directly at the bed of the patient, where he or she receives a wristband featuring the previously generated barcode. The next step depends on whether an intravenous cannula was already inserted previously to administer pain medication or not. If the latter is the case, this task has to be performed now, to allow the surgical team to rely on it later on. The determination if the cannula is already present or not also influences the next activity that consists of the drawing of blood. If it is still missing, the samples can be taken while inserting the new intravenous access. If this is not the case, it has to be performed while using another method (i.e. winged infusion set). The resulting vials are then sent to the lab for further examinations.

Once the samples arrive, they are subject to various tests depending on the instructions given by the ward physician during a separate process flow that runs in parallel and will be treated in this section later on. Examples would be the determination of the blood clotting, amount of blood corpuscles and measurement of the electrolytes. So the activity of examining the blood has dependencies to two distinct sequences. The results are then sent back to the requesting station.

The next step consists of the reservation of the blood supply that is necessary to safely perform the surgical intervention. This is only done if the internal approval has previously been signed by the doctor and therefore the surgery is not canceled. The request is originally submitted by the ward physician who is responsible for the initial examination of the patient and is passed along with the blood samples. After the lab team is finished with their tests, they prepare the blood units.

Afterward, the upcoming tasks that have to be executed, now depend on the condition of the patient. If the case is highly acute (i.e. immediate danger to life), then the staff directly proceeds with the last preparations for the surgery. Otherwise, there are still some preliminary steps that have to be taken care of.

During the first one, the nursing team performs the anamnesis, where information concerning the organizational situation is recorded. This includes questions like to which degree the patient is autonomous and where he or she requires the support by the nursing staff (e.g. washing). Furthermore, also more general data is gathered (e.g. contact number of relatives). Then the patient (or his or her relatives) has to fill out some administrative forms as a legal requirement (e.g. disclaimer of liability). Once this is done, and an electrocardiogram (ECG) was not already taken before within the context of the initial clinical record, this should be done now.

Afterward, the patient has to change into the surgical clothing and remove any jewelry, prostheses (especially dental) and piercings. If the case at hand is highly acute, the next step consists of filling out the checklist as a necessity for the upcoming surgery. If not, the patient is given a sedative medication (e.g. Dormicum) as a preparation for the real anesthesia that will be administered during the first phase of the surgery.

Now a point is reached, where the subsequent processes can only be executed if all preceding tasks are completed. As stated before after the patient was physically transferred to the surgical ward, the actions performed by the doctors and the nursing staff run in parallel. The latter ones were already explained above, but the other category is still missing.

When the patient arrives at the ward, the internist checks if the initial clinical report is up to date and sufficient. If not he or she organizes additional examinations like an x-ray or blood tests. After all the results have been received, the decision has to be made if the surgery should be performed or not. The reason for a cancellation could be that, with a high probability, the patient would not survive the intended procedure. But if the doctor decides in favor of the intervention, he or she has to sign the internal approval.

If during the initial examinations it came out, that the patient takes a blood-thinning substance that needs to be antagonized, a corresponding medication is administered at this point. Afterward, the sobriety is determined as a prerequisite for the safety of the surgery. Another aspect within this context concerns again the identification of possible allergies. As already noted above, this is a point that has to be performed by every medical agent in the process, to approach this critical aspect as carefully as possible.

Another topic that has to be targeted, is the preparation for the anesthesia later on. Therefore, the doctor in charge gathers and records the physical data like height and weight. This is necessary to correctly adjust the medication. Now the patient has to be informed about the upcoming anesthesia as well as the actual surgical intervention. As a result and legal requirement, a written confirmation and allowance has to be signed (one for each part).

At his point, all preceding processes should be completed and there is again the question to which level the case is acute. In dangerous situations, the patient is directly transferred to the surgical theatre for the intervention. Otherwise, depending on the sobriety of the patient that was determined by the anesthetist, the surgery is set on hold until an appropriate level is reached.

The last activity that constitutes a part of the overall service process is described by the physical transportation of the patient into the operating room.

3.3 Creating the Service Blueprint

The case information as described above was used to capture the overall service flow in a Service Blueprint. As previously indicated, the first model was created after an initial talk and then extended to a great degree during ongoing discussions. Even during the phase of writing this chapter, several additional aspects of the process seemed to require further examinations and therefore were added later to the case description and the corresponding model.

The first ideas for the model itself were captured simply using pen and paper. But of course, this approach has some limits when it comes to action flows that have more than a couple of tasks and need to be reworked a number of times. So for the further creational steps, a computerized method was necessary to aid the design process. After looking at several available tools, due to the provided freedom when it comes to the notational diversity and the applicability of the software within the context of the prototypical implementation which will be discussed later on, the diagramming platform *Draw.io* of the equally named provider has been used[Dra]. In accordance with the fact that Service Blueprints do not follow a fixed standardized set of concepts, the tool provides a very high degree of variability during the creation of the model.

Nevertheless, of course, it was necessary to agree on a fixed notational approach at some point. So to illustrate the service elements of the *emergency admission of a patient at a surgical ward within an Austrian hospital*, the elements as shown in Figure 3.1 have been applied. As one can see, the core elements *Action*, *Decision point*, *Sequence flow*, *Start point* and *End point* are similar to the more classical flowcharting as presented throughout the literature. The markers for the *Failure point* and *Waiting point* however are represented as suggested by Christopher Lovelock and Jochen Wirtz [LW11], but with the distinction, that each element is directly associated to a specific *Action*. This should make it easier for the user to identify the corresponding positions in the model.

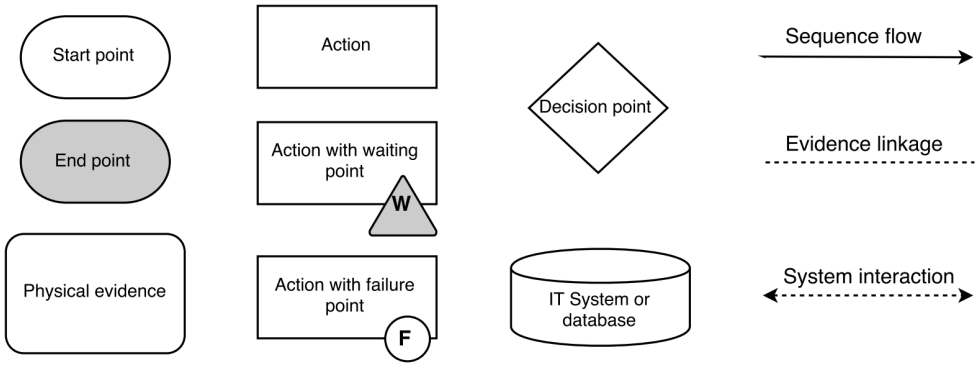


Figure 3.1: Graphical notation applied for the case study [Dra]

Another distinction to the more traditional Blueprints as presented in the literature is the linkage of the *Tangible Objects*. Very often they are simply positioned within their corresponding lane without any explicit association. This is fine for smaller Blueprints, but in case of more comprehensive ones, the assignment without some kind of indication is not so easy anymore.

When it comes to the lane separation, within the literature a number of different possible lanes and resulting areas have been proposed [Sho84] [BOM08] [FK04]. As stated within the chapter targeting the concepts of Service Blueprinting in general, depending on the intention of the modeling approach, the situation at hand and the available knowledge of the model designers, not every lane has to be applied. Therefore, for this case study, a lane separation as has been introduced, dividing the canvas into the following areas.

- Physical Evidence
- Patient Actions
- Front-Stage Interactions
- Back-Stage Interactions
- Support Processes
- IT Elements

During the design of the chosen service case, two main diagrams have been created to represent the required processes. Besides the detailed version, also a more abstract model has been developed to be able to provide a simpler overview. The resulting Blueprints illustrating the service process as described above can be seen in Appendix A, along with the transformation result of the former one. The Service Blueprints will be used as a conceptual basis for the upcoming discussions targeting a possible transformation between the two methodologies.

CHAPTER 4

Development of the Transformation

This chapter and the contained sections will form the core element of the thesis. After the examinations of the two transformation sides, Service Blueprinting and the Business Process Model and Notation (BPMN), it is now time to focus on their combination.

But of course, like everything else in life, this undertaking is definitely not a straight road and distinct paths can be taken to reach the intended goal. That said, not every possible way or solution provides the same utility. This means that to increase ones chances to be able to come up with a satisfying resolution, its development has to be well planned and elaborated.

For the context of this thesis, as an implication it is not enough to look upon each side separately, but one has to consider also the commonalities and differences of the two languages when compared with each other directly. The insights that were gathered this way in combination with the basic information, may than be used to design a possible approach. But this would entail the risk that one becomes fixated with the new solution at hand and do not consider other possibilities that could overcome the limitations that are imposed with a very high probability. So to prevent this issue, it might be better to think about and consider several approaches and examined their characteristics, advantages and disadvantages for the users as well as the created models. Doing so might provide the necessary input to make out one solution that fits the intended purpose best.

In accordance with the demands as explained above, the next topic to be discussed is the actual transformation while taking the insights that were gathered in Chapter 2 into account. Along the way, several different partial approaches shall be explained. Afterward, their characteristics will be discussed and a recommendation for one single overall solution shall be given.

4.1 Designing the Transformation

Within this chapter, the process of transforming an existing Service Blueprint into a corresponding model of the Business Process Model and Notation will be developed. To do so, the chapters targeting the two methodologies individually as well as the comparison between them that was discussed in the previous chapter shall serve as the theoretical basis. To be able to relate the specific concepts that are drafted while addressing this issue to the domain of its application, the model created within the context of the case study will serve as a practical example.

After outlining the preconditions that are necessary to start with the conceptual development, possible solutions and variants for the element-wise transformation will be discussed.

4.1.1 Theoretical Considerations targeting the Area of Model Engineering

Before the actual transformation will be targeted as the main subject of the ongoing discussions, this section shall serve as a remark concerning the classification of the planned undertaking within the domain of the scientific area of model engineering in general.

Theoretical Basis

First of all, a model can be considered as an abstract representation of a real-world phenomenon. For example, this could be a specific object (e.g. a car), a mechanism (e.g. automaton) or a dynamic system (e.g. service process). According to its nature, a model can be categorized as being a static or dynamic representation. Overall, they are necessary to cope with the vast complexity of reality and, as an implication, every aspect that can be seen as an artifact of such (e.g. process flows within an organization) [BCW12].

Model engineering, as seen within the context of information technology and especially software engineering, has the creation and processing of these abstractions as a main objective. Among others, the core aspects are the fast technological progress within this domain as well as its dissemination throughout our daily life that makes it necessary to come up with concepts that enable a corresponding development. Along the way, an important part concerns the validity of the assumptions made and, in general, the final abstraction [BCW12].

One area in this field of study also targets the transformation of a specific model. Within this regard, according to Marco Brambilla, Jordi Cabot, and Manuel Wimmer, the following dimensions can be distinguished [BCW12].

- Horizontal vs. vertical
- Endogenous vs. exogenous
- Model-to-text vs. text-to-model vs. model-to-model

Looking at the first aspect of this list, it should be noted that while reviewing a real-world phenomenon using a model, the way how this is done is subject to a specific level of abstraction. So, for example, it is possible to create a representation of an organizational process with a different degree of fidelity which ranges from vague orchestrations of rather general procedures to exact constellations including expected timings, possible delays, and detours. In this regard, it is possible that the method that is applied to create the different levels of abstraction is the same. However, this does not have to be the case. Either way, if such representations with different natures should be created (e.g. drill-down), a corresponding transformation that converts the corresponding models between the layers seems advisable. Reasons for this are the maintainability of the situation in general as well as the efficiency during the model's creation. Concerning this first point shown in the listing above, such a conversion corresponds to the category of a vertical transformation. Thinking about a more practical example from the domain of software engineering, this kind of operation is performed between the representations of platform independent (i.e. neglecting the exact technical details) and platform-specific models (i.e. including the technical details). Accordingly, a horizontal transformation is the conversion of two model with a similar level of abstraction [BCW12] [Pol04].

The next issue is about the differentiation between endogenous and exogenous transformations. As indicated above, for a use case it is possible to apply very distinct tool-sets and methodologies. When thinking about the way how a specific conversion could be implemented, for a given source model, it is possible to target a result that is built up while using the same language or a different one. Accordingly, the first option is called endogenous and the second one exogenous. Going back to the last paragraph, the four categories can be combined to classify a specific approach. So for example, considering a vertical transformation there is the possibility to perform the conversion within the domain of the same modeling language and so create a model similar to the source, but with an increased level of detail. On the other hand, if a horizontal approach is chosen while maintaining the given methodology, the result inherits the same level of abstraction but may provide significant changes within its structure. A possible goal for this latter approach could be the improvement of the model's quality [SA16] [Men13].

The last category that was listed above, concerns the distinction of the transformations according to the nature of the source and target structure. In general there are the major classifications *text* and *model*. Whilst the model-to-model conversion describes the creation of target model inheriting a different structure while taking another model as a source, the model-to-text processing focuses on the generation of textual artifacts like program code. The last remaining option in this regard represents the other way around [BCW12].

However, these three dimensions are not the only base characteristics that can be applied to categorize a given model transformation. According to Marco Brambilla, Jordi Cabot, and Manuel Wimmer, the cardinality of the direction of the conversion can be taken into account as well. Probably the most straightforward approach is the 1-to-1 interpretation which takes exactly one source model and creates a single

output representation. Additionally, it is possible to have a 1-to-N, N-to-1 or a N-to-M transformation, which either uses the input as a basis for multiple targets, merges several sources to generate one goal state or consists within a combination of both [BCW12].

The last grouping terminology that shall be mentioned in this context and also was elaborated by Marco Brambilla, Jordi Cabot and Manuel Wimmer targets the differentiation between out-place and in-place transformation strategies. The basis for this characterization is the handling of the source model. Thinking about a processing mechanism bound to the generation of a specific target, it is either possible to create a completely new model that fits the requirements and leave the original one untouched or alter the source accordingly. After the second alternative, the initial state of the transformation input is no longer available. Of course, both strategies have different advantages and disadvantages and the preference heavily depends on the actual case at hand. However, going one step back to the dimensions that were listed above, the in-place transformation should probably be considered mainly within an endogenous undertaking [BCW12].

Classification of the Transformation of Service Blueprinting towards BPMN

The previous paragraphs were focused on the discussion of the terminologies that enable the theoretical classification of approaches targeting model transformations. The next task is dedicated to the application of the corresponding aspects on the undertaking that constitutes the core topic of the upcoming chapters. As a result, the intended concept and mechanism will be classified with respect to the domain of model engineering and the field of study that may benefit from the findings made within the context of this work will be characterized.

To begin with, the first step shall consist of the application of the three dimensions listed in the initial part of the last section. Targeting the differentiation between the horizontal and vertical transformation, the planned conversion can be considered as a member of the latter one. Although Service Blueprinting, but BPMN as well, can be applied on cases with the intention to maintain different levels of abstraction, in general, due to the smaller set of notational concepts of the service-oriented language, it tends to inherit fewer details than the corresponding counterpart. Accordingly, the planned transformation uses a model with a higher level of abstraction and generates a more detailed version within another methodology that shall than be used to further specify the representation. However, at this point, an aspect shall be considered that occurred during the actual development of the prototypical implementation. While creating the graphical BPMN representation of the target model, it was necessary to preprocess the Service Blueprint source to be able to avoid conflicting situations afterward. For a detailed discussion please have a look at the Sections 5.6.3 and 5.7.2. The goal of this step is to retrieve a Service Blueprint representation conform to the determined notational concepts, but with amended graphical components. Considering the current dimension, this step can be considered as a horizontal transformation since the level of abstraction does not change at this point.

The next dimension focuses on the nature of the methodologies used for the source and target models. This point is rather obvious for the main conversion and since the intention is to transform a given Service Blueprint into a BPMN model, the undertaking corresponds to an exogenous resolution. However, considering the preprocessing of the source model, both sides of the mechanism refer to the same modeling language. Therefore at this point, an endogenous approach can be assessed as well.

Moving on to the last point mentioned within the listing, the differentiation according to the type of the source and the target, the transformation of Service Blueprints into BPMN clearly belongs to the area of model-to-model conversions. Looking once more onto the graphical preprocessing, this statement still holds.

As discussed above, besides these dimensions, there is also the possibility to take the cardinality and the applied strategy into account while thinking about the classification. Considering the first aspect, for the actual transformation as well as for the mentioned additional processing step, the intention is to read in a specific source model and create exactly on output representation. Accordingly, the planned transformation can be considered to be a 1-to-1 conversion.

Moving on to the second aspect, the distinction between out-place and in-place transformations. When applying this classification to the mechanism developed within this thesis, once more there is a difference between the two modeling steps. First of all, the graphical pre-processing consists within an amendment of the original source model without creating a new one. Thinking about the definitions mentioned above, this pre-processing can be seen as an in-place transformation. The core component, on the other hand, creates a BPMN representation from scratch and does not alter the given input. Therefore, this step belongs to the area of out-place conversions.

After this theoretical considerations, the upcoming chapters will focus on the actual design and implementation of the prototypical transformation between Service Blueprints and BPMN.

4.1.2 Starting Points

Looking at the target of this chapter, the general aspects, and components that shall be a part of the final result need to be specified. Of course, the main methodologies are Service Blueprinting and the Business Process Model and Notation. But as mentioned in previous sections, the actual version respectively the contained set of semantic elements that is used to illustrate the service processes at hand, may inherit room for interpretations. However, to be able to carry out the upcoming steps in a structured and well-defined way, the key elements need to be determined.

Considering this aspect, for the result of the transformation that will be displayed using the process-oriented modeling language, basically several releases could be applied. Within the context of this work, BPMN in its version 2.0 that was published in the year 2011 with its complete set of available concepts and notations shall be utilized. Of course,

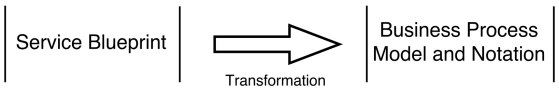


Figure 4.1: Direction of the transformation

due to the diverse level of abstraction between the two sides of the transformation, it probably will not be necessary to apply each semantic element that was proposed within the official standard. However, at this point, it shall be noted that for the version 2.0 no restrictions are imposed.

For Service Blueprinting, on the other hand, this issue is different. Once more, due to the missing standardization, no fixed tool-set can be applied, which results in the situation that a specification has to be agreed upon before starting with the creation of the transformation. For this purpose, it is probably a good idea to start with the best practices and most common elements that were suggested and used in the literature.

In general the concepts *Action*, *Action Flow*, *Communication Flow*, *Line Separation*, *Actor categories* and *Physical Evidence* need to be covered. Specifying this point, the *Action Flow* and *Communication Flow* will not be considered within its more simple form as for example applied by Kazemzadeh, Milton and Johnson during their comparison [KMJ15], but in a more complex one like the process illustrated within the chapter targeting the case study or as suggested by Arash Shahin [Sha10].

Concerning the *Line Separation* that is probably the most characteristic element of Service Blueprinting, the set as discussed within the chapter targeting the concepts of the customer-centric approach and as shown in Figure 2.3 shall be examined at this point. As stated, these also represent the most common categorizations presented within the literature.

Additionally, the concepts *Failure Point* and *Waiting Point* have proven to result in valuable insights during the creation of a Service Blueprint and therefore shall also be considered within the next steps.

Furthermore, if it seems helpful or necessary to add complementing concepts to the overall set, of course, they shall also be addressed during the upcoming tasks.

4.1.3 Further Preconditions

Besides the definition of the applied base methodologies, another specification has to be made as a precondition before considering the step by step transformation. As it is probably the case for most undertakings, to be able to work towards the intended goal in a more structured and efficient way, it is necessary to clearly specify the outcome.

In the case of this thesis, the target on a meta level consists of course within a process model conform to BPMN version 2.0 that was automatically created out of an existing Service Blueprint. However, up until now, besides the notational restrictions as suggested above, no further definition describing the nature of the result has been made.

Within this context, also the characteristics of the modeling language itself have to be considered. For example is Service Blueprinting intended to provide a simpler and customer oriented approach that shall enable the designers to focus on the overall service flow and not so much on the details that are required to set up the actual process later on. While considering the transformation towards BPMN, it is, of course, advisable to take this intention into consideration. When thinking about the extension of the basic set of concepts that could be introduced to achieve a higher rate of conformity towards the semantic elements of BPMN, this might stand in conflict to the reasons why this methodology was developed and is used in the first place.

On the other hand, following this demand results in the conclusion that the left side of the transformation (i.e. Service Blueprinting) as shown in Figure 4.1 remains more abstract than the target (i.e. right side). This means that, while applying the process on a practical example, it might still be necessary to enhance and complete the resulting BPMN model. Of course, if there is the possibility that a requirement of the process modeling language can be satisfied while extending the conceptual set of Service Blueprinting with a simple notation that does not stand in conflict with the methodologies nature, it should be added.

While reviewing the chapter targeting the nature and especially the advantages of Service Blueprinting, several characteristics have been identified that should be considered while developing the final result. The first aspect within this context was already addressed in the previous paragraphs and consists within the demand, that the approach shall still provide a simpler notational set that enables the designers to concentrate on the actual situation at hand. This also constitutes a condition for the next aspects that have been observed. As mentioned in the literature, besides its function to support the designers and the organization when it comes to the development of new services, a Blueprint may also serve communication purposes [BY05]. This means that due to its graphical nature it is theoretically suited to support entities while sharing these ideas and the corresponding information. Of course, this is only possible if a conceptual set is preserved, that has the potential to express the intended characteristics, but at the same time stay simple enough to be understood by different individuals without requiring advanced modeling knowledge about a specific language.

Additionally, Service Blueprinting also provides the possibility to use the created graphical models to be used to document an idea or an already existing service process [BY05]. For this purpose, a very important aspect consists of the conformity of the illustration towards a common understanding of process modeling concepts and their usage. Only if this requirement is satisfied it will be possible to a greater degree, that a model designed in the past can also be interpreted the same way after some time. Looking at the undertaking of this work, this means that when thinking about additional graphical elements or the orchestration of already known ones, this should happen in a way that is conform to the usage of similar concepts from other modeling approaches or areas. Due to the missing standardization of the modeling language, concerning this aspect, a strive towards an understanding in accordance with the best practice that can be observed throughout the

literature could contribute at this point.

Another topic that entails quite similar requirements as the previous one, targets the circumstances of the creation of a service as emphasized by Stickdorn [Sti16b]. It is probably quite rare that within an average organization a specific service offering is designed and developed by one person. Instead, a more collaborative approach is applied, including agents from several distinct areas within the environment of the service. As a result, it is the case that also persons participate in the creational process who inherit a business expert role but do not possess fundamental modeling knowledge. Again, while considering the result of the transformation later on, these circumstances need to be taken into account and as a result, a methodology shall be maintained that continues to support such an approach.

However, while considering the modeling language Service Blueprinting, also more fundamental aspects need to be considered. As indicated above, the nature of the methodology has should be taken into account while developing the transformation. Of course, this is also valid for its core characteristics. Two of them, where there is the potential risk that they are neglected while concentrating on the transformation are the customer focus and the integration of the physical pieces of evidence. The reason for this is, that when looking at BPMN, a deviation concerning these points can be observed as discussed in the chapter targeting the comparison of the concepts of both tool-sets. Now during the attempt of matching the two worlds, it shall be a part of the discussion to decide if such a transformation is necessary to achieve the intended goal.

In contrast to Service Blueprinting, the process-oriented side of the transformation may not face the same risks during the undertaking. The intention remains within the generation of a valid BPMN model that can then be completed while adding the missing details to be able to describe the service processes in a more comprehensive way. Due to the existing and rather strict official standardization by the Object Management Group, for this purpose it is required to keep the results within the boundaries of this specification. If additional concepts and corresponding notational elements should seem necessary or at least advisable to successfully transform a Service Blueprint, the conformity with the standardization has to be taken into account.

4.1.4 Transforming the Concepts

Within this section, suggestions shall be made how the transformation from Service Blueprinting towards the Business Process Model and Notation actually could be done. For this purpose, it is now necessary to consider every aspect of the languages that shall be a part of the process and examine the individual semantic elements and their potential counterparts.

Within the previous paragraphs, the starting points and preconditions were specified that will now be taken into consideration while approaching the step-by-step transformation of the concepts. The final recommendation targeting the transformation shall be part of discussions afterward.

To be able to elaborate the individual parts in a more structured way, the case study that was developed as explained in the corresponding chapter shall be utilized. But looking at the resulting model as shown in Appendix A, as one can see, due to the real-life situation, the overall sequence flow is rather comprehensive and complex and therefore probably not that well suited for the development to begin with. To overcome this issue, several partial components will be applied. Of course, to be able to cover all important aspects of the modeling languages, it is necessary to choose subsets that provide the required details. Along the way, these will range from simpler ones to be able to discuss the basic elements of the modeling approach, to the combinations of several concepts allowing the transformation of the more complex components.

The main part of this chapter shall be used to focus on the aspects that were identified as already being a part of the tool-set of Service Blueprinting. For this purpose, the elements suggested within the literature and especially the notations and semantics as applied within the case study will serve as a basis. Along the way, also possible enhancements and more complex approaches suggested within the literature shall be subject to the ongoing examinations.

Action

The first element of the methodology Service Blueprint that has to be considered is the *Action*. It represents the time-consuming performance of a specific operation carried out by some participant of the sequence flow. Within the overall model, an *Action* can be uniquely associated to a specific *Actor Category* that is a result of the applied *Line Separation* which will be discussed later on. As presented previously, due to the varying level of abstraction of the approach, an element of this category does not have to be atomic in terms of process steps. Nevertheless, a drill-down functionality that allows the designer to refine a certain point of the illustration within one model representation is, at least within the tool-set as applied in the literature, not intended. If this has to be done, another, finer-grained representation should be created. In contrast to other languages, only one version of the element *Action* is available.

Looking at the target of the transformation, of course, the Business Process Model and Notation offers a comparable element to capture operations executed during the process flow. *Activities* are meant to illustrate operations whether they are performed manually, automatically, by a user, or some kind of mechanism. Due to the more generic and versatile nature of the modeling language, in contrast to *Actions* of Service Blueprinting, there is the possibility to add more concrete semantic behavior to a default element using *Activity Markers* and *Task Types*. Furthermore, the methodology enables the designer to specify aggregated sets of *Tasks* as so-called *Sub-Processes*.

At this point, the coverage while considering the transformation, as shown in Figure 4.2, can be achieved without any problems and was also suggested by Kazemzadeh, Milton and Johnson within their corresponding article [KMJ15]. The observation that BPMN in general is more generic and able to support finer grained model representations

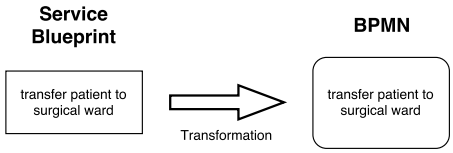


Figure 4.2: Transformation of *Actions*

makes this aspect rather straightforward. However due to the simpler elements of the service-oriented language, only the simplest form of the BPMN *Activity* can be supported which is called *Task*. Furthermore, the *Actions* do not offer any additional indication of semantics as offered by BPMN.

Action and Communcication Flow

The next concept of Service Blueprinting that shall be subject to the transformation is the logical arrangement of the specific *Actions*. Within the context of this methodology, two categories are distinguished. First of all, the sequence of operations within a specific *Actor Category* is called *Action Flow* and can be illustrated implicitly (i.e. without any notation), allowing only one possible path, or explicitly using arrows directing the sequence. Additionally, the *Communication Flow* is used to illustrate the interaction between the specific participants of the service. It is typically represented while applying a standard sequence notation using arrows. If an explicit illustration is chosen for the *Action Flow*, both concepts can be linked to a global arrangement of elements and so be seen and treated as one unified flow across the diagram. Due to the resulting versatility of this latter approach, it shall be the basis for the transformation.

BPMN on the other side also supports the explicit display of the sequence flow between the different *Tasks*. With the application of *Gateways*, quite complex routing mechanisms can be illustrated. In addition to this concept, also the so-called *Message Flow* is utilized to display the exchange of information within a diagram. As discussed in the chapter targeting the nature of BPMN in general, the difference between these two elements and their applicability depends on the structure and setup of the whole diagram. While the *Sequence Flow* links the elements of a specific process, possibly encapsulated within a *Pool*, the *Message Flow* is used to transmit information between distinct processes (i.e. different *Pools*).

For the overall undertaking this means that basically, it is possible to transform the *Action and Communication Flow* of Service Blueprinting into the *Sequence and Message Flow* of BPMN as shown in Figure 4.3. This was also suggested during the comparison of the two concepts as discussed above [KMJ15]. However, this aspect is far from deterministic due to the way how the different concepts of BPMN are applied depending on the orchestration of the overall model. So in general it is possible, but strongly depends on the way how the transformation is used to create *Pools* and *Swimlanes* in the end. Looking at the chapter targeting the comparison, for this aspect the handling of the *Line*

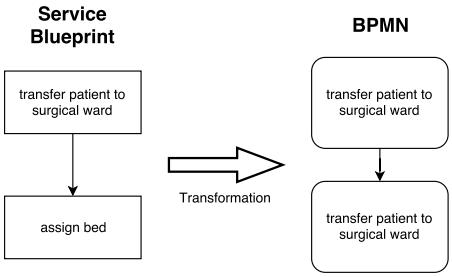


Figure 4.3: Transformation of the simple *Action and Communication Flow*

Separation and the resulting *Actor Categories* of Service Blueprinting will have a major impact.

Decision Point

When discussing the logical sequence of the *Actions* as done above, one has to consider the routing of the elements as well. As mentioned previously, if the implicit and therefore simple *Action Flow* is applied, this aspect can be neglected. But when taking a version as shown in the case study into account, at least some concepts targeting this issue need to be addressed.

The *Decision Point* is one of them and represents the splitting of the path into two or more branches. The performed selection is done similar to the logical *Exclusive OR* which means that exactly one path is chosen. The conditions that trigger the respective route are indicated at the outgoing flow notations (i.e. arrows) as it is common in flowcharting languages. Of course, due to the creation of alternative paths, there is the need for the illustration of synchronization points which are necessary to join the distinct routes back together. In contrast to more structured and flow-oriented languages, Service Blueprinting does not have the consideration of the token principle known from Petri Nets at heart. Although the derived assumptions still hold in general, the methodology is not built around the representation and satisfaction of each requirement. Therefore the topic of the synchronization of alternative paths is not explicitly addressed. When it comes to the graphical representation, this means that there is no special notation for this point. Rather it is illustrated using multiple incoming arrows for the receiving *Action*.

BPMN, on the other hand, supports quite complex routing mechanisms allowing the designer to create very comprehensive systems. Within the context of this methodology, the corresponding generic element is called *Gateway* and provides different semantic possibilities, as explained in the chapter targeting the nature of the language in general. One of them, the *Exclusive Gateway* is also intended to represent the logical *XOR-Operator*. Similar to the suggestion for Service Blueprinting, the conditions for the possible alternatives are indicated at the notation of the outgoing pathways. Looking at the merging of the alternatives, due to the process orientation of BPMN, the official

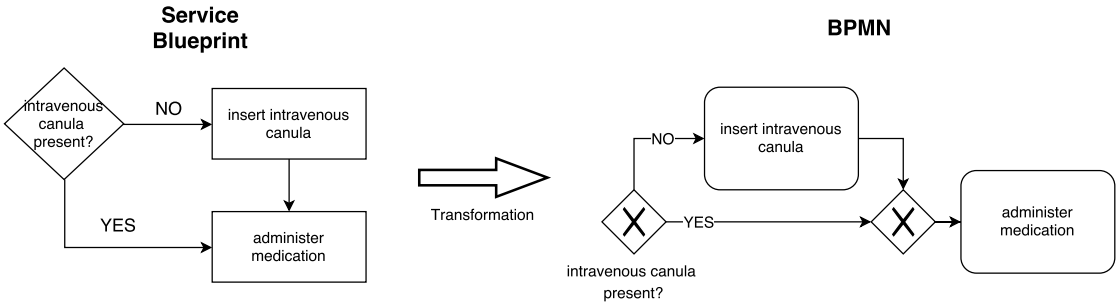


Figure 4.4: Transformation of *Decision Points*

standardization suggests the usage of an explicit notation [OMG11]. However, since only one path is taken, no explicit synchronization mechanisms is executed in the background.

Considering the transformation of this concept, it seems to be appropriate to match the *Decision Point* from Service Blueprinting with the *Exclusive Gateway* of BPMN as shown in Figure 4.4. Both mechanisms describe a similar logical behavior, which leads to this decision. One thing however that needs to be addressed, at the latest when thinking about a technical implementation, is the differences while handling the merging points of the alternative paths. While BPMN applies a specific notation to identify them, due to the simpler graphical representation, Service Blueprinting only uses multiple incoming edges for the next common *Action*. This may not look like a big issue, but when considering the possibility to use other splitting mechanisms for the service-oriented language as it was introduced within the literature [Sha10], it could be a problem. The reason for this lies within the different semantic functionalists inherited by the synchronization points depending on their nature. For example, there is a different process for joining paths of a parallel execution and one for an alternative one with exclusive conditions. If a token arrives at the synchronizing section, in the first case it waits until every other incoming edge provides a token on their own. In the other case, this is not necessary since there will never be a second token in this execution cycle.

Due to its practical implication, this topic shall be further discussed when it comes to the creation of the prototype later on.

Parallel Path

The next concept that shall be discussed at this point also belongs to the context of the routing mechanisms. As mentioned previously, besides allowing the creation of alternative paths, also the parallel execution can be added to the conceptual set of Service Blueprinting as done during the case study. Again, this allows the designer to create sequence flows that are to a degree more complex and may be useful to develop service models with a greater similarity to real-life situations.

While considering the notational set that was applied for the more abstract modeling approach within this thesis, in contrast to the *Decision Point* as explained above, for this

concept no explicit graphical representation is used. Instead, the splitting point within the *Action and Communication Flow* is indicated using multiple outgoing edges of the last common *Action*. However, the semantic behavior is quite similar to other modeling languages and the token concept known from Petri Nets is applied. This means that when the splitting point is reached, all outgoing paths are chosen and the contained subsequent *Actions* are executed. When it comes to the synchronization of the paths, similar to the *Decision Point* the next operation common to all *Parallel Paths* has multiple incoming edges. Although Service Blueprinting basically does not determine the behavior of the routing mechanisms to such a degree, this concept requires a quite clear specification at this point. To target this issue, once more the functionality of other modeling languages is used as a template and so when one token arrives at the synchronization point, it has to wait until all other incoming paths are finished as well.

Looking at BPMN, due to the comprehensive set of routing mechanisms, it also inherits the possibility to cope with the demand of parallel processing. Similar to the topic of *Decision Points*, the corresponding concept is once more represented by an element of the category *Gateway*. The so-called *Parallel Gateway* supports the same semantic behavior as it was complementarily defined for Service Blueprinting, which means that once arrived at the splitting point, all outgoing paths are executed. When it comes to the graphical representation, within the official standard of BPMN an explicit illustration is suggested using the diamond shapes known from general flowcharting in combination with a contained plus-sign. This notation is applied for the splitting as well as for the merging point. The functionality of the latter one is also quite similar to the systematics as described above. When a token enters the merging *Gateway*, it has to wait until further tokens arrive on all other incoming paths.

For the transformation of *Decision Points* from Service Blueprints to BPMN models, the identified mechanisms of both sides as described above seem to be appropriate. The transformation could be performed as shown in Figure 4.5. Considering the actual splitting point, although there is a mismatch when it comes to the nature of the graphical representation (explicit vs implicit), it should be possible to identify the corresponding positions within the Blueprint since the way how those are handled in the service-oriented language is only applied for these cases. But, as explained for the illustration of alternative ways, the merging constitutes a potential issue for the prototypical implementation later on. The concluding element of both concepts, *Decision Point* and *Parallel Path* consists of the usage of multiple incoming edges for the next common *Action*. This situation prevents the luxury that when such a point is identified in the model, it can be uniquely associated with the corresponding concept and the inherited synchronization semantics. So to overcome this issue, the synchronizing element has to be reviewed under consideration of the nature of the splitting point. Depending on this insight, the fitting BPMN element shall be selected.

The findings that will be made during the practical part of this thesis will show to which degree this matching process can be put in place.

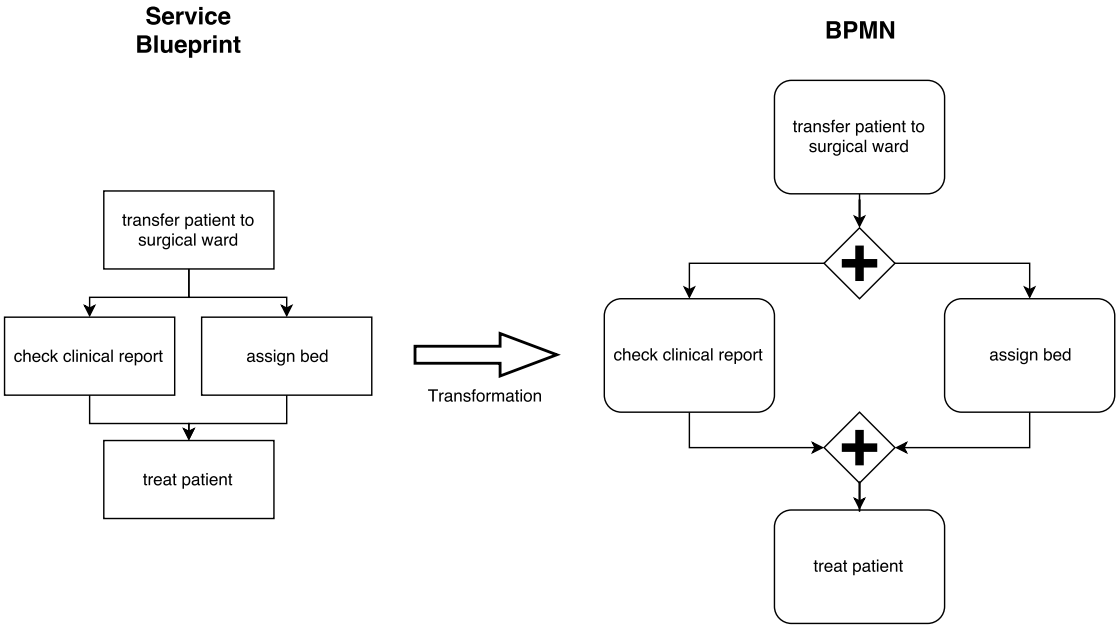


Figure 4.5: Transformation of *Parallel Paths*

Start- and End-Point

The aspects that shall be discussed at this point, can be considered as the last elements that are a direct representational part of the *Action and Communication Flow* of Service Blueprinting. The *Start-Point* and *End-Point* are very well known concepts in the domain of common flowcharting and are used to mark the beginning and the end of a specific sequence flow. Within the area of the customer-oriented modeling language, in the case of notational sets that are well suited for more abstract illustrations of services, among other elements, these points are only represented in an implicit way, without the use of corresponding symbols [LW11]. In these situations, it is simply assumed that the process starts with the first *Action* assigned to the customer. If the general purpose of the created model is targeting such a high-level representation (e.g. first drafts), this is an appropriate way to design the situation at hand. However, when it comes to more detailed and finer grained models, such an approach might not be enough and does not satisfy the demands concerning the accuracy. To overcome this issue, within the literature, the use of explicit *Start- and End-Points* has been suggested and applied within the case study of this thesis [Sha10]. At this point, it is important to note, that along with other elements of the flow notation as discussed above, these two concepts inherit semantic characteristics already known from other modeling languages. This means that the *Start-Point* consists within a graphical representation that has only outgoing and no incoming paths. Of course for the *End-Point* it is the other way around.

Due to the intentions of the Business Process Model and Notation, the notational set does suggest the use of an explicit indication of the beginning and the end of the process

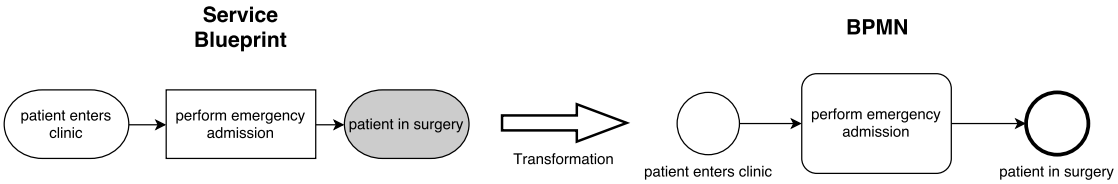


Figure 4.6: Transformation of *Start- and End-Point*

flow. The corresponding elements can be found within the category *Event* as explained during the chapter targeting the concept in general. As such, these points occur while considering the overall process flow and have an impact on the execution [OMG11]. Looking at the positioning of the elements which is relevant for the classification of *Events* in BPMN, of course, the *Start-Event* and the *End-Event* constitute the most basic representative of the start and end group.

For the transformation it should not be a big issue to match the *Start- and End-Points* of Service Blueprinting to the basic *Events* of BPMN as shown in Figure 4.6. However, a difficulty could lie in the handling of parallel paths in the service-oriented modeling language. As explained above, there is no explicit graphical representation and the splitting respectively the merging point is indicated while using multiple outgoing or incoming paths for the corresponding elements. In general, due to the lean definition of the Blueprinting approach, there is no restriction on which concepts it is allowed to apply the specific routing mechanisms. As a result, it is entirely possible to use a *Start-Point* to split the sequence flow into *Parallel Paths* using multiple outgoing edges. For BPMN this is not possible since the language requires specific notational elements to indicate these points in the diagram. So when it comes to the transformation, in such a case the procedure will have to identify the nature of the concept and generate the appropriate counterparts for the process model. An alternative approach would be to restrict the usage of *Parallel Paths* to the *Actions* of the Blueprint. To provide the designers of the models with as much freedom as possible during the phase of creation, of course, the first variant should be preferred. However, if this appears to be a greater issue during the prototypical implementation, the second solution could be applied.

Failure Points

Within the literature, the ability of Service Blueprinting to be used to analyze and evaluate service processes was mentioned as a key aspect of the overall methodology [Sho84]. To support this characteristic, the use of so-called *Failure Points* offers the possibility to indicate positions within the overall diagram that may be subject to errors that negatively influence the process execution and therefore the value as perceived by the customer [ZBG06]. Considering the service-oriented modeling approach, these points do not inherit any semantic characteristics and therefore do not influence the sequence flow in any way. Instead, the basic intention of this notational element is to mark situations that have to be addressed separately and constitute critical aspects when it comes to the

evaluation of the offer.

Targeting this issue, for its illustration different graphical notations have been suggested by various authors. For the purpose of the case study and in general for the context of this thesis, an approach has been chosen that links the detected *Failure Point* directly to the corresponding *Action* while placing a corresponding marker on the border of the operational element.

During the discussion of the matching between the two modeling methodologies, on the side of BPMN the so-called *Error Events* have been identified to serve a similar purpose when it comes to the indication of possible failure points within the diagram. However, possibly the biggest difference can be traced back to the observation that was already mentioned within the first paragraph of this section, the fact that *Failure Points* do not have an impact on the flow of the service itself. Due to the strong focus on the execution phase of the process-oriented language, this does not hold for BPMN. As a result, *Error Events* are elements that indicate positions within the overall diagram where an error might occur and change the sequence flow. To satisfy the language's demands concerning completeness and consistency, it is not enough to use this element as it is done with *Failure Points* in Service Blueprinting. To resolve such a situation it is necessary to either use the *Error Event* as the end of the process or to integrate a corresponding additional sequence of tasks.

When it comes to the core aspect of this thesis, of course, these aspects need to be taken into account. There are several ways how to perform the transformation while applying the insights as discussed above. First of all, considering a specific *Failure Point* a possible solution could be to simply use a boundary interrupting *Error Event* that is fixated on the boundary of the *Activity* which represents the corresponding context and is a matching-result illustrating the *Action* of the underlying Service Blueprint. This way, the BPMN model would also indicate the determined critical point within the process as shown in Figure 4.7. However, looking at the completeness and validity of the overall diagram, this would constitute a problem, since such an intermediate event needs to refer to a subsequent process flow. Nevertheless, this could be a possible solution if one considers the distinct levels of detail and abstraction of the two modeling concepts that was also subject to previous examinations. A BPMN model that is created as a result of such a transformation probably has to be reviewed and enhanced with more details to be able to satisfy all demands that are imposed on a modern process model. During this upcoming task, the responsible designer would have to think about how these situations should be treated in case the error occurs. Due to the lack of semantic representativeness of Service Blueprinting at this point and the fact that it might be necessary to create very individual resolutions for different error situations, it is not possible to offer a complete and satisfying result directly after the transformation.

But it might still be the intention to create a valid and at least a more syntactical complete BPMN model as a result of the transformation. In this case, what needs to be added during the revision afterward is the naming of the error case. The amendment of the solution sequence itself, although still advisable, is more of an optional undertaking.

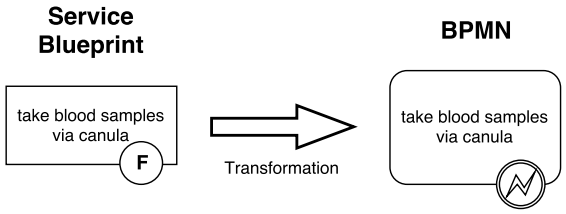


Figure 4.7: Transformation of *Failure Points* (boundary interrupting *Error Event*)

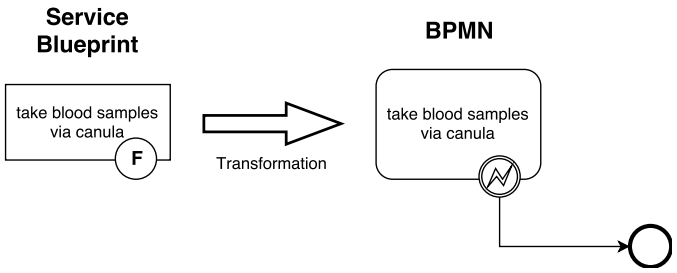


Figure 4.8: Transformation of *Failure Points* (boundary interrupting *Error Event* with process *End Event*)

At the same time, it should not result in a process orchestration consisting of a number of BPMN elements which require some time to be changed into the actual resolution of the critical point within the model. To do so, a possibility would be to connect the boundary interrupting *Error Event* with a process *End Event* via a simple sequence flow as shown in Figure 4.8. For the resulting process diagram this would mean, that in case of the failure's occurrence, the process has come to an end. If this does not correspond to the intention of the designer, which is very likely, he or she has simply to exchange the *End Event* with the actual resolution steps.

Another possible approach could be to design the default representation in a way that it does not terminate the process in case the failure occurs. To do so, after the boundary interrupting *Error Event* is triggered, the alternative process flow could be routed directly to an *Exclusive Gateway*. This element is used to merge the regular sequence with the exceptional one as shown in Figure 4.9. Of course, within this initial version, this constellation would only catch the failing operation, but the execution of the ongoing process would not be affected. Again, during the review of the transformation result, the designer should extend these points according to the requirements of the organization.

An additional option how to implement a generic compensatory sequence flow could be also to place the suggested merging *Exclusive Gateway* in front of the *Task* where the error might occur, as shown in Figure 4.10. In that case, a loop is triggered which causes the process to repeat this step until it concludes successfully. A possible downside would be that, if the thrown error cannot be resolved without additional operations, the execution is stuck in an endless loop.

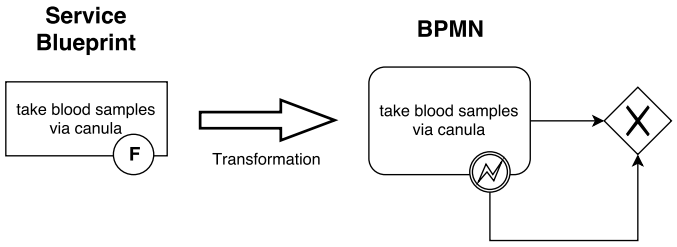


Figure 4.9: Transformation of *Failure Points* (boundary interrupting *Error Event* with catching behaviour)

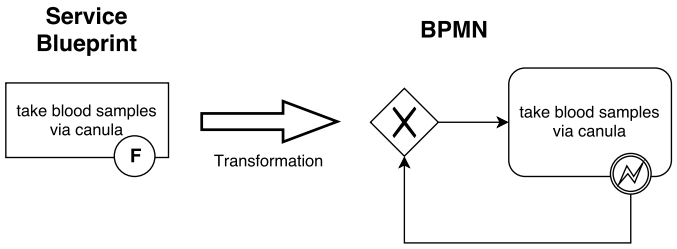


Figure 4.10: Transformation of *Failure Points* (boundary interrupting *Error Event* with looping behaviour)

Waiting Points

As mentioned above, within the context of Service Blueprinting, a *Waiting Point* is used to indicate a point in the service flow that may be subject to a frequent exceedance of the operation's time limit [LW11]. Within the context of the notational set as chosen for the case study of this thesis, a marker similar to the one proposed by Christopher Lovelock and Jochen Wirtz is placed on the border of a specific *Action* to be able to create a direct relation [LW11].

During the chapter targeting the matching of BPMN and Service Blueprinting, the observation was made that, due to the fact that the basic concept of *Waiting Points* and the occurrence of such a situation does not have an impact on the execution of the processes on its own, an appropriate BPMN counterpart cannot be identified.

As an implication for the intended transformation and because of the missing customer focus of the process modeling language, it would be possible to simply neglect this point. However, due to the importance of this aspect for the overall area of service design, creation and maintenance, it should still be a part of the concept that has to be developed at this point. Taking the informative nature of *Waiting Points* under consideration, the flow components as suggested by BPMN like *Events* for example, do not constitute a fitting match. One possibility however, could be to use BPMN *Annotations* to indicate the corresponding positions within process diagram as shown in Figure 4.11. This way the designer could decide to target such a critical aspect with additional compensatory

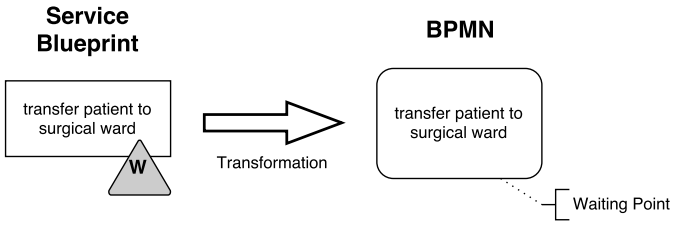


Figure 4.11: Transformation of *Waiting Points*

process flows.

Line Separation

To distinguish the different areas of a Service Blueprint, a *Line Separation* as discussed in the chapter targeting the conceptual set of the customer-centric modeling language is applied. It is used to create the actor specific categories where the individual *Actions* are allocated according to their nature. Additionally, it is used to differentiate between the illustration of the *Action and Communication Flow*, the *Physical Pieces of evidence* and IT elements. Each line is intended to represent certain characteristics that inherit a special relevance for the overall service model under consideration of the end-users demands.

With regard to this last statement, within their article targeting the comparison of the concepts by Kazemzadeh, Milton, and Johnson, they argue that due to this fact, it is necessary to consider each line and the resulting areas individually [KMJ15]. Of course, it would be possible to transform the graphical representation as applied in Service Blueprinting towards BPMN, but since the intention of this thesis is mainly to concentrate on the conceptual meaning, this would not suffice. On the contrary, because of the different characteristics, it might be necessary to use different concepts of BPMN to be able to cover all aspects.

To satisfy this demand, the following concepts shall be subject to the upcoming discussions targeting a possible transformation to the Business Process Model and Notation. Due to their relevance for the overall topic, *Tangible Objects* will be examined later on.

- Line of interaction
- Line of visibility
- Line of internal interaction
- Line of order penetration
- Line of implementation
- Line of internal IT interaction

BPMN, on the other hand, is of a more generic nature and hence does not offer such predefined categorization of its elements. Nevertheless, to support the structuring of created process models, the designer has the possibility to use *Pools* and *Swimlanes* as separating mechanisms. As explained before, while the first concept is applied to encapsulate different processes that exchange information via the BPMN *Message Flow*, the second one organizes one specific instance while linking its elements via the *Action Flow*.

To approach this issue, at the beginning it shall be discussed how the concepts of BPMN can be utilized to capture the line separation in general. Afterward, the focus will be on how the intended semantics of the different lines can be captured while using the suggested variants.

Possible Variants

When it comes to the transformation of the models between the two languages, due to the versatility and flexibility of the concepts of BPMN, there are several possibilities how the necessary matching can be performed. At this point, the findings of Jochen Meis, Philipp Menschner, and Jan Marco Leimeis are particularly interesting. As discussed within the chapter targeting possible extensions of Service Blueprinting, the authors wrote an article about how to use BPMN as a flowcharting notation for the service-oriented language. During their examinations, they also came across this problem and identified three variants of how the concepts *Pools* and *Swimlanes* can be used to reflect the *Line Separation* of Service Blueprinting. As a result, they summarized their findings with the following listing [MML10].

1. One *Pool* for the customer and one for the organization
2. One *Pool* per Blueprint area
3. Everything within one *Pool*

Looking at the first variant, the authors suggest to use two different *Pools* for the whole diagram. The first one represents the customer and therefore encapsulates all *Activities* and other flow elements that are directly executed or simply a part of his or her domain (e.g. automated customer tasks). The second *Pool* is applied to represent the organization and its corresponding process flows. To illustrate the *Line Separation* as a part of Service Blueprinting, *Swimlanes* are used to further divide the two process areas. As a result, the two major *Actor Categories* are explicitly captured while using two distinct processes. In accordance with the semantic restrictions of BPMN, the *Communication Flow* between them is reflected while applying the exchange of messages indicated either between the borders of the *Pools* or the contained *Activities*. In addition the actual routing of the processes displays the *Action Flow* [MML10].

Within the context of the second alternative, for each area that is the effect of the applied *Line Separation* of Service Blueprinting, an individual BPMN *Pool* is created. Again, in compliance with the corresponding semantic restrictions, this means that each logical

subdivision inherits its own independent process flow. Accordingly, the service-oriented *Action Flow* is exclusively represented while using the BPMN *Sequence Flow* and the *Communication Flow* while using the *Message Flow* linking the individual processes. This kind of illustration enables the designer to create a diagram with a strong focus on the distinction between the different *Line Separations*. Additionally *Swimlanes* can serve as organizational elements allowing the introduction of a complementary structure which helps to reach a greater level of detail when it comes to the classification of the contained tasks [MML10].

In contrast to the previous ones, for the third variant, the authors Meis, Menschner and Leimeis suggest using only one large BPMN *Pool* that contains the whole diagram. *Swimlanes* are then used to capture the Service Blueprint's *Line Separation*. The graphical display of the containing element is basically not necessary, although recommended for the sake of a better understanding. Considering the *Action and Communication Flow*, due to the illustration while only using *Swimlanes*, only one BPMN *Sequence Flow* is necessary. At the same time, a corresponding *Message Flow* is not applicable. Targeting a finer grained organization as discussed within the context of the second variant, it is possible to utilize nested *Swimlanes*, but with the downside of a loss of comprehensibility [MML10].

As a conclusion, within the context of their article, the authors examined the three versions under consideration of the overall case topic which is the *computerized support for services within the area of housing and living*. They came to the result, that for this domain of application especially the versions one and two are suitable. The third one however was found to be inferior due to the missing capability to explicitly display the classification of customer and author and the negligence of external and internal interfaces [MML10].

In general, while reviewing and comparing the three variants as explained above, it can be seen, that the usage of two distinct *Pools*, one for the customer and one for the organization, probably provides the clearest distinction between the two major groups of participants of a service. On the other hand, a downside exists in the characteristic that, due to the structure, there is no explicit illustration of the communication within the organization. On the contrary, every internal interaction between the participants is captured while using the *Sequence Flow* of BPMN.

When looking at the second variant which suggests to only use *Swimlanes* to represent the model structure, this strong grouping of tasks belonging to the domain of the customer respectively the organization is no longer present. But now there is a striving towards a more equal subdivision of the modeling canvas. Additionally, this allows to directly use the BPMN *Messages* to capture the *Communication Flow* and the BPMN *Sequence Flow* while targeting the *Action Flow*. Considering these observations, this second variant seems to correspond very well to the more traditional Service Blueprinting concepts as discussed in the literature.

The third and final version, that was outlined by Meis, Menschner, and Leimeis does not overcome the loss of the ability to clearly distinguish between the customer and the

organization to the degree as provided by version one. In addition, due to the missing utilization of BPMN *Pools* the flow of communication and information is not explicitly displayed within the resulting model at all. But despite these observations and although the authors do not recommend the usage of this option when looking only on the graphical representation, it probably constitutes the variant that is closest to a more flowchart oriented Service Blueprint as it was applied by Arash Shahin [Sha10] and during the case study of this work.

As stated above, to be able to further examine the characteristics as well as the advantages and disadvantages of the suggested possibilities, the following paragraphs shall be used to discuss the semantic nature of the individual lines that are a part of Service Blueprinting.

Line of interaction
The *Line of interaction* is used in Service Blueprinting to distinguish between the customer's *Actions* and the ones that are executed by the organization that provides the service at hand [BOM08]. The operations that are allocated to the area directly beneath this line are said to be front-stage activities and can be observed by the customer during the execution phase. When thinking about the transformation, the resulting model should be able to clearly represent the distinction when it comes to the association of a specific, contained *Action* respectively *Activity* to the correct agent (i.e. customer vs. organization). And in fact, while using BPMN *Pools* and *Swimlanes*, when applying the correct naming for the generic elements, this can be supported. Thinking about the desired clarity at this point, the version 1 as suggested by Meis, Menschner, and Leimeis [MML10] probably offers the best match concerning this issue. But also the other two options constitute valid representations. Another characteristic that has to be reviewed at this point is the visibility of the front-stage tasks to the customer. In Service Blueprinting this is indicated while assigning the corresponding *Actions* to the respective area of the modeling canvas. This is also possible with BPMN. Additionally, this relation can be illustrated while using the *Message Flow* to display the transmission of the information directly to the customer. Due to this suggestion, when going back to the three versions that were discussed above, option three is probably not the best choice since there cannot be a *Message Flow* due to the missing application of the BPMN *Pool* concept [MML10].

Line of visibility
Next on the list of separating lines is the *Line of visibility*. In Service Blueprinting it is applied to distinguish between the organization's *Actions* that are visible to the customer and the ones that are not [BOM08]. In accordance with this definition, the area above this line is called front-stage (or onstage) and the one directly below back-stage. Despite the characteristic that the elements that belong to this second area are not directly visible to the customer, they are executed as a direct response to the customer's demands and requirements during the specific situation. Again, during the transformation, it should be sufficient to correctly allocate a specific area within BPMN and assign the respective flow elements. Additionally, the semantic linkage as suggested for the front-stage tasks should be applied. Hence, the absence of a direct relation between the elements of this domain

and the customer's area can be used to illustrate these circumstances. As suggested above, the BPMN *Message Flow* could contribute to the validity of this aspect. So again, the last version discussed by Meis, Menschner, and Leimeis does come with a drawback concerning this point [MML10].

Line of internal interaction

The *Line of internal interaction* is used to further differentiate the organization's *Actions* within a Service Blueprint. The basic intention of this concept is to distinguish between elements that are a response to the current service instance and those that are necessary to be able to provide the intended offering in the first place [LW11]. So these *Actions* are of a more supportive nature which is also a term often used to refer to the containing area beneath the line. At this point, it is important to note that these elements have a direct relation to the customer's demands and therefore the current service case. If one considers the general topic of the *Line Separation* as a hierarchical structure, these elements also belong to the category of back-stage operations and are therefore not directly visible to the customer. Due to this affiliation, basically the same semantic requirements are applicable while considering the transformation. Additionally, it would be a viable intention to strive towards the illustration of the independence of such tasks within the overall modeling approach. For BPMN this would require decoupling the corresponding sequence flows. When looking at the different versions as discussed above, only the second option is able to cover this extended demand since mainly pools are used to create its structural organization.

Line of order penetration

The area containing the supportive *Actions* can be further partitioned while applying the so-called *Line of order penetration*. Using this kind of conceptual element, the designer has the possibility to introduce an additional distinction depending on the impact of the contained *Actions*. While the supportive elements above this line are contributing to the service value as perceived by the customer (e.g. sharpening the razors) the ones beneath the separator are of a more general nature (e.g. managerial activities). This causes the situation that the latter ones are basically independent of the customer. Along with this observation, these elements do not provide a strong relation to the other areas of the Service Blueprint [FK04]. At this point Kazemzadeh, Milton and Johnson assess that BPMN might lack the possibility to back the semantic characteristics of this area beneath the *Line of order penetration*. And in fact, despite the possibility to simply use a *Pool* or *Swimlane* to encapsulate the corresponding operations and sequence flows, due to the inherited independence, there is not a clear, advisable approach to capture the intentions behind this concept. One problem that could be a cause of this issue is the possible distinction when it comes to the level of abstractions of the *Actions* above and beneath this line. Since the elements below are of a more general nature, they might not be captured with all the details as used for the rest of the service flow. Additionally, at this point it should be mentioned that when thinking about an explicit illustration of the *Action and Communication Flow* as shown within the case study and the notational set that constitutes the basis for the intended transformation, the more

general supportive *Actions* beneath the *Line of order penetration* should not be a part of the sequence of operations that are directly linked to the value generation. Instead, the contained elements should reflect their independent nature while standing separately and only provide a linkage to the main elements to display them as potential preconditions. To transform this concept towards the process-oriented language, because of the fact that this kind of orchestration is not possible with BPMN, it could be an option to decouple this area from the rest of the model and use a separate diagram representation. Another attempt to capture this point in BPMN would be to simply use *Annotations* to indicate those *Activities* within the diagram that need to have such supportive actions as a general precondition. Of course this way the elements are not a direct component of the diagram flow itself, but this might not be necessary since it already has more general characteristics in the first place. This last statement can also be used to argue for the negligence of this area beneath the line during the transformation. Due to the fact that the contained *Actions* are not directly connected to the sequence flow, they can be seen as not being a part of the core process that is responsible for the value generation and should be captured using BPMN.

Line of implementation

The next line is called the *Line of implementation* and is used to further divide the area located beneath the *Line of order penetration*. The *Actions* located above are intended to be generally supportive operations that are executed as a preparation of the processes responsible for the value generation [Wit04]. But due to their nature, they are independent of the current execution of the service process itself and therefore constitute a separate sequence flow. The *Actions* beneath this separator also inherit the same characteristics, but with the distinction that they do not target the value generation and are intended to constitute preceding managerial activities. Due to the common basic properties, the *Line of implementation* also suffers the same fate as the previous line when it comes to the transformation towards BPMN. Since the concepts are independent of the execution of the actual service process, it is not possible to integrate the elements into the corresponding process diagram targeting the value generation, without violating the overall purpose of these areas. To approach this point, the way how the *Line of order penetration* is handled during the transformation has to be taken into account. If the solution is to use a separate BPMN diagram to capture the more general supportive *Actions*, it can also be used to integrate the *Line of implementation*. To do so, the three possibilities as discussed by by Meis, Menschner and Leimeis should be considered [MML10]. Since there is not the necessity to model the customer as an actor participating in these processes, the first and the third version would produce a similar result where everything is contained in one BPMN *Pool* and the Blueprint areas are illustrated using *Swimlanes*. However, this fact has no impact on the second version and each Blueprint line that should be a part of this complementary diagram would represent a separate process contained within a distinct *Pool*. Both variants constitute viable solutions. In contrast to this suggestion, the use of BPMN *Annotations* was also discussed as another possibility. But due to the situation that the target concept lacks scalability, it might be difficult to capture multiple supportive areas in a structured way. Furthermore,

depending on the number of the corresponding *Activities*, this might negatively influence the comprehensibility of the resulting BPMN model. As a last possibility, of course, the negligence of this line can also be a solution.

Line of internal IT interaction
The last separation that shall be considered while designing the possible transformation is the *Line of internal IT interaction*. It was discussed by Christopher Lovelock and Jochen Wirtz to be able to target the linkage of modern service orchestrations and the applied information technology [LW11]. To do so, the area that is created at the bottom of the Blueprinting canvas contains representations of the applied systems and IT components. So in contrast to the line separations as mentioned above, at this point, not *Actions* are modeled but more static elements. As a result, this issue is different from the other areas. To target this topic, among others, BPMN offers the possibility to capture *Data Stores* within the process diagram. They are represented by a specific notation and located somewhere on the BPMN canvas. Herein it is not necessary for these elements to be placed in a specific conceptual area (i.e. *Pool* or *Swimlane*) since it is possible for distinct processes to access the same data collections. For the purpose of the transformation, the systems contained within the area beneath the *Line of internal IT interaction* may be represented while using these BPMN *Data Stores*. An additional *Swimlane* or *Pool* should not be necessary.

Physical Evidence

The aspect that shall be discussed at this point is a very important part of a service and as a result, also of Service Blueprinting. *Physical Evidence* is the conceptual and notational representation of tangible elements and objects that are observable by the customer during the execution phase of the service. Examples would be the received bill, the waiting room of the dentist or a certificate of a training event. As mentioned in a previous chapter, the importance is a consequence of their role in the overall offer and the potential impact on the perceived value. When it comes to the creational process of a Blueprint, the allocation and integration of *Physical Evidence* is one of the final steps. It is represented using the top lane on the modeling canvas.

Looking at the work of Kazemzadeh, Milton and Johnson, during their conceptual comparison the authors came to the conclusion, that for this aspect a process-oriented counterpart from the Business Process Model and Notation cannot be found [KMJ15].

Now, when it comes to the transformation of Service Blueprinting towards BPMN, due to the importance of this component for the topic of service design in general, a viable solution could be relevant for the expressiveness of the result. But while attempting to create a plausible representation, several critical aspects have to be discussed along the way.

First of all, it has to be considered, that BPMN basically serves a quite different purpose then Service Blueprinting and therefore has a distinct point of view at the processes at hand. When it comes to the topic of *Physical Evidence*, for the service orientation

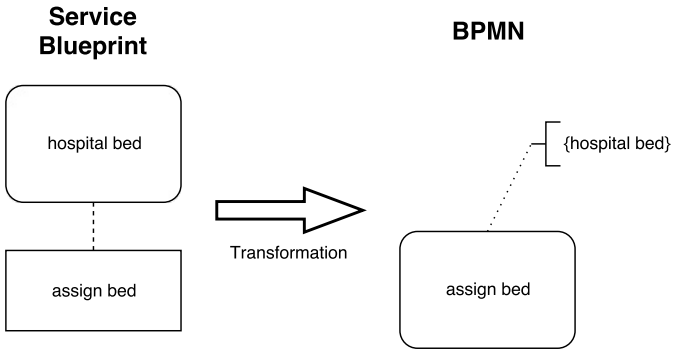


Figure 4.12: Transformation of *Tangible Objects*

the corresponding elements are subject to the domain of the customer (i.e. perceived by the customer) and therefore situated on top of the area representing the specific *Actions*. When thinking about BPMN, the correct position within the overall process has to be determined. Considering the representation using Service Blueprinting and the notational set that was applied during the case study, for this purpose the relation to the *Actions* defining the context of their occurrence has been indicated. But it might also be interesting for the process illustration to indicate other steps that are related to these aspects in general. This constitutes another point where a possible extension of the resulting BPMN model has to be performed after the actual transformation.

Another issue within this context is, of course, the graphical representation of *Physical Evidence* in the final process model. Since there seems to be no appropriate notation available, some alternatives have to be developed. At this point there are two possibilities. The first one was already suggested for other concepts and consists within the recycling of present BPMN functionalities. Due to the strict specification of the process-oriented language, only two existing elements can be taken into account. Once more, one way could be to use simple text *Annotations* to indicate the corresponding tangible objects. However, looking at the processability of the final model, there is the downside that such elements are harder to identify. After all, among others, *Annotations* are used to place notes and additional information on the canvas to improve the readability of the diagram. Additionally, this BPMN concept was also a suggestion for other Blueprint components where a counterpart could not be found. To support the applicability of this element, the designer would have to think of a way to differentiate *Annotations* representing tangible objects from other cases. One possibility how this could be done can be seen in Figure 4.12.

The second existing element of BPMN that could be used are *Data Objects*. Within the modeling methodology itself, the notation is used to represent relevant data that is generated and flows through the process [OMG11]. One similarity that supports the usage is that like *Physical Evidence*, *Data Objects* inherit a more static and kind of tangible nature. Although both concepts can be very important for the respective model, considering the corresponding sequence flows they can be seen as an accompaniment.

However, there are a few arguments against the usage of *Data Objects* to represent *Physical Evidence*. First of all, probably the most obvious one is the fact that in contrast to tangible objects as they are used for a Service Blueprint, the BPMN elements are actually not of a physical nature. They are tightly linked to the aspects of the information technology, that is necessary in order to perform the activities of the process. Furthermore, in BPMN so-called *Data Associations*, represented as dotted arrows, are used to link *Data Objects* to specific *Activities* and indicate if it is used as an input or an output. This way, the designer has the possibility to illustrate the flow of the data. This level of detail is not supported by *Physical Evidence* of Service Blueprints.

Considering these differences, if *Data Objects* are used to represent *Physical Evidence* later on, the specification of BPMN as explained within the official standard targeting these elements would not hold. As a conclusion, this matching cannot be recommended for the transformation of the concepts. At his point, a potential resolution could be found in an extension of the concepts currently present within BPMN version 2.0. A new element would need to be introduced that reflects the static nature of *Physical Evidence* and is linked to the *Activity* that describes the context of their occurrence within the process. Due to the nature of tangible objects within the context of service design, a flow similar to the one as applied for *Data Objects* would not be necessary.

Although this solution seems applicable it does come with a quite obvious, but great disadvantage. As mentioned before, the Business Process Model and Notation is a very well known and widely adopted approach towards the design of process orchestrations. Over the time various software developers have created their own solutions to support the modeling of new processes. The unified XML background structure that is part of BPMN version 2.0 supports the general applicability of a model created with a tool following this standard. Considering the transformation, thinking ahead to its potential result, the outcome is probably represented using a BPMN conform XML file. If the official standard is extended to be able to capture *Physical Evidence*, the generated diagram cannot be interpreted by common BPMN editors supporting version 2.0 anymore. As a consequence, the ability of the models to be easily communicated to other entities and automatically interpreted using the corresponding software solutions is affected.

Further Enhancements

The last topic that shall be approached while discussing the possibilities for a transformation from Service Blueprinting towards the Business Process Model and Notation, concerns potential enhancements that could be introduced to either side of the matching.

Within this context, of course, an enhancement consists within the addition of a new or the amendment of an already existing concept. When looking at the service modeling methodology, one might find it difficult to determine the actual set of basic components and their exact implementation in the first place. Thinking about the simpler versions, the conceptual catalog as used for the case study already provides several enhancements which were also considered during the design of the transformation above. Examples are

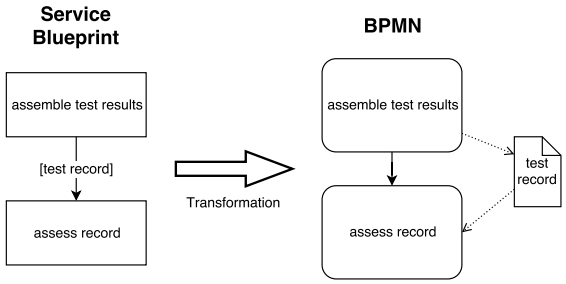


Figure 4.13: Transformation of the *Data Flow*

the explicit illustration of the *Action and Communication Flow*, more enhanced routing mechanisms (i.e. *Parallel Paths* and *Decision Points*), the linkage of *Failure and Waiting Points* to specific *Actions*, as well as the integration of additional layers like the *Line of internal IT interaction*.

Nevertheless, there is always room for improvement or at least some aspects that could be added to the methodology to be able to provide a more comprehensive way to aid the overall undertaking. One example could be to strengthen the focus on the information technology, which is a critical component of modern services. In addition to the illustration of IT systems as suggested by Christopher Lovelock and Jochen Wirtz [LW11], it could positively contribute to capturing not only these static elements but also data objects similar to BPMN. Up until now, a more common extension in this direction is to use tangible objects to represent such digital elements that are visible to the customer. However, it could be a viable and useful extension to also illustrate data artifacts that are of a more dynamic process nature (e.g. for internal processes) and, of course, their flow through the service instance. Under consideration of the simplicity of Service Blueprinting, the way how this additional information is stored in the diagram has to be rather intuitive. The suggestion at this point is to simply use text annotations directly associated with the *Action and Communication Flow* as shown in Figure 4.13. The applied bracketing is used to identify the information artifact as being an instance of the data flow throughout the model. When it comes to the transformation, the logical counterpart consists, of course, within the usage of BPMN *Data Objects* in combination with *Data Associations* illustrating the information flow in the process diagram. Thinking about the usability of this aspect during the creation phase of a service, the application of this additional concept should remain optional.

Another point where the conceptual set could be extended is the introduction of the ability to further specify the involved *Actions*. Currently, there is only one basic element which is then assigned to the context of its occurrence using the *Line Separation*. For the expressiveness and the clarity of the model, it could be beneficial to apply different elements of the category *Action*. Examples could be the indication of operations that require a certain type of knowledge (e.g. IT tasks) or if it is a standardized task or one that strongly depends on the customer and the situation at hand (i.e. reflecting the degree of necessary customization).

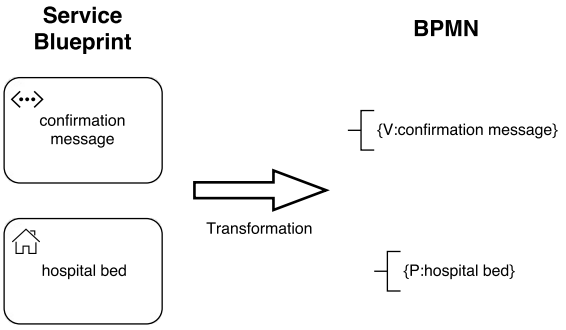


Figure 4.14: Transformation of the *Evidence Category*

Although it might be tempting to integrate this aspect into the conceptual set of Service Blueprinting, one should not neglect the core purpose of the language and the circumstances of its application. The methodology is intended to constitute a tool to be used already during the early stages of a development process. As discussed in a previous chapter, this does also require a certain creativity. In this context, the popularity of Service Blueprinting can also be traced back to the simplicity of its concepts and, in general, the inherent utility in these cases. Therefore, the usage of a broader selection of different *Actions* cannot be recommended.

However, the basic idea as proposed above could also be applied to another concept, where such a finer-grained distinction may positively contribute to the overall process. In compliance with the demand to facilitate the integration of information technology into the domain of Service Blueprinting, at this point, the extension of the concept targeting tangible objects is suggested. Again, an additional imposed overhead regarding the complexity of the modeling approach should be avoided. To do so, only the two major groupings *Physical* and *Virtual* shall be optionally supported within this context. Examples would be an actual tangible object like the hospital bed and a digital confirmation message for an order. Thinking about the transformation, above three different approaches have been suggested. When it comes to the usage of *Data Objects* or *Annotations* it would be possible to indicate the group affiliation of the identified elements with a simple lettering as shown in Figure 4.14 for *Annotations*.

One enhancement that imposes only a minor conceptual complication, but might be very helpful to cope with the potential complexity of the overall case, could be the introduction of a marker indicating critical *Actions*. These should be treated in a separate Blueprint or more complex process diagram later on. In such a situation a drill-down could be applied to be able to focus on these points without considering the rest of the service flow all the time. Thinking about the transformation, as a possible counterpart, BPMN *Tasks* containing the sub-process marker could be used as shown in Figure 4.15.

When looking at the main purpose of Service Blueprinting, another aspect could be introduced to enhance the modeling capabilities within this context. Due to the strong customer focus, it might be a good idea to explicitly address the end-user's expectations

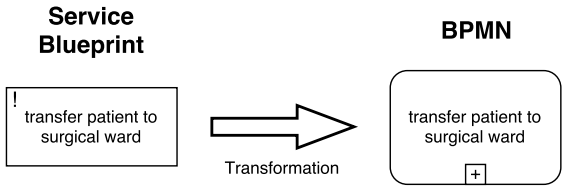


Figure 4.15: Transformation of *Specification Points*

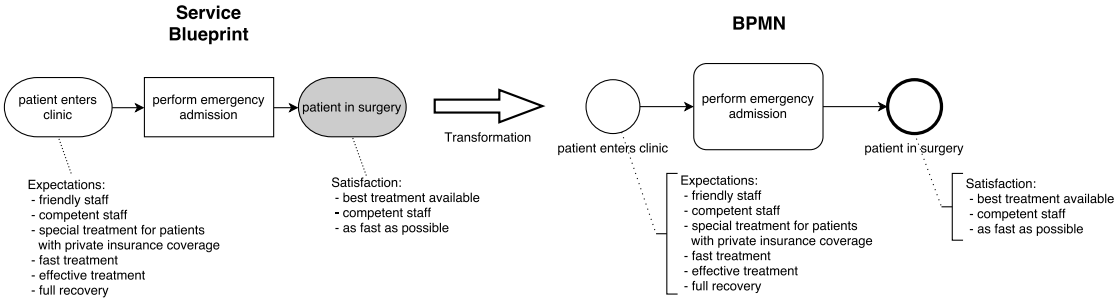


Figure 4.16: Transformation of *Expectations and Satisfactions*

and the final outcome of the diagram. Doing so would enable the designer to use this information to check which imposed requirements can be satisfied and which potentially result in deviations that negatively influence the offering's value as perceived by the customer. Considering the service-oriented language, the indication could be done by applying a listing to the *Start- and End-Point* of the model as shown in Figure 4.16. The interpretation of this information is, of course, completely up to the designer. Due to this more informative nature of the elements, when it comes to the Business Process Model and Notation, a logical counterpart would be the usage of *Annotations*. Once more it is necessary to be able to identify these notes later on within the resulting model.

Looking at the other side of the transformation, in contrast to Service Blueprinting, the Business Process Model and Notation does have a specific set of concepts. Due to the purpose of the language, it offers quite comprehensive possibilities to capture the structure and the flow of a specific process. Thinking about possible extensions and enhancements, considering the topic of this thesis, possibly the greatest shortcoming of the methodology is that it is not possible to increase the focus on the customer's perspective of the overall situation. This can also be observed when it comes to the actual design of the transformation. One issue within this context that was already mentioned above is the transformation and illustration of *Physical Evidence*. But as discussed previously, the compatibility of the resulting models with other entities, other models, and the available tool support, leads to the conclusion, that an extension of the official standard is not advisable at this point.

Summary

At this point, possible variants have been discussed how an existing Service Blueprint can be transformed into the Business Process Model and Notation. To get a clear overview, the individual approaches targeting the conceptual set of the service-oriented language are shown in Table 4.1. These listed options also represent the basis for the upcoming final decision and the prototypical implementation later on.

4.2 The Recommendation

This section is intended to constitute the conclusion of the conceptual work while targeting the transformation. The goal is to come to a decision about the actual approach while considering the different options displayed in Table 4.1. To do so, concerning the conceptual elements of Service Blueprinting where multiple variants have been pointed out, a clear recommendation towards the preferred approach will be made. Along the way, of course, it is necessary to justify the motivation behind the result. Within this context, also the requirements imposed by the starting point and the preconditions as discussed at the beginning of the chapter have to be taken into account.

Looking at the provided overview in Table 4.1, for the conceptual elements *Action*, *Action and Communication Flow*, *Decision Point*, *Parallel Path*, *Start-Point*, *End-Point*, *Waiting Point*, *Data Flow*, *Specification Point*, *Expectations and Satisfactions* and the *Line of internal IT interaction* a clear recommendation for the transformation has already been made and discussed within the previous section.

The *Failure Point* on the other hand, is an element where several possibilities have been identified that could be applied during the transformation. As a logical counterpart, the BPMN boundary interrupting *Error Event* has been suggested. But this concept alone does not satisfy the functional requirements of a BPMN model without an additional compensatory sequence flow. The first suggestion was to simply neglect this aspect and leave the specific implementation up to the model designer who is responsible for the completion of the resulting BPMN diagram afterward. Although this has to be done in any case, this option does come with the downside, that the generated BPMN model is not valid. To overcome this issue the other three variants propose a generic compensatory flow. The first one uses an end node to indicate that the process stops if the corresponding error occurs. The remaining two approaches apply a combination with a merging *Exclusive Gateway*, so that either the process flow continues even if the failure occurs or the process step is repeated until it is successful. Of course, no option is best suited for every case since the actual error handling has to be implemented depending on the specific situation at hand. However, at this point, the usage of an *End-Event* shall be suggested. As mentioned above, it overcomes the issue of a functional incomplete solution. Furthermore, it seems not advisable to (automatically) create a model where the occurrence of a defined error does not have an impact on the rest of the execution phase (as it is the case with the catching behavior). Additionally, when considering

the looping compensation, there is the risk that if the error cannot be resolved without additional *Tasks*, the process flow would be stuck.

The next topic where multiple possible variants have been identified concerns the *Line Separation* of Service Blueprinting in general. The decision that will be made does also reflect on the implementation of all model areas that are in direct relation with the service value as perceived by the customer (i.e. *Line of interaction*, *Line of visibility* and *Line of internal interaction*). At this point, the findings of Jochen Meis, Philipp Menschner, and Jan Marco Leimeis were considered which led to the suggestion of the three variants *One Pool for the customer and one for the organization*, *One Pool per Blueprint area* and *Everything within one Pool* [MML10]. Basically all approaches inherit specific characteristics, advantages, and disadvantages and may be viable solutions for the transformation. However, for the recommendation in this context, the first variant should be preferred. The main motivation consists within the fact that this version supports the clearest distinction between the customer's area and the one of the organization. This is an important aspect due to the specific focus of the service-oriented language, which results in the situation that the model references these two major participants during the execution phase.

In contrast to the Blueprinting areas directly responsible for the perceived value of the service, the ones created while applying the *Line of order penetration* and the *Line of implementation* have to be considered separately due to this very reason. During the previous discussions, to handle these two concepts, again three variants have been proposed. The first one consists within the transformation into a distinct BPMN diagram which should reflect the independence of these aspects. Another approach is to use text annotations that are applied to the *Tasks* contained within the main model to indicate the dependencies towards the supportive operations. Last but not least, the negligence of these concepts was also discussed since the contained elements are not part of the process at hand and therefore are not directly related to the main sequence flow. As a recommendation, the first variant shall be applied using one BPMN *Pool* with two *Lanes* in accordance with the chosen approach for the overall *Line Separation*. In general, it does not seem to be a good idea to omit details while thinking about a transformation into a modeling environment which inherits a lower level of abstraction. To overcome the issue concerning the differences in nature of the specific areas, the usage of *Annotations* would also be a plausible solution. However, there are two arguments against it. First of all, using the textual element in BPMN causes the diagram to lose the logical sequence of the more general operations among each other (i.e. in BPMN there is no sequence flow for *Annotations*) and their dependencies. Furthermore, this textual counterpart is also a recommendation for other points of the transformation and it is not advisable to exaggerate its use since would be prone to ambiguity afterward.

The last decision that has to be made at this point concerns the transformation of physical evidence. Their impact on the overall topic of service design and the perceived value of a service was already emphasized before. Now to process this aspect, it would be possible to use *Annotations*, *Data Objects* or to simply neglect it since it does not correspond very

well to the present concepts of the process-oriented modeling approach. But because of the importance of this issue, it is not advisable to skip the transformation. However, using *Data Objects* as suggested, has the downside that an existing BPMN concept is applied to a context that does not hold while considering the specification contained in the official standard provided by the Object Management Group. This may induce misunderstandings when it comes to the sharing and communication of the model and in the worst case to the invalidity of the result. These arguments lead to the decision to use *Annotations* for the application of tangible objects in BPMN. Following up on this suggestion, the indication of the proposed *Evidence Category* as shown in Figure 4.14 can be applied.

When applying the recommendations as suggested above, and highlighted in Table 4.1, each element of the extended conceptual set of Service Blueprinting is allocated to a logical counterpart on the side of the Business Process Model and Notation. This way it should be possible to perform the transformation, with regard to the discussed requirements as planned in the context of this thesis. Considering this theoretical approach, the next step should consist of the attempt to prove that this undertaking can actually be done. To do so, the next chapter will describe the prototypical implementation of the transformation while using the conceptual mapping as shown in the provided table.

Service Blueprint	Source	Target	Standard BPMN v2.0
Action	■	➡	(simple) Task
Action and Communcication Flow	■	➡	Sequence and Message Flow
Decision Point	■	➡	Exclusive Gateway
Parallel Path	■	➡	Parallel Gateway
Start-Point	■	➡	Start-Event (untyped)
End-Point	■	➡	End-Event (untyped)
Failure Point	■	⇨	boundary interrupting Error Event
		➡	boundary interrupting Error Event with process End-Event
		⇨	boundary interrupting Error Event with catching behaviour
		⇨	boundary interrupting Error Event with looping behaviour
Waiting Point	■	➡	Annotation
Line Separation (general)	■	➡	One Pool for the customer and one for the organization
		⇨	One Pool per Blueprint area
		⇨	Everything within one Pool
Line of interaction	■	➡	see Line Separation (general)
Line of visibility	■	➡	see Line Separation (general)
Line of internal interaction	■	➡	see Line Separation (general)
Line of order penetration	■	➡	separate BPMN diagram
		⇨	Annotation
		⇨	No Transformation
Line of implementation	■	➡	see Line of order penetration
Line of internal IT interaction	■	➡	Data Stores
Physical Evidence	■	➡	Annotation
		⇨	Data Objects
		⇨	No Transformation
Data Flow	■	➡	Data Flow
Specification Point	■	➡	Sub-Process Marker
Expectations and Satisfactions	■	➡	Annotation

Table 4.1: Overview transformation possibilities

CHAPTER 5

The Prototypical Implementation

This chapter is dedicated to the description of the prototypical implementation which was developed as proof that it is, in fact, possible to automatically transform an existing Service Blueprint into the Business Process Model and Notation. For this purpose, the suggested representational counterparts that were elaborated within the previous chapter will be used as a basis and a guideline. Accordingly, the final output consists of a model that is a direct result of the theoretical considerations that were discussed before.

After a brief statement summarizing the main motivation, the used technology including their functionality shall be explained. Afterward, the overall development process, as well as, the final architecture of the prototype will be discussed. As important aspects for this topic, the issues concerning the transformation of the *Action and Communication Flow* and the creation of the graphical model representation will be addressed separately. In a final section, the actual transformation process shall be explained step-by-step.

5.1 Purpose and initial Approach

The way how a transformation from Service Blueprinting towards BPMN could be performed was the main subject of the previous chapter. As a result, a selection of transformation rules as shown in Table 4.1 was presented and discussed. This constitutes the theoretical basis that is necessary to approach the bridging between the distinct modeling worlds. But despite the given reasoning, it cannot be considered as valid on its own neither for the thesis, nor the general issue at hand.

The reason for this is simply the fact that such a hypothesis still needs to be examined in accordance with its feasibility. Now to satisfy this demand, of course, there is a multitude of different ways how this can be done depending on the topic. For the current context, a suitable solution is to use the bundled transformation rules and build up a prototypical implementation around them. The resulting piece of software should then be able to interpret an existing Service Blueprint and automatically create a valid BPMN structure representing the specific case. If this undertaking is successful, it has been shown that the former theoretical proposition is viable.

Thinking about this suggestion, the basic components of the transformation environments and therefore the abstract features of its architecture can be identified. Of course, first of all there has to be some tool-set that enables the user to capture a service idea using the Blueprinting mechanism in a structured way. This is a necessary prerequisite due to the requirement that the result of this very first step has to be interpreted for further processing.

Once this is done, the actual conversion mechanism has to read in the new model and identify the core elements and structures. Afterward, while applying the transformation rules, as discussed above, the designed service offering should be converted into a corresponding BPMN model, which can then be written into a matching BPMN file structure. The final goal should consist of a resulting template that can be further processed while using other, complementary tool-sets and frameworks that are compliant with the BPMN 2.0 methodology.

Summarizing this first layout, while developing the prototypical implementation the following components have to be addressed and integrated into the final framework. During the upcoming section, the respective details will be examined in more detail. This shall then constitute the basis for the decision concerning the technologies used for the actual implementation.

1. Service Blueprint Modeling Environment
2. Transformation Mechanism
3. Creation of BPMN Files

5.2 Technology

At this point, the three main components that are necessary for the final prototype have been identified. But, of course, these aspects still inherit a very high level of abstraction and therefore need to be analyzed in more detail and finally matched by choosing a corresponding technology that satisfies the imposed demands. When thinking about the upcoming elaboration, a logical initial approach is to move along the direction as required by the transformation process itself, which is also illustrated in the listing above.

5.2.1 Service Blueprint Modeling Environment

So to begin with, the first functional element that needs to be considered is about the way how the model representation of the service is created and stored for further treatment. While taking the main goal into account, which describes the attempt to integrate the modeling of services into an IT environment which enables the automated processing of its results, it seems a logical conclusion to think about a computerized development mechanism.

Now as it is often the case, facing such a situation, one has the decision to either create the required tool support or look for already available alternatives. Of course, the first option has the advantage that while developing a new solution, it can be tailored to the specific needs. On the other hand, when using an already existing one, there is the benefit that such a mechanism might be already accepted by some users and integrated within different development environments. Furthermore, especially when it comes to the domain of graphical modeling editors, the creation from scratch is not an easy task to pull off. One has to take the main purpose of the planned undertaking into account and decide if this is justifiable. But these considerations may be unnecessary if there exists no suitable solution in the first place. So while targeting this aspect of the overall architecture, the initial step was to look for some already available possibilities.

Realtime Board

During the corresponding research, the first application that was found is the Service Blueprinting tool presented by *Realtime Board* [Rea]. It offers an easy-to-use model editor that is completely accessible via their webpage. While applying several graphical shapes that can also be individualized using different colors, the user has the possibility to create simple Service Blueprints very fast. The explicit illustration of the *Action and Communication Flow* as a demand stated during the previous sections can also be integrated. Another feature that shall be mentioned at this point, is the possibility to easily share created models with other participants and so introduce a more collaborative working environment. However, a downside consists of the variety of the graphical elements that can be combined to capture the service at hand. While it is theoretically possible to assemble several shapes to form individual ones, it does not seem practicable. The tool's main purpose is rather to offer a modeling canvas for the creation of representations inheriting a higher level of abstraction. A possible analogy is the consideration as a

digital whiteboard. Despite the fact, that it feels very good to use for such a use-case when thinking about the notational set that was chosen within the theoretical chapters of this thesis, it does not support the required logical complexity. Furthermore, due to the strong focus on the graphical representation, the functional dependencies between the elements, are not supported to the degree that is necessary. Another aspect that is of a rather vital interest for the overall undertaking is the possibility of the tool to export the created models. Concerning this point, the environment offers several formats like PDF, images, Jira attachments, or a CSV export. When considering the fact that it should be possible to process the results in an automated fashion, only the last one seems to be feasible in general. But, although the data format basically would be very convenient to be interpreted as required, the details that are contained within the file are not sufficient. It simply offers a listing of the element names placed on the Service Blueprint without representing further logical or graphical details of the model in general.

Canvanizer

Another possibility that was identified while searching for an already existing tool support targeting Service Blueprinting was the *Service Blueprint Canvas* available at *canvanizer.com* [Can]. In general, they offer an online modeling area that is built upon the extensive usage of predefined templates. Besides well-known ones like the *Business Model Canvas*, some are specifically targeting the domain of service design like the *Customer Journey Canvas,* the *Feedback Canvas* or the *Service Blueprint Canvas.* Of course, this last one is especially interesting at this point. Due to the strong focus of the environment on the usage of predefined templates, for this version, a given modeling canvas with a fixed selection of Blueprinting Areas is presented. Additional ones cannot be added. In general, the system has a more abstract approach when it comes to the modeling process itself. In contrast to the tool that is offered by *Realtime Board*, the *Canvanizer* does not allow the user to place elements by their own choosing. For each new *Action* a predefined area exists within a specific lane which makes it rather easy for users without any knowledge about modeling languages to create new Blueprints. But, of course, the downside can be found within an inherent loss of the freedom to create customized service flows that are necessary to capture more complex situations. The explicit illustration of the *Action and Communication Flow*, along with the use of other shapes besides *Actions* is not possible. Due to this strong limitation, this tool does not seem appropriate for the planned undertaking.

Draw.io

At this point during the research targeting the aspects of the planned transformation environment, once more the missing standardization of the general methodology Service Blueprinting and the lack of an extensive tool support as a side effect is very evident. Besides the two applications which where mentioned above, no comparable tool has been identified. As mentioned within the introductory paragraphs, a next step would be to develop a customized tool that fulfills the desired purpose. But before doing so, one

might also consider other already existing tools that do not support Service Blueprinting on their own, but offer some kind of extensibility that might allow the creation of a customized modeling setup.

Draw.io has been identified to be such a modeling tool, which enables the user to choose from an extensive selection of different shapes (e.g. BMPN, UML, Entity Relation) and even allows the definition of customized ones. The tool is a free to use modeling environment that can be accessed in several ways. One can either visit the webpage *https://www.draw.io*, download the offline clients available for *ChromeOS*, *Windows*, *Linux* and *macOS*, use the existing *Google Drive* or *OneDrive* integration, install the corresponding *Google Chrome* extension, or utilize the available *Jira* and *Confluence* extension (payed version) [Dra].

The way how a user interacts with the system is more oriented on straight forward modeling tools. The main area is of course the modeling canvas itself and is positioned in the middle of the screen. The different graphical shapes are situated in the left-side toolbar. Each element can be applied with drag and drop and linked while using the connecting arrows that occur when hovering over a specific shape with the cursor. Additionally, it is possible to add a new linkage similar to other shapes when selecting it from the left side of the screen. If a graphical element is selected on the canvas, the right side of the screen can be used to specify the style of the shape and the contained text as well as attributes targeting the positioning of the element on the canvas (e.g. rotation, size) [Dra].

Now the characteristic of the tool that makes it especially interesting for the problem at hand, is the fact that it allows the user to combine every shape available and also store individual collections of elements as customized libraries. In contrast to other more purpose driven modeling tools like the *Camunda Modeler* for BPMN, in *Draw.io* there are no functional restrictions governing the combination possibilities of the elements and their context. This makes it possible to reuse concepts and notations from very different domains and create sets tailored to special use cases [Dra].

However one essential feature is still missing that represents an important prerequisite for the applicability of the tool within the environment of the modeling and transformation of Service Blueprints. As examined for the previous tools, the possibility to export the created representations has to be possible in a way that enables the automatic processing afterwards. Looking at *Draw.io* in addition to formats like PDF, jpg and png the user can also create a HTML and XML export. Especially the latter one is interesting since it offers a typical structured representation that inherits the potential to be automatically interpreted later on (in its uncompressed form) using a suitable application [Dra].

The characteristics described above are assessed to be sufficient to use *Draw.io* for the first attempt to build up a prototypical modeling environment that is compatible to the potential transformation towards BPMN. For this purpose the next section will discuss a corresponding library that can be integrated into the modeling tool.

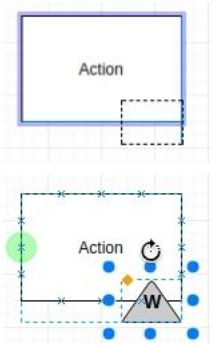


Figure 5.1: Element grouping in *Draw.io*

Draw.io - Service Blueprinting Library

After analyzing the tool according to the high-level demands imposed on a Service Blueprinting mechanism that shall be used within the context of the planned conversion, the upcoming paragraphs will target a corresponding library including the resulting XML export structure for *Draw.io*.

To begin with, the first step has to be the creation of the modeling concepts within the chosen tool. To do so, as mentioned above, a new library has to be put in place to offer a predefined set of available notational and functional concepts to the end users. But before this can be done, a decision has to be made, which elements shall be a member of this new environment. Of course, this issue was already discussed to some extent within the chapter targeting the theoretical aspects of the transformation. As a result, the goal at this point has to be the development of the modeling capacity to capture all concepts subject to these preceding examinations within the framework of *Draw.io*. The left side of Table 4.1 can be considered as an overview of all elements that need to be covered.

Some of the concepts for the intended Service Blueprinting, like *Failure and Decision Points* consist of multiple different shapes. Instead of providing already finalized elements for such cases, the individual components can be selected separately. After the correct placement of the symbols, a grouping mechanism takes care of the functional coherence, as shown in Figure 5.1. The palette can be combined to represent the concepts as illustrated in Figure 5.2 and was also used for the model snippets presented within previous chapters.

A short guide how the library works with *Draw.io* can be found in Appendix B.

Draw.io - Service Blueprinting XML Export

Once the Service Blueprint is created, it needs to be extracted to be able to perform the intended transformation. As mentioned previously, targeting this aspect *Draw.io* offers an XML export functionality that has the potential to enable this conversion.

To do so, the export mechanism has to be triggered in its uncompressed version, which then contains the complete graphical and logical information related to each element. For

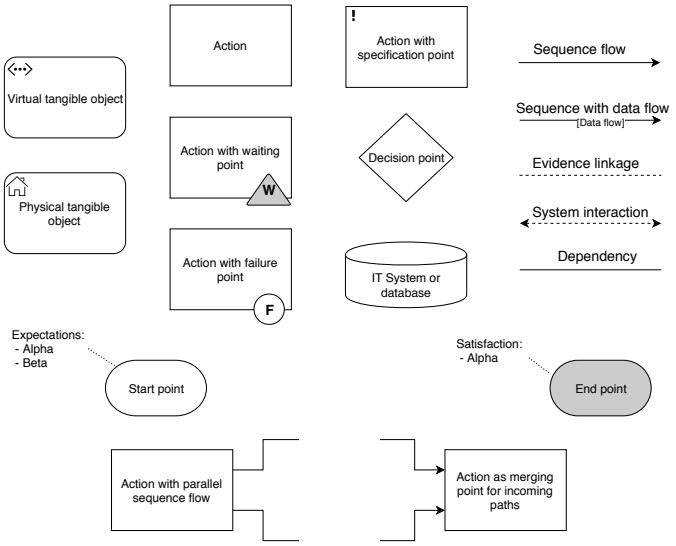


Figure 5.2: Service Blueprint concepts in the customized *Draw.io* library

the model structure the *mxGraph* library that is based on *Java Script* is used [JGr18]. Herein each specific object is captured while using a *<mxCell>* tag that contains the logical characteristics as attributes as well as the geometry data (i.e. width, height, X coordinate, Y coordinate) represented using a *<mxGeometry>* tag as a child element. As it is common for such structures, each instance has an assigned id attribute that is used as a reference for the source and the target of model edges and for all other functional relations. In accordance with this logic, the association of a specific shape to a parent element as it is the case for *Actions* and the containing *Areas*, is stored while using the id of the parent element within a corresponding attribute associated to the children.

At this point, another interesting feature of *Draw.io* can be used to aid the interpretation of the resulting model structure afterward. The main question concerning this aspect is of course how the information contained within the file can be correctly processed. This means that there is the need to have some logic that allows an automated extraction of the Service Blueprint components including their interdependencies. To handle this issue, while assembling a new library, the tool offers the ability to edit the style attribute of the resulting *<mxCell>* tag. This makes it possible to add new information to each element that can then be used to identify the corresponding Service Blueprinting concept within the XML file. For this purpose, each component of the library contains a specific *sbType* element.

The following XML code shows the Service Blueprint representation of a simple *Action* without additional information (i.e. *Failure and Waiting Point*). A more comprehensive example can be found in Appendix D.

```
<mxCell id="2fc9635e5d0fdac0-8" parent="1"
    style="sbType=area_front_stage_interactions;swimlane;html=1;
```

```
        horizontal=0;swimlaneFillColor=#CCFFCC;swimlaneLine=0;
        align=center;verticalAlign=middle;labelPosition=center;
        verticalLabelPosition=middle;glass=0;fillColor=#99FF99;
        rounded=0;comic=0;labelBackgroundColor=none;
        strokeColor=#000000;strokeWidth=1;fontSize=14;
        fontColor=#000000;" value="Front-Stage Interactions"
        vertex="1">
        <mxGeometry as="geometry" height="230" width="720"
            x="10" y="200"/>
</mxCell>
<mxCell connectable="0" id="2fc9635e5d0fdac0-9"
        parent="2fc9635e5d0fdac0-8"
        style="sbType=action_container;shape=action_container;
        strokeColor=none;fillColor=none;rounded=0;comic=0;
        labelBackgroundColor=none;fontSize=14;fontColor=#000000;
        align=center;html=1;" value="" vertex="1">
            <mxGeometry as="geometry" height="55" width="120"
                x="210" y="88"/>
</mxCell>
<mxCell id="2fc9635e5d0fdac0-10" parent="2fc9635e5d0fdac0-9"
        style="sbType=action;shape=action;rounded=0;html=1;
        whiteSpace=wrap;align=center;labelBackgroundColor=none;
        strokeColor=#000000;strokeWidth=1;fillColor=#FFFFFF;
        gradientColor=none;fontSize=12;fontColor=#000000;comic=0;"
        value="prepare medication" vertex="1">
        <mxGeometry as="geometry" height="55" width="120"/>
</mxCell>
```

The code segment contains three *<mxCell>* tag elements which means that it represents as much different model components. Each of them has an assigned style attribute that contains the mentioned *sbType* that is used to identify the corresponding role within the Service Blueprint. Accordingly, the first component is of type *area_front_stage* and therefore represents the *Front-Stage Area* of the Service Blueprint. The remaining two both refer to a simple *Action*. While the element marked with *sbType=action* is actually representing the core element including the contained textual description (i.e. *value="prepare medication"*), the other one corresponds to the grouping mechanism that is used to cohere the underlying *Action* with its possible parameters. The resulting *action_container* can be considered as the top level element when it comes to this category of flow element and therefore also contains the reference to the containing area with its parent attribute and the graphical location on the canvas (i.e. x and y coordinates).

Using this modeling and export functionality of *Draw.io* as a starting point, the next examinations can be made towards the technology that could be used to interpret the generated files, transform them towards a structure conform to BPMN and write the result into another XML file that follows the specification of the official BPMN 2.0 standard.

5.2.2 Transformation Mechanism

At this point, one can assume that it is possible to extract Service Blueprint XML files from a computerized modeling environment, which have the potential to be processed using additional mechanisms. The functionality that should be the main subject within this context is, of course, the transformation towards BPMN.

The technology for accomplishing the previous step was chosen while relying on an already existing tool that can be configured to fulfill the imposed task. However, a similar approach at this point is futile simply due to the nature of the objective. Since there is no comparable tool that provides the transformation of such a Service Blueprint XML structure towards BPMN, the second available option has to be chosen, which consists within the development of a customized systematic.

To do so, a programming language and an environment have to be selected that suits the purpose. *Java* (in its current version 1.9) in combination with the *Eclipse IDE* (version *Oxygen*) was assessed to be a sufficient basis for the intended development process. Besides the fact that the language provides all necessary functional characteristics (e.g. XML processing capacity), during the research targeting the general aspects of the transformation mechanism, other third-party frameworks that operate within the context of BPMN were found to be based on the same foundation. Examples are the well known BPMN 2.0 platform *Activiti* [Sof18], the *Camunda* environment [Ser18b] and of course the *Eclipse BPMN Modeler* [Fou17]. Although this might not contribute to the current task, it has the potential to aid the generation of a valid BPMN 2.0 XML file during the final step of the conversion and in general the compatibility of the resulting program with other frameworks.

To be able to develop the necessary code segments in an appropriate way, the technical setup is complemented by the use of *Git* via *Bitbucket* [Atl18] and *Apache Maven* [Fou18]. The first one is a version management system that allows the storage and synchronization of different versions of the code and so enables the developer to work in a more efficient way. The second one is a build tool that is used to take care of the dependencies towards other, external libraries of the final program.

Following the assumption that the development environment which was outlined within this subsection is sufficient to develop the core aspect of the prototype, the next considerations can be made while targeting the last step towards a valid BPMN file.

5.2.3 Creation of BPMN Files

Until now, the technical basis was specified that shall provide a modeling environment which enables the end-users to create a Service Blueprint while using a computerized modeling mechanism, hand over the result to the conversion functionality, which then shall transform the interpreted source model towards BPMN 2.0. What is still missing concerns the necessary basis for the generation of *.bpmn* files conform to the definition stated in the official documentation of the standard.

The .bpmn File Structure

Looking at the target structure, after the initial *definitions* tag, there are three top-level elements as shown within the following code snippet (based on an example file created while using the *Camunda Modeler*) [Ser18a]. At this point, it shall be noted that the following remarks are intended to provide a rough overview of the contained elements. A more comprehensive example can be found in Appendix D. For a detailed explanation of the file structure please have a look at the official documentation [OMG11].

```
<?xml version="1.0" encoding="UTF-8"?>
<bpmn:definitions
    xmlns:bpmn="http://www.omg.org/spec/BPMN/20100524/MODEL"
    xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI"
    xmlns:di="http://www.omg.org/spec/DD/20100524/DI"
    xmlns:dc="http://www.omg.org/spec/DD/20100524/DC"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    id="Definitions_1"
    targetNamespace="http://bpmn.io/schema/bpmn"
    exporter="Camunda Modeler"
    exporterVersion="1.11.3">
    <bpmn:collaboration id="Collaboration_1p6mp31">
        <bpmn:participant id="Participant_1wrbcch"
            name="customer" processRef="Process_1" />
        ...
    </bpmn:collaboration>

    <bpmn:process id="Process_1" >
        <bpmn:startEvent id="StartEvent_16q9z5y"
            name="enter clinic">
            <bpmn:outgoing>SequenceFlow_0nzim1q</bpmn:outgoing>
        </bpmn:startEvent>
        <bpmn:task id="Task_1ts9d7s" name="express discomforts">
            <bpmn:incoming>SequenceFlow_0nzim1q</bpmn:incoming>
            <bpmn:outgoing>SequenceFlow_0w7iooi</bpmn:outgoing>
        </bpmn:task>
        <bpmn:sequenceFlow id="SequenceFlow_0nzim1q"
            sourceRef="StartEvent_16q9z5y"
            targetRef="Task_1ts9d7s" />
        ...
    </bpmn:process>
    ...

    <bpmndi:BPMNDiagram id="BPMNDiagram_1">
        <bpmndi:BPMNShape id="StartEvent_0ee7l0w_di"
            bpmnElement="StartEvent_16q9z5y">
            <dc:Bounds x="205" y="581" width="36" height="36" />
            <bpmndi:BPMNLabel>
                <dc:Bounds x="222.0" y="620.0" width="0"
                    height="12" />
            </bpmndi:BPMNLabel>
        </bpmndi:BPMNShape>
        <bpmndi:BPMNEdge id="SequenceFlow_0nzim1q_di"
            bpmnElement="SequenceFlow_0nzim1q">
```

```
            <di:waypoint xsi:type="dc:Point" x="241" y="599" />
            <di:waypoint xsi:type="dc:Point" x="310" y="599" />
            <bpmndi:BPMNLabel>
                <dc:Bounds x="275.5" y="578" width="0"
                    height="12" />
            </bpmndi:BPMNLabel>
        </bpmndi:BPMNEdge>
        ...
    </bpmndi:BPMNDiagram>
</bpmn:definitions>
```

The first section is used to capture the participants of the model. For the situation at hand which consists within the representation of a service offering, this tag contains one child for the customer and one for the organization, since the decision was made to illustrate them while using distinct processes. If it is the case that also *Preparative and/or Managerial Actions* are a part of the source model, a third participant is added. Furthermore, the grouping is used to encapsulate the interactions that occur between the participants of the model (i.e. the BPMN message flow).

The next one, the *process* tag, of course, represents a specific process of the current collaboration. As it is probably the case within the context of the intended transformation, there are multiple participants and this tag may occur multiple times. Each participant references to a specific process via the attribute *processRef*. Contained inside the *process* tag, the assortment of XML elements referring to the logical structure of the current process can be found. This includes information about the lane separation (if present) as well as the typical flow components like *Tasks*, *Events*, *Gateways* and of course the *Sequence Flow*. As shown within the given code snippet, in contrast to the XML structure representing the Service Blueprint model, a connecting *Sequence Flow* with its source and target is not only captured using the corresponding tag alone but also as child elements associated to the linked elements.

The final top-level element concerns the assignment of the graphical details necessary to render the BPMN diagram within a compatible model editor. For this purpose, each object that needs to be a part of the final illustration is required to have an entry mirroring the coordinates, the width, and the height. This includes all flow elements as well as the participants and the lanes. For the interpreter to be able to correctly assign the shape type (e.g. circle for events) during the recreation phase, each shape element is linked to the corresponding logical counterpart contained within the *process* tags using the *bpmnElement* attribute.

Generating .bpmn Files

Now in the context of this section, the objective is to come up with a base technology that enables the final transformation environment to export the converted models into the file definition that was outlined above.

Once more, a possibility would be to develop a new mechanism from scratch and so have a solution tailored to the specific needs of the problem statement. But naturally, due to the strong dependency of this task on the widely adopted standardization of BPMN 2.0 and the current relation of existing frameworks (i.e. *Activiti*, *Camunda*) to the chosen programming language *Java*, it might prove beneficial to look into already available alternatives.

Very soon after starting the corresponding research, a solution was found within the Model-API offered by *Camunda* [Ser15a]. It is a lightweight *Java* library that, once integrated, offers the possibility to interpret existing *.bpmn* files and create a customized BPMN structure as a *Java* object (*org.camunda.bpm.model.bpmn.bpmnModelInstance*) including all necessary sub-elements and flow notations. As indicated within the corresponding official documentation, up until now, not all concepts available in BPMN 2.0 are supported. But after a short review, it was assessed that basically all necessary items needed within the context of the transformation are already present and therefore, within this regard, the extension should suffice.

But of course, the main issue at this point is to create the required *.bpmn* files. This can be considered as one of the key features of the *Camunda Model-API*. It offers a method that takes care of the whole XML processing part in accordance with the target file structure. As a prerequisite a valid *Java* BPMN representation is necessary, which either can be achieved by interpreting and amending an already existing *.bpmn* file or creating a new one from scratch. Targeting the processability of the model that is passed onto the XML creation, within the context of the *Camunda* framework and therefore the *Camunda Model-API* as well, the environment offers a built-in validation mechanism that can easily be applied as a part of one's individual development. If so, this has to be done shortly after the finalization of the *Java* BPMN object. The function uses the BPMN 2.0 specification to control the conformity of the new model and throws exceptions accordingly [Ser15b].

However, this kind of validation does not say anything about the correctness of the resulting model in terms of representing the targeted service. Concerning this point, basically there are two main possibilities. First of all, it does not seem advisable to do so while reviewing the *Java* BPMN object structure since there are still some steps that need to be performed until the final result is achieved. So it has to be the generated *.bpmn* XML file. Now of course it would be possible to check the background structure, but, although this might be not a bad idea to do so during the initial steps of the development process, when it comes to more comprehensive service models, such a file can inherit a considerable length and therefore would proof to be complicated and very time-consuming to check. To overcome this limitation, a logical step would be to use a graphical editor that is capable to read in the newly generated files. This is also a very relevant step from a general point of view of the suggested transformation concept. Since the ultimate target is to use a created Service Blueprint model to automatically generate a BPMN template structure that can then be used to add more details and finalize the process model, the development of the prototype should go as far as possible towards the step

where this final amendment will be done. Thinking about how people work within the modeling domain in general, it would be practicable to do so in the most convenient way, which probably is such a graphical editor.

For this purpose, to complete the topic concerning the base technology for the development, also a graphical modeling environment should be selected that is capable of interpreting the output diagrams afterward. For this purpose, another element of the *Camunda* BPMN environment can be utilized. The *Camunda Modeler* is a rather straightforward and easy-to-use BPMN editor which stores the created files within the discussed *.bpmn* format [Ser15a]. Therefore it can theoretically also be used to interpret files that were created while using other mechanisms like the intended transformation.

After this definition of the last elements that shall be a part of the final development environment, the actual creation of the transformation mechanism and the complementary functionalities can be initialized.

5.3 Development Process of the Transformation Environment

The following paragraphs shall provide a brief overview of the performed development process targeting the setup and creation of the transformation environment. Each step is based on the technology that was outlined above and can be seen as being a part of the rough outline that was suggested in the very first section of this chapter. An impression of the actual layout concerning the final components of the transformation shall be given within an upcoming section.

This last statement leads to the starting point of the actual work. Since, while considering the intended mechanism, it can be observed that each subsequent step is heavily relying on the previous one. For example, the transformation part towards BPMN can only be designed effectively if a tested and validated form of the programmatic interpretation of a Service Blueprint is already present. Since such a functionality does not already exist within this context, this can only be the case if the development process is completed to a significant degree. As a logical conclusion, the overall execution was chosen to be aligned with the direction of the transformation. So the first step towards the final solution has to be the consideration of the computerized creation of Service Blueprints using the chosen technology, the *Draw.io* modeler.

To begin with, the target was to create a method within the boundaries of the chosen environment that enables the computerized creation of Service Blueprint models. To do so, as discussed in the previous chapter, a new model library for *Draw.io* was created including all necessary conceptual elements that were also a part of the theoretical elaborations. This functional set was developed and enhanced during the creation of a number of test models as well as the diagrams illustrating the case study as shown within Appendix A and discussed in the corresponding chapter. Of course, along the way of the further development steps, frequent amendments have been made which results in

the libraries final form as shown above. But for the time being, after the finalization of the first test models, the next step, the interpretation of the exported Service Blueprint XML files can be approached.

Taking the given structure as a starting point, the basic task is the one of straightforward XML parsing[1]. For this purpose a *Java* DOM parser was used to read in the specific files and store the identified Service Blueprint details within the corresponding *Java* objects. Of course, the elements that are a part of this last aspect need to be created first and so before the actual interpretation logic can be developed, the static Service Blueprint *Java* representation has to be implemented. More details about the final layout can be found in the chapter targeting the architecture of the prototype. Once this is finished, the algorithm for reading Service Blueprint XML files can be set up. To begin with, a simple functionality was put in place that can be used to recursively interpret each node including its children. Along the way, subsequently more details and cases were added which, in their final constellation, enable the interpretation and storage of all conceptual elements and orchestrations that are offered and supported by the predefined Service Blueprint *Draw.io* library.

At this point, the already available components enable the creation of a new Service Blueprint while using *Draw.io*, the interpretation of the corresponding XML export and the storage of the identified information within an appropriate *Java* object structure. The next step has to be the processing of these created data elements and the conversion into a format that can then be passed on to the creation of the BPMN files. As stated during the elaboration of the technical components, the *Java* object structure for the BPMN part is provided by the *Camunda Model-API*. So what has to be done next is the development of the transformation steps between the customized Service Blueprint *Java* objects and the *Camunda BpmnModelInstance*.

The official documentation of the *Camunda Model-API* provides a couple of very simple code snippets that show how BPMN elements can be created while using *Java* and so serves as a starting point [Ser15a]. Thinking back to the XML representation of the *.bpmn* files, two major parts have to be considered when it comes to their creation. First of all, a BPMN model needs to have a valid logical structure. This is sufficient if the generated processes only have to be treated while using additional automated functionalities. But when it comes to the interpretation with a graphical editor that enables a more convenient way for the users to amend the transformation results, also the diagram information has to be present within the file. Sadly no possibility has been found that takes care of the creation of the complementary graphical information and so while transforming an existing Service Blueprint model, at the same time the logical, as well as the graphical details, have to be created and assigned to the BPMN *Java* structure. This tends to make the overall process to a significant degree more complex since when thinking about the transformation rules as discussed previously, a simple one-on-one conversion is not possible and the resulting model will inherit a more complex

[1] a very good and comprehensible example can be found on the following website *https://www.mkyong.com/java/how-to-read-xml-file-in-java-dom-parser/*

layout than the original Service Blueprint (i.e. BPMN does require more details and more specific elements). Especially during the case wise transformation of the *Action and Communication Flow* the development proved to be anything but an easy task. Due to the complexity of this sub-topic, the details will be discussed in a separate chapter later on.

To cope with this situation, at first, the development efforts were concentrated on the creation of the logical file components. To be able to check the validity of the results, in total about 70 different test models were created. This was done, not only by using *Draw.io* for the Service Blueprinting part but also while utilizing the *Camunda Modeler*, which made it possible to have a valid BPMN background structure fitting the specific cases. This way a template was available that could be compared against the prototypical results of the transformation. Of course, without the graphical details contained inside the generated files, the provided editor could not be used for the validating steps. Instead, at first for the smaller test cases, the background XML structure was compared to the template that was created using the *Camunda Modeler*. It shall be noted that at this point, due to the usage of the built-in validation mechanism provided by the *Camunda Model-API* it has been ensured that the generated results were already compliant with the BPMN 2.0 specification.

Once the creation of the logical structure was implemented and found to be tested to a satisfying degree while using this first approach, the decision was made to go on to the graphical counterpart. As discussed above, due to the different nature and level of detail of the two modeling methodologies, this step inherits a great degree of complexity on its own (accordingly it will be discussed in an upcoming section in more detail). Although it would be theoretically sufficient to have a valid logical BPMN structure to show the transformation capabilities, due to the implications it was found to be an important final step to guarantee the validity and usability of the solution. So once implemented, the newly generated file can be viewed when using a graphical editor capable to interpret the *.bpmn* XML format. For testing purposes mainly the *Camunda Modeler*, but also the *Yaoqiang BPMN Editor*[2] has been used to carry out the final tests.

As indicated above, due to their importance and impact on the development process and therefore the final form of the transformation mechanism, the handling of the *Action and Communication Flow*, as well as the creation of the graphical elements, will be discussed in more detail within the upcoming sections.

5.4 Transformation of the Action and Communication Flow

The following paragraphs shall be used to elaborate the transformation of the *Action and Communication Flow* of a Service Blueprint. This sub-topic is addressed separately since it has great implications for the development as discussed within the previous section

[2] can be downloaded under *https://sourceforge.net/projects/bpmn/*

and in general on the way how a given service model is transformed into the BPMN. Furthermore, this chapter can also be seen as being complementary to the theoretical transformation rules.

The starting point for this discussion can be found when looking at the decision made concerning the transformation of the lane separations in general. During the theoretical elaborations, the following three possibilities outlined by Jochen Meis, Philipp Menschner and Jan Marco Leimeis as presented in their article were listed as potential solutions [MML10].

1. One *Pool* for the customer and one for the organization
2. One *Pool* per Blueprint area
3. Everything within one *Pool*

As indicated in the final recommendation, the first variant was chosen due to the clear separation of the two main domains, which forms an important characteristic of Service Blueprinting in general.

As a consequence of this decision, the resulting BPMN model inherits at least two different processes. If *Preparative or Managerial Actions* are present within the source model, a third one is added. However, while applying the agreed flow notations for Service Blueprinting, in the initial diagram only one flow across the whole canvas is illustrated. So the main goal within this context is to split this unified sequence and create the correct representations inside the corresponding BPMN pools. Although this basically sounds rather straightforward, it does have implications for various aspects of the resulting model, which shall be discussed within the following sub-sections.

5.4.1 Simple Flow across different Processes

The first characteristic of this point targeting the transformation has to be the way how a simple flow from the customer to the organization or vice versa is resolved when it comes to the conversion towards BPMN.

Let us assume a very common situation where the *Action and Communication Flow* of the Service Blueprint goes across the *Line of Interaction* between the customer and the organization as shown in Figure 5.3.

Now when it comes to the representation while using BPMN, this cannot be done in the same way since it is impossible for the BPMN *Sequence Flow* to cross the borders of the pool that contains the current process. The concept that is offered by the official standard to capture the communication between the participants of different domains is the application of the *Message Flow*. However, this is not tightly linked to the *Sequence Flow* on its own, but by an appropriate usage of *Messaging Tasks* or *Messaging Events* that are applied to propagate the overall flow over the whole diagram and coordinate the distinct processes.

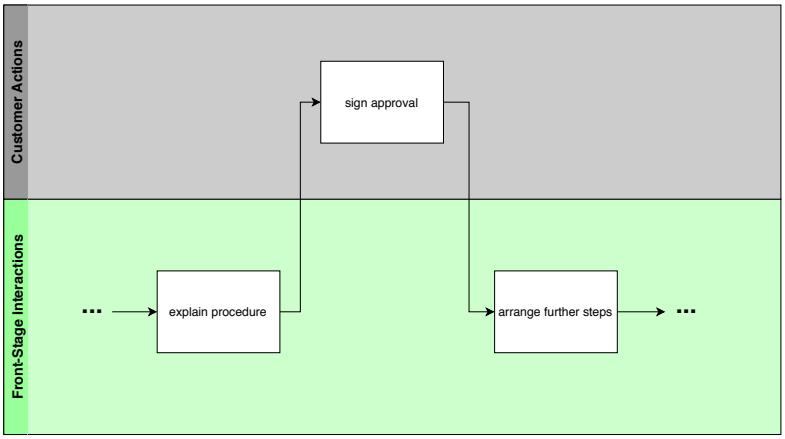


Figure 5.3: Simple service flow across different processes in Service Blueprinting

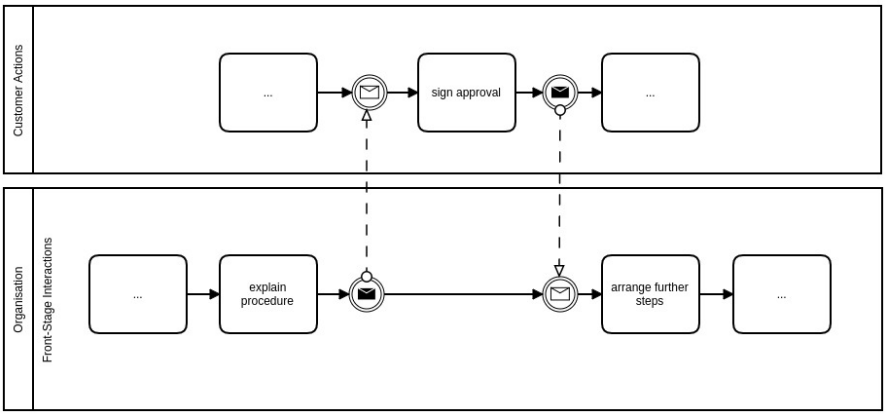


Figure 5.4: Simple service flow across different processes in BPMN

Looking at the transformation rules as applied for the prototype, the decision was made to use *Message Throw* and *Message Catch Events* for this purpose since the best practice for the usage of *Tasks* requires that they are primarily intended for the handling of the message that is sent [Pol14]. However, this cannot be ensured for the elements resulting out of the *Actions* of a Service Blueprint.

Applying this new rule on the example from Figure 5.3, for the BPMN representation this would mean, that after the *Task* {explain procedure} there needs to be a *Throw Event* that communicates the local progress to the remote process. On the other side, the message is received using a *Catch Event* that, when the message arrives, triggers the customer *Task* {sign approval}. Within this regard, the *Catch Event* stops the ongoing *Sequence Flow* until the communication is performed and then initiates the further flow. This way it is possible to synchronize the two processes. As a result, the BPMN segment as shown in Figure 5.4 is created automatically via the transformation process.

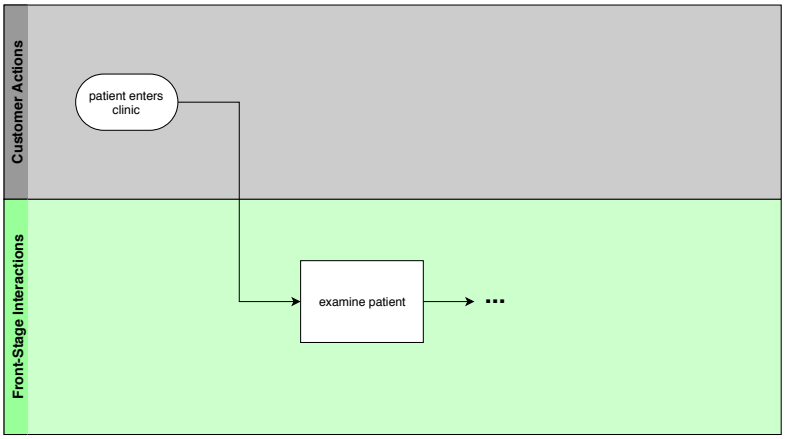


Figure 5.5: Initial service flow across different processes in Service Blueprinting

5.4.2 Implications for Start and End Events

The next topic that is related to the propagation of the unified *Action and Communication Flow* across multiple *Pools*, is the handling of the *Start and End Events* of the process. At this point it should be noted that within the context of this work, a Service Blueprint is expected to have exactly one *Start Point* and at least one *End Point*.

This is sufficient to logically open and close a consistent diagram flow. But, due to the nature of BPMN, within the target of the transformation, now there are multiple distinct flows that are required to have their own *Start and End Events*. However, the direct transformation of the Service Blueprint counterparts is only able to provide the elements for very specific processes. For example, if the Service Blueprint *Start Point* is present within the *Customer Area*, as an output it is possible to transform this point into a corresponding *Start Event* within the customer process. But the one for the organization is still missing. Due to this constellation, the transformation mechanism needs to take care of the consistency of the generated solutions and add *Start and End Events* accordingly.

As a small example, let us assume a situation where a service is initiated via a *Start Point* on the customer's side, as shown in Figure 5.5. Logically, the subsequent *Action and Communication Flow* crosses the borders of the contained area at some point during the execution phase and so the situation as outlined above has to be handled when it comes to the transformation process. As a solution compliant to the general handling of flow steps that have an impact on different processes as discussed above, the very first *Catch Event* located within the remote *Pool* that does not contain the *Start Point* of the Service Blueprint has to be a *Message Receiving Start Event*, as shown in Figure 5.6.

A special case that needs to be considered in this context concerns the treatment of Service Blueprint *Decision Points* and *Parallel Paths*. It is possible to have a flow constellation where the first activity that affects both parties is related to such a splitting

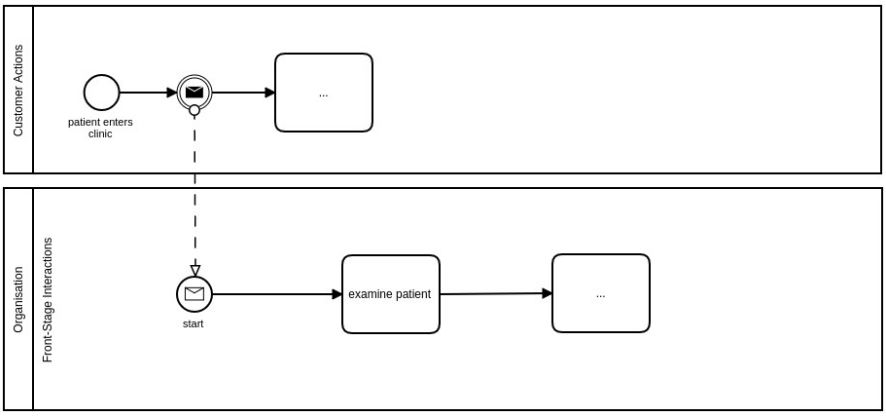


Figure 5.6: Initial service flow across different processes in BPMN

point. In these situations, it may be necessary to mirror the resulting *Gateway* in the foreign process (this aspect will be discussed in more detail in an upcoming section). If this is the case and a corresponding *Start Event* is not already present it has to be created and linked to the foreign splitting point. Now this newly created *Event* is not associated with a specific message, but to a general initiation of the respective process flow.

Looking at the other end of the *Sequence Flow*, as a result of the transformation steps, it is possible that some processes are not concluded via necessary *End Events*. To resolve these situations, the transformation mechanism is simply designed to close loose ends by adding the required *End Events*. The consideration of possible *Message Flows* can be neglected at this point.

5.4.3 Treatment of Decision Points

As already briefly discussed above, another aspect that needs to be taken care of when distributing the unified flow of the Service Blueprint across multiple distinct processes concerns the handling of *Decision Points*.

Basically, a *Decision Point* is logically equivalent to the routing mechanism as imposed by the *Exclusive Gateway* of BPMN, which is, as a consequence, used as a counterpart during the transformation. Furthermore, there are three possibilities to conclude such a splitting point within a modeling language in general. It could be the case that all outgoing, alternative paths meet again while using some kind of merging point (or multiple shifted ones), or each path stops on its own with a corresponding end node (e.g. Service Blueprint *End Point*, BPMN *End Event*). The third possibility is, of course, a mixture.

When it comes to the transformation, this needs to be considered, since it might be necessary to reflect the implications of a *Decision Point* within another process linked via the *Message Flow*. But this only has to be done if the splitting point at hand actually has an impact on the foreign process. During the transformation, this is detected by

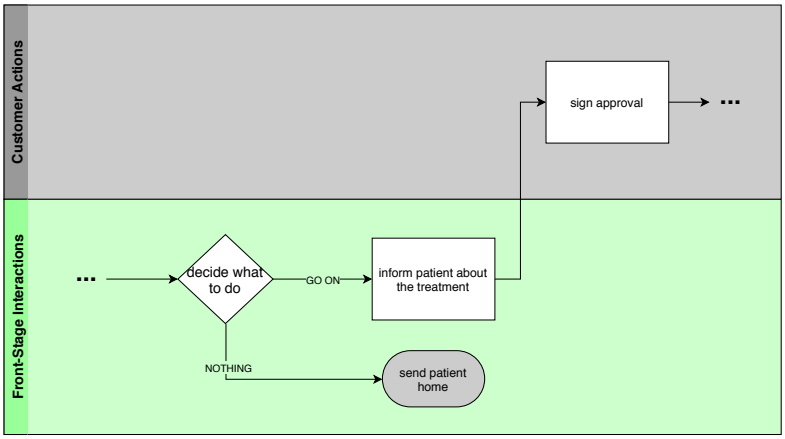


Figure 5.7: Service flow with *Decision Point* in Service Blueprinting

checking for the subsequent steps whether they cross the borders of the current process or not. To be able to correctly assess a specific point, the functionality has to be able to determine if it is concluded via a corresponding merging point or the closing of all outgoing paths.

An example can be seen in Figure 5.7. For this kind of splitting mechanism, it is considered to be sufficient if at least one outgoing path crosses over to another process. This is the case since the other participant needs to be informed about the ongoing execution of the flow. For instance, if the *Decision Point* is not reflected and a local path is chosen, the foreign process could not be completed due to the missing alternative.

Thinking about a proper way how to represent these situations using the notational concepts offered by BPMN, a suitable solution was found in the application of *Event-based Gateways* where each outgoing path is linked to a receiving *Message Event*. Once more, for each subsequent linkage of the local *Exclusive Gateway* it is determined if it crosses over to the foreign process. If this is not the case and the path is only locally relevant, the next logical element is a throwing *Message Event* that is used to propagate the decision that was made. Otherwise, along the path there will be a point where a message is sent to the foreign process which is then caught again by a *Message Event* linked to the *Event-based Gateway*, as shown in Figure 5.8.

5.4.4 Treatment of Parallel Paths

Another issue that needs to be addressed within this context is the handling of *Parallel Paths* that are assessed to have an impact on other processes. Although this point seems similar to the topic of *Decision Points*, which is true to a certain degree, especially when thinking about the concluding characteristics, there are some aspects that require a separate consideration.

The main reason for doing so lies within the very nature of the base mechanisms. Whereas

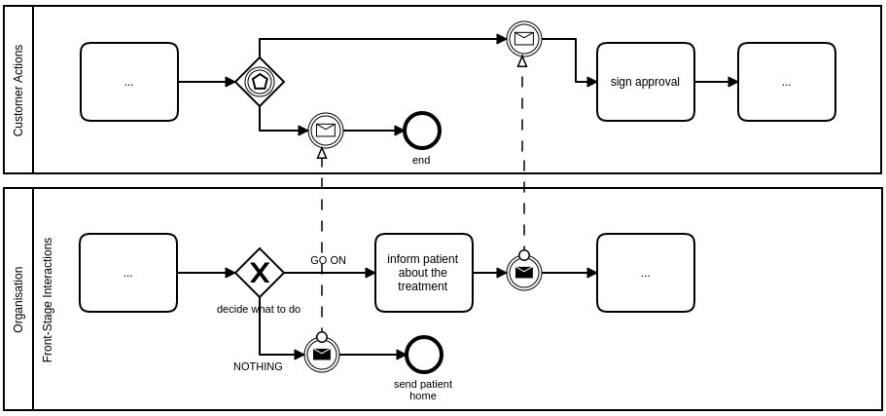


Figure 5.8: Service flow with *Decision Point* in BPMN

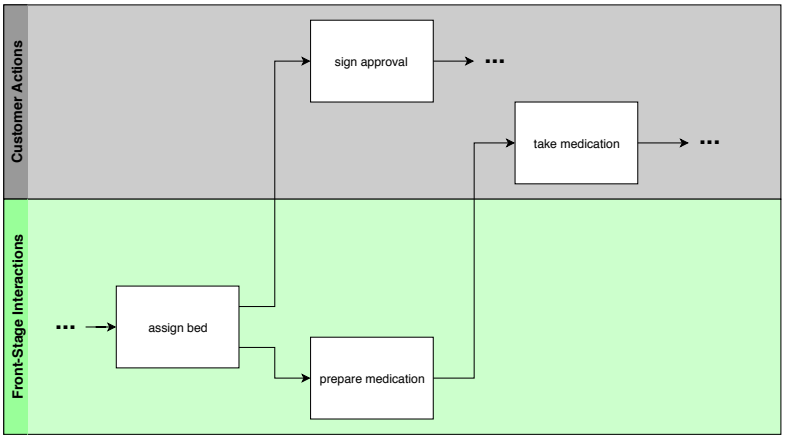


Figure 5.9: Service flow with *Parallel Path* in Service Blueprinting

for the *Decision Point* at least one outgoing path is taken, when it comes to the *Parallel Path* all are processed. At this point, this is relevant since this behavior reflects on the decision whether it is necessary to mirror the splitting functionality inside the foreign process or not. Let us assume, that the *Parallel Path* is concluded without crossing the borders of the current BPMN pool. In this case, as it is for *Decision Points*, a reflection of the local situation is of course not possible. Thinking about a case where one path has been identified to impose a foreign impact, in contrast to the previous topic, due to the situation that all outgoing paths are processed anyway, there is no need to mirror the *Gateway*. This is only required if two or more were identified. In such cases it is necessary, but not to reflect some decision, but to enable the parallel handing that comes with the nature of the underlying functionality, as shown in Figure 5.9. The corresponding BPMN result can be seen in Figure 5.10.

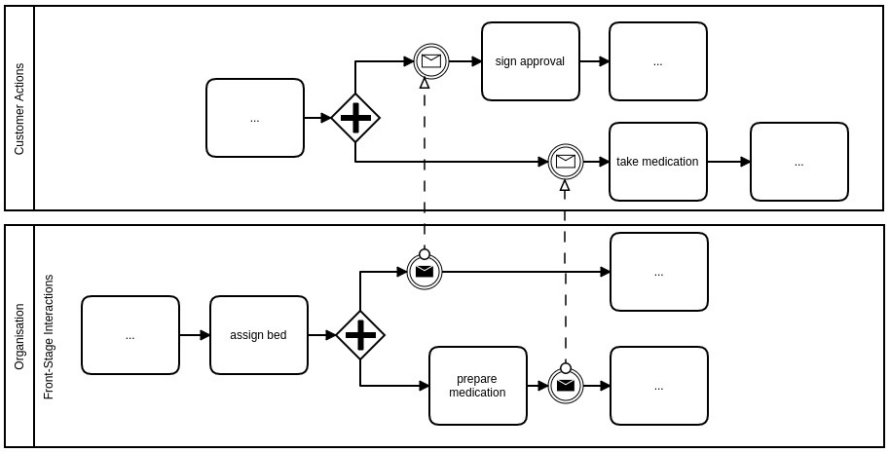


Figure 5.10: Service flow with *Parallel Path* in BPMN

5.4.5 Treatment of Preparative and Managerial Actions

The next topic that needs to be considered when thinking about the separation of the Service Blueprint *Action and Communication Flow* are *Preparative and/or Managerial Actions*. Looking at the service-oriented modeling language, these elements do not directly participate during the execution phase but can be considered as providing the representation of interdependencies with the environment of the organization's offering.

As such they are only illustrated while using *Actions* that are decoupled from the common flow notation and connected only using edges indicating the observed dependencies. Following this description, the transformation towards BPMN can also rely on the same principle as the previous topics, but for a different reason. Whereas conditioned by the separation of concerns, using distinct customer and organizational processes results within the division of the common Service Blueprinting flow, due to the loose coupling, the current issue does not require such a functional transition. Therefore this aspect can be seen as a more logical consequence of the situation already present within the source model.

Nevertheless, if there are in fact *Preparative and Managerial Actions* present within the Service Blueprint, the resulting BPMN model inherits a third process that needs to be created. In contrast to the other two (customer and organization), the complexity of the contained flow notation can be expected to be lesser. The corresponding Blueprint areas are not intended to have *Start and End Points* and no logical relation of its elements among each other. Since, of course, this is not possible within the domain of process modeling, a generic *Start and End Event* has to be created within the respective *Pool* structure. Furthermore, logically, due to possible missing relations between the source elements, it basically cannot be determined which elements should be subsequent. This problem can be dealt with while applying *Parallel Gateways* that initiate a parallel execution of all components. If there is some linkage present within the source model, a

corresponding orchestration is recreated within the context of the target notation.

A corresponding example can be found in Appendix C.

5.5 Creating the graphical Model Representation

This section is dedicated to the discussion targeting the creation of the graphical representation within the target language BPMN. As indicated above, this step is especially important for the usability of the final transformation and ensuring the validity of the resulting models. To do so, the *Java* library *Camunda Model-API* is integrated within the development environment. It offers the possibility to create the graphical counterpart referencing to the logical structure of the target concept while setting the necessary measures and coordinates.

5.5.1 The Representation in General

Due to the varying level of abstraction and detail between Service Blueprinting and BPMN, the latter one requires a greater number of notational elements to capture a specific situation. Furthermore, the fact that the *Action and Communication Flow* needs to be divided and synchronized between two distinct processes, amplifies this situation.

While thinking about a way how the diagram part of the model could be created, one idea was to reuse the graphical details that can be extracted out of the source model. The main target was to generate a representation that is quite similar to the original Service Blueprint and so helps the user while maintaining the different diagrams. The following code snippet shows how the creation can be achieved while applying the functionality provided by the library. One thing that should be noted at this point concerns the different nature of the coordinates stored within the two different diagram representations. Whereas *Draw.io* uses a relative positioning, which means that the coordinates are valid within the context of the surrounding object (i.e. the containing *Area*), the graphical part of the *Camunda Model-API* uses absolute ones (i.e. valid within the context of the top-level modeling canvas). Thinking about the functional conversion, this aspect needs to be considered.

```
//create graphical representation
BpmnShape shape = bpmnModel.newInstance(BpmnShape.class);
shape.setBpmnElement(bpmnNode);
processPlane.getDiagramElements().add(shape);

//create bounds for shape
Bounds bounds = bpmnModel.newInstance(Bounds.class);
bounds.setHeight(height);
bounds.setWidth(width);
bounds.setX(x);
bounds.setY(y);
shape.setBounds(bounds);
```

But of course, at this point, the observations that were stated in the previous sections impose a complexity when applying the gathered graphical data.

First of all, one has to take under consideration, that at least due to the best practice of BPMN, it is very advisable to reuse measurements for the applied elements as done within the context of official documentations and common tool-sets (e.g. the *Camunda Modeler*). Now it is theoretically possible to work with a graphical notation for Service Blueprinting where the corresponding counterparts of the transformation do not inherit similar measurements. If the specific element of the Service Blueprint is larger than the one of BPMN this does not impose an issue. However, if it is the other way around, it is theoretically possible that within the result of the transformation, the newly generated graphical elements stand in conflict towards each other, which cannot be considered as a desirable outcome.

To target this issue, a very straightforward and easy solution is to simply offer a graphical set that is compatible with the BPMN counterpart. Since the Service Blueprint notation and the corresponding *Draw.io* library was created within the context of this work and the measurements as offered by the process-oriented language have proven to be applicable and convenient for the intended modeling purposes, this has been considered as a viable resolution.

5.5.2 Graphical Conflicts and their Resolution

What still remains, is the general observation, that the BPMN representation of a Service Blueprint does, in fact, require more details to capture the situation at hand. However, this does not automatically cause conflicts within the diagram part of the model, since the result is related to the way the Service Blueprint is modeled. To be more precise, it depends on the arrangement and especially on the density of its elements. As shown in Figure 5.11 and Figure 5.12, it is possible that two elements are placed in a way that, when transformed to BPMN, causes conflicts during the automatic positioning of the additional elements. Furthermore, when thinking about the necessary representation of a local *Gateway* within a foreign process (e.g. mirroring the local decision), the complementary elements might stand in conflict towards the ones predefined by the source model, as shown in Figure 5.13 and Figure 5.14.

By instinct, the main approach to resolve such situations would be to perform some kind of graphical conflict management for the resulting model. However, there are some issues that need to be considered in this regard. First of all, if this should be done, it has to be at the same time as the creation of the logical and the diagram part of the model. Basically, it could be a good idea, to first generate the model without any amendments and perform the conflict resolution afterward. However, due to the fact that each element is set via specific coordinates including the way-points of the model edges, such an undertaking would probably require to dismantle and reassemble the whole diagram after the actual transformation process is finished. So the more convenient solution is to already think about the corresponding steps when converting the contained

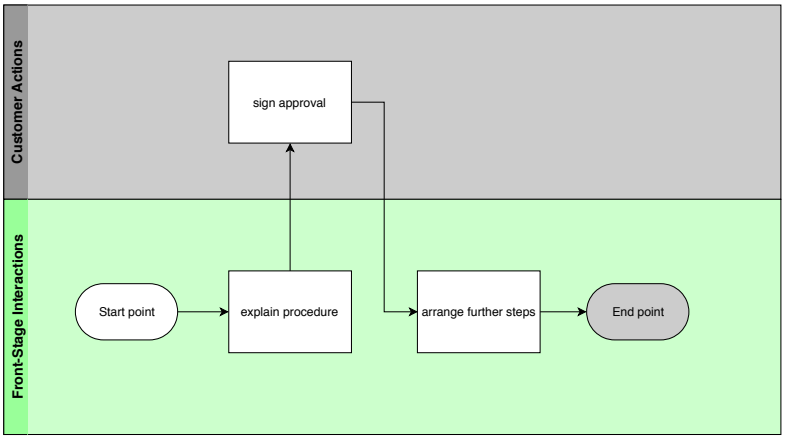


Figure 5.11: Simple service flow in Service Blueprinting that leads to graphical conflicts

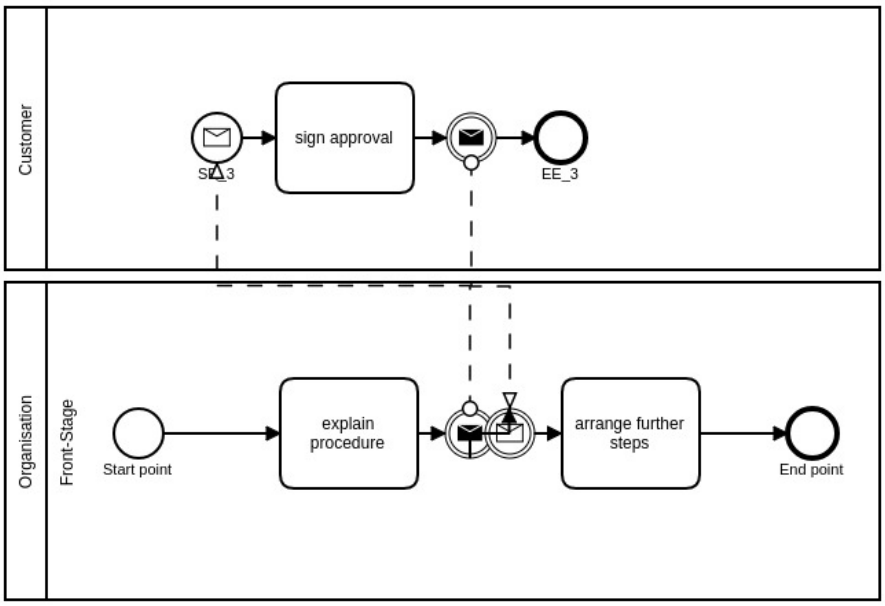


Figure 5.12: Transformation result of Figure 5.11 without graphical processing steps

elements. For the task at hand, this is a possible solution due to the fact that with the Service Blueprint model as a source, one has a template, a layout that can be used to calculate possible conflicting situations. Furthermore, with the already partly existing BPMN structure during transformation time, additional checks could be performed.

Model Density

However, when looking at conflicting model constellations and cases as the ones shown in the illustrations mentioned above, the observation has been made, that such resolving

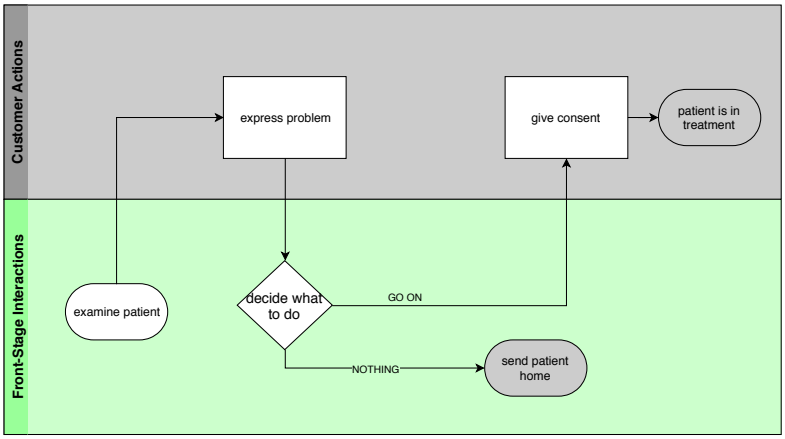


Figure 5.13: *Decision Point* in Service Blueprinting that leads to graphical conflicts

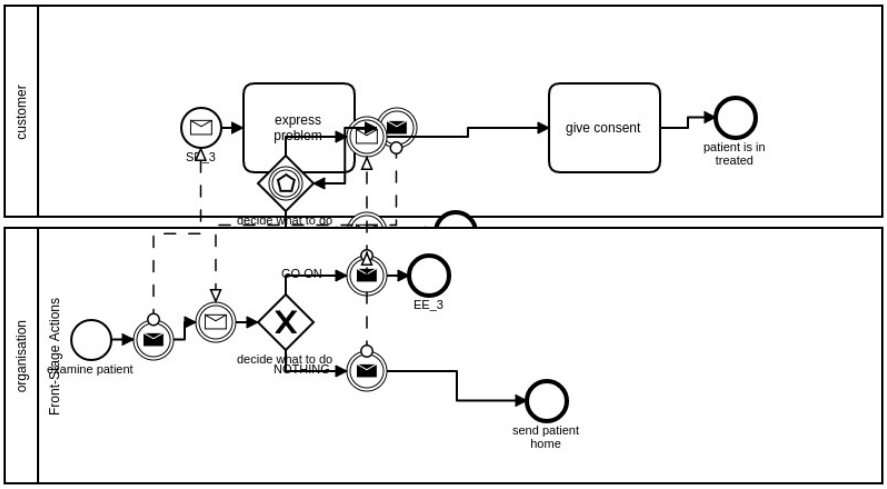


Figure 5.14: Transformation result of Figure 5.13 without graphical processing steps

steps might not only be very complex due to the vast amount of specific arrangement of the corresponding model elements, it would not suffice to generate a final model that does not inherit a conflicting diagram component. The reason for this lies within the initial approach towards the creation of the underlying data structure. As indicated above, the graphical coordinates applied for the BPMN diagram are extracted out of the Service Blueprint, to be able to create a transformation result that does correspond very well to its source structure. This however introduces the problem, that during the creation phase for the Service Blueprint, the designers do not and, of course, should not take care of the model's density (i.e. how much elements are placed on the modeling canvas outlined by the respective areas). This leads to the situation that it is extremely difficult to resolve conflicting elements if the density is too high. However, the main problem is, that this is also true for source models that do not seem very crowded but

may be perceived as quite common within the domain of general flow modeling.

To tackle this topic, the approach is now to think about a way to handle this initial situation, instead of trying to fix conflicts when they occur. As indicated above, during the various test runs it has been observed, that the source model's density is a main cause of graphical conflicts afterward. So the main idea is to come up with a way, that targets mainly this aspect and tries to decrease this indicator.

First of all, when talking about the model's density, within this context it is not sufficient to review this aspect from a subjective point of view. It is necessary to provide a corresponding calculation to be able to use it as an objective basis for the treatment afterward. A straightforward approach would be to use the available model space given by the graphical properties of the containing areas and the number of notational objects. However, one thing that has to be considered in this regard, is the fact that not every Service Blueprint element and area is relevant for the transformation towards the flow notation of BPMN. Neglecting this aspect would result in a skewed calculation and therefore it is necessary to restrict the efforts only to the *Customer, Front-Stage, Back-Stage and Support Area* and the flow elements contained within. The following formula is the result of these considerations.

$$ServiceBlueprintDensity = \frac{NumberOfFlowElements}{Area}$$

However, when it comes to actual models the given method tends to result in very small numbers. Since this indicator is only used in the context of this topic, the decision was made to correct the calculated area by a factor 1000 which makes it easier to work with the output afterward. As a result, in its final form, the formula looks the following way.

$$ServiceBlueprintDensity = \frac{NumberOfFlowElements}{Area/1000}$$

Now based on this calculation, the assessment is made, if the model at hand should be subject to measurements bound to decrease this value.

Graphical Preprocessing

The next issue that has to be targeted is how such a preparative functionality could be put in place. First of all, it is not sufficient to only aim bluntly towards the reduction of the density indicator. The easiest way to do so consist within the enlargement of the available area space. But, obviously, this does not prevent the conflicting situations within the diagram part of the BPMN model. Although this step is basically necessary, along the way each element contained within the corresponding areas has to be vertically and horizontally re-positioned. For this purpose, the final solution implemented with *Java* takes shifting factors for the X and Y coordinates as parameters and subsequently applies the values on the geometric details of the flow nodes. Especially when it comes to the horizontal dimension this requires a recursive algorithm that performs the reorganization

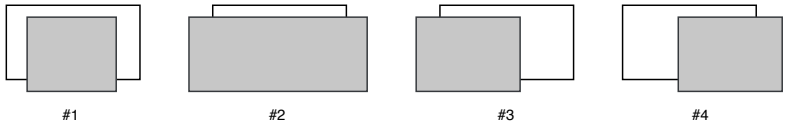


Figure 5.15: Theoretical conflicting cases on the horizontal axis (i.e. the X coordinate)

while considering the previous elements of the sequence flow and takes care of *Parallel Paths* and *Decision Points*.

After applying the overall mechanism, the result is a Service Blueprint model that is resized according to the passed horizontal and vertical factors including a rearrangement of the contained elements. After several test runs, this was found to be sufficient to resolve a large number of graphical conflicts that were a result of the base models layout.

Conflict Management

Although this step does positively contribute to the graphical outcome of the transformation, one point that has to be dealt with separately consists within newly generated elements that are necessary to be able to propagate *Decision Points* and *Parallel Paths* across multiple processes as discussed in the previous section. Since these cases cannot be handled the same way, the logical consequence would be to take care of the conflicting situations when they occur. The corresponding resolution has to be an integral part of the transformation mechanism itself.

To do so, when such an additional *Gateway* is necessary within a foreign process, the first attempt is to place it horizontally aligned with the local one. A control mechanism is used to determine whether a conflict is present or not. This is done while applying the calculated X and Y coordinates and check if there is already an element present at the given position, or that it can be expected that there will be one. For this purpose, two distinct data structures are used. First of all, while creating the graphical representation for new BPMN elements, every shape is kept within a corresponding store. This makes it possible to look for conflicting situations with already positioned elements. The second aspect concerns upcoming new elements that are not yet available when reviewing the previously mentioned reference. The positioning of these objects can be estimated when reviewing the set of flow elements stored in the Service Blueprint model. In detail, the mechanism uses the conflict categories as displayed in Figure 5.15 to assess the situation. The illustration shows the possibilities how two objects, grey and white, can stand in conflict towards each other while considering the horizontal axis (i.e. the X coordinate). The same cases can be applied on the vertical dimension if rotated by 90 degrees.

If a conflict is detected, an additional functionality looks for the next subsequent flow element linked to the current, additional splitting point, that does have the smallest X coordinate and use this new value to rearrange the necessary foreign *Gateway*.

During the ongoing test cycles performed within the context of the development process,

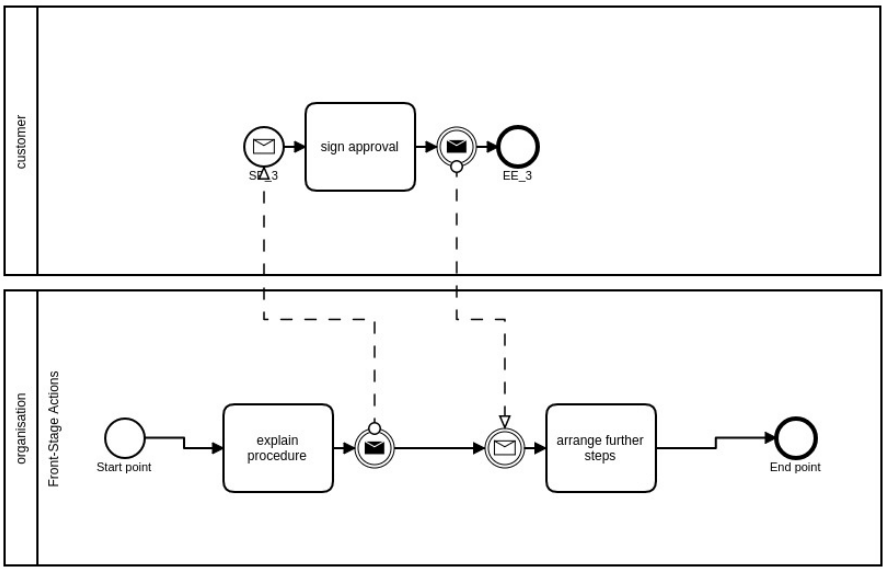


Figure 5.16: Transformation result of Figure 5.11 with graphical processing steps

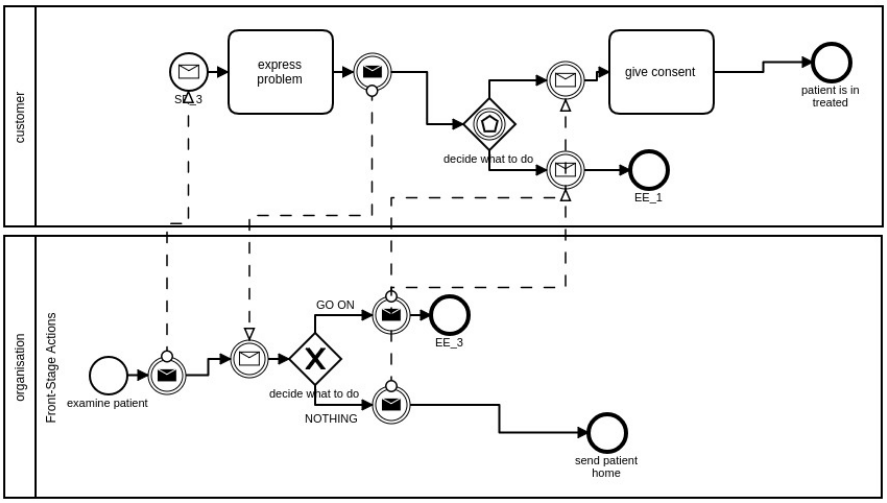


Figure 5.17: Transformation result of Figure 5.13 with graphical processing steps

the resolution steps as discussed in this section have been found to be sufficient to take care of most conflicts. After all the result of the transformation is intended to be a template that can then be applied by the user to finalize the desired BPMN representation of the Service Blueprint model.

Applying the measurements for the graphical processing as discussed within this section, the conflicting situations mentioned above can be corrected as shown within Figure 5.16 and Figure 5.17.

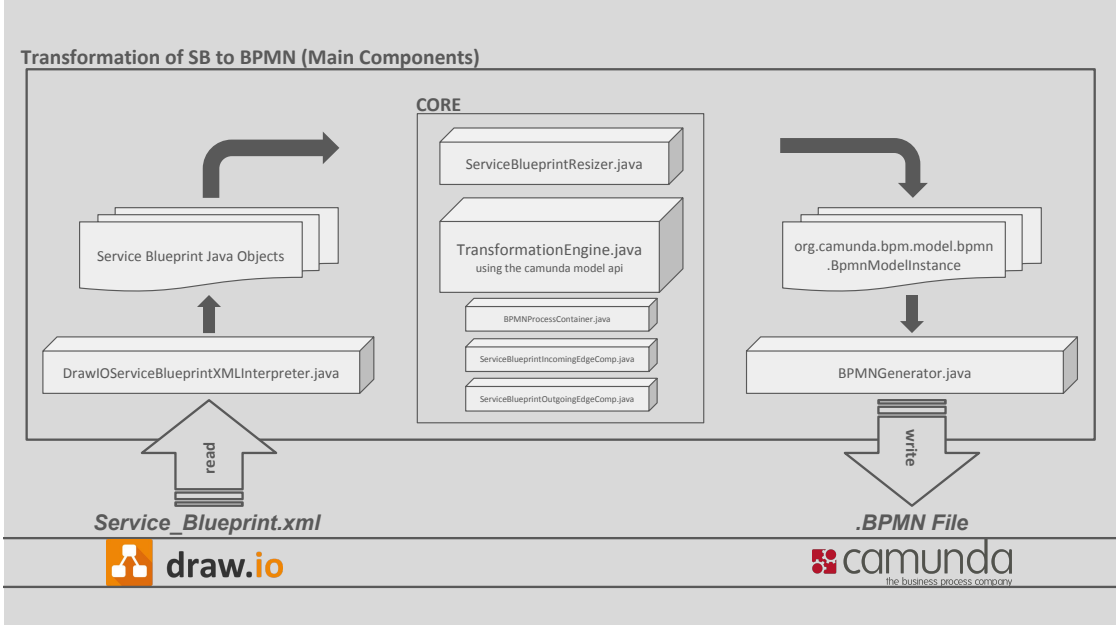


Figure 5.18: Prototype architecture of the transformation from Service Blueprinting towards the BPMN

5.6 The final Architecture and Components of the Prototype

Up until now, the most important aspects and critical issues that occurred during the development phase have been approached. As a result of the realization and testing of the functional topics that were subject to the previous sections, at this point, an overview of the final architecture and the contained components providing the mentioned mechanisms shall be given. After looking at the general outline, the specific elements of the overall layout including their inherent tasks will be discussed.

The architecture of the prototype targeting the transformation of Service Blueprints into BPMN can be seen in Figure 5.18.

Basically, there are three top-level elements. The Service Blueprint modeling environment provided by *Draw.io* in combination with the customized Service Blueprinting library, the graphical modeling environment for BPMN in form of the *Camunda Modeler* and the actual transformation component developed in *Java*. As one can imagine, the general transformation flow starts with the XML export of the model created with *Draw.io*. Afterward, the transformation component takes care of the interpretation and the conversion of the model components. The transformation result is then exported as a *.bpmn* file that can then be interpreted using the *Camunda Modeler*, or another modeling environment that supports the corresponding format (e.g. *Yaoqiang BPMN Editor*).

The transformation component itself can be logically further divided into three parts. The interpretation of the Service Blueprint XML files, the transformation core component and the generation of the *.bpmn* files. The corresponding elements work together while relying on a Service Blueprint *Java* representation that was developed within the context of this work and the BPMN *Java* Object provided by the *Camunda Model-API* (i.e. *org.camunda.bpm.model.bpmn.bpmnModelInstance*). This separation makes it possible to use the core algorithm responsible for the transformation of the source model and the generation of the *.bpmn* file within another context where a corresponding Service Blueprint is created while using a different modeling environment than *Draw.io*. This seemed advisable since the use of the mentioned tool-set was intended as a first step towards the general computerization of the corresponding concepts targeting services. Assuming the transformation environment should be set-up for real-life scenarios, it could be the intention to rethink the provided tool support for the more abstract language. If so, it would be sufficient to only develop a corresponding adapter that reads in the respective files (not necessarily XML) and passes on the generated *Java* object.

The following paragraphs will briefly explain the functionality of the main components as they are illustrated in Figure 5.18. The modeling environments *Draw.io* and the *Camunda Modeler* including the corresponding file structure have already been discussed within a previous section and will be neglected at this point. The more detailed functionality shall be elaborated within a later chapter.

5.6.1 DrawIOServiceBlueprintXMLInterpreter.java

This component contains the algorithm implemented in *Java* that takes care of the interpretation of the specified Service Blueprint XML file according to the specification made above and the generation of the Service Blueprint *Java* object, which will be discussed within the next section. For this purpose the *Java* class expects the path to the corresponding file as an input parameter, which is then imported while using a common *Java* file reader. The general XML processing is carried out via the standard *javax.xml.parser.DocumentBuilder* [Mky08].

As a result, an object of the type *org.w3c.dom.Document* is created that offers a *org.w3c.dom.NodeList* data structure that can be used to recursively iterate through the XML nodes and their children. While doing so, along the way, each node is interpreted according to the Service Blueprint XML specification and a corresponding *Java* entry is created and added to the target data element. When finished, the method used to start-up the core functionality returns a Service Blueprint *Java* Object representing the source model that was drafted while using *Draw.io*. It is compliant with the structure outlined in the following section.

5.6.2 Service Blueprint Java Object

The Service Blueprint *Java* Structure is generated as a result of the interpretation of the corresponding Service Blueprint source file (e.g. created via *Draw.io*). It builds up

the basis for the subsequent transformation steps including the corresponding graphical preprocessing. Since the target of the overall undertaking is to capture as many aspects as possible while using BPMN as the target language, it is necessary that the structure at this point reflects all details present within the given Service Blueprint.

5.6.3 Transformation Core Component

The core component is responsible for all tasks related to the actual transformation process. This also includes preparative steps as discussed targeting the graphical representation of the final BPMN model. For this purpose the core environment expects a Service Blueprint *Java* representation and generates a model object corresponding to the *org.camunda.bpm.model.bpmn.bpmnModelInstance* as a component of the *Camunda Model-API*.

The next paragraphs shall provide an overview of the base functionality of the main parts of this core element.

ServiceBlueprintResizer.java

The first member of the main section of the prototypical transformation environment is the Service Blueprint Resizer. As discussed within the chapter targeting the graphical representation of the final BPMN model, due to the larger number of conceptual elements that are necessary to capture the service within the process-oriented language compared to Service Blueprinting, it is necessary to take preliminary steps regarding the conflict resolution.

Within this context, as input parameters, the *Java* class expects an instance of the Service Blueprint *Java* object, as well as factors for the horizontal and vertical resizing of the given source model. As a result, if the calculated model density exceeds a specific threshold, the functionality returns the same model but graphically augmented by the recursive application of the given values to the model areas and the contained elements. As discussed previously, this step is executed to reduce conflicts when it comes to the positioning of the graphical elements by a significant amount.

TransformationEngine.java

The Transformation Engine is the heart of the transformation towards BPMN. It is responsible for all steps directly related to the conversion between the two modeling languages. Due to the inherent complexity of this task, the corresponding *Java* class is the largest one in the whole environment with about 2800 lines of code in its final form.

Of course, as an input parameter, the functionality expects the preprocessed Service Blueprint stored in the mentioned *Java* object structure. After the initiation of the process, at first the main components like the BPMN model elements itself, the processes, the participants and the lanes of the targeted model are created. Afterward, a recursive mechanism goes through the *Action and Communication Flow* of the Service

Blueprint source and subsequently creates the required BPMN counterparts. To complete the creation of the model content, the *IT Elements* and *Preparative and Managerial Actions* contained inside the Service Blueprint are generated. The last step during the transformation consists within the closure of loose ends that were created along the way.

As a result of the overall undertaking, the top level function of the *Java* class returns the final BPMN model in form of the *org.camunda.bpm.model.bpmn.bpmnModelInstance*.

BPMNProcessContainer.java

The BPMN Process Container is a helper class used within the context of the Transformation Engine. The main purpose is to enable a convenient management of the necessary BPMN processes with respect to their *Start and End Events* and a special focus on the unfinished business (i.e. loose ends). Especially when it comes to the last step of the overall transformation, each time a recursion comes to an end or in general when it seems necessary (e.g. flow crosses over to another process), this data structure makes it possible to correctly handle process elements that are not directly or indirectly linked to an *End Event* via a corresponding continuous sequence flow.

ServiceBlueprintIncomingEdgeComp.java & ServiceBlueprintOutgoingEdgeComp.java

While creating the graphical representation for the model (i.e. the BPMN diagram part) during the transformation, thinking about an element with multiple incoming respectively outgoing flow edges, the vertical order of the multiple predecessors or successors is not naturally given. The logical order of the elements heavily depends on the sequence of the specific tags within the source XML file. However, since it was necessary to determine this aspect according to the vertical positioning, a Comparator (*java.util.Comparator*) implementation was created that enables the sorting of the respective listings.

5.6.4 BPMNModelInstance.java

As mentioned on several occasions, the BPMNModelInstance is a *Java* element provided by the *Camunda Model-API* and used as a target for the transformation process. For further details concerning the complete structure of the library please have a look at the official documentation [Ser18c].

5.6.5 BPMNGenerator.java

The last member of the actual transformation component implemented in *Java* is the BPMN generator. It encapsulates the functionalities for the validation and storage of the final BPMN file which are provided via the *Camunda Model-API*. For this purpose, as an input parameter, the *Java* class expects the BPMNModelInstance that represents the result of the transformation mechanisms processed within the context of the Trans-

formation Engine. After the execution of the validation step, the final model is exported into a *.bpmn* file using a given target directory [Ser15b].

After the current class completes its work, the result can be further processed using a compatible functionality (i.e. automatic processing) or graphical editor (e.g. Camunda Modeler, Yaoqiang BPMN Editor).

5.7 The Transformation Environment and Process in Detail

The next sections shall be used to describe the prototype and the corresponding actual transformation process in more detail. Up until now, a more abstract view was given which will serve as a basis for the upcoming elaborations. For this purpose, after a small remark targeting the complete structure of the implementation and its environment, a step by step description of the conversion mechanism shall be provided.

5.7.1 The Environment

As mentioned in a previous chapter, the main implementation was done using the programming language *Java* and *Eclipse Oxygen* as a developing environment. For the management of the external dependencies *Maven* was applied and *Git* via *Bitbucket* as a versioning mechanism. After the completion of all necessary steps, the project's structure consists of the components and packages as shown in Figure 5.19. Besides the main elements that were already subject to corresponding discussions, also customized exception classes have been added to be able to propagate failures during the run-time of the conversion accordingly.

5.7.2 Transforming a Service Blueprint into BPMN

The purpose of this section is to provide a detailed illustration of the core transformation process. To be able to give a more clear understanding of the corresponding program parts, the model that can be found within the Appendix C will serve as a practical example that guides through the mechanisms. It consists of the abstract representation of the admission of a patient in a medical facility. Please note that this example is only intended to support the purpose of this chapter and does not represent actual workflows. As a starting point, it shall be assumed that the exported Service Blueprint was already successfully interpreted and stored within the Service Blueprint *Java* structure, which makes the graphical preprocessing the first step that will be discussed within the context of this section.

Graphical Preprocessing

The logical component responsible for the graphical preprocessing is situated in the *Java* class *ServiceBlueprintResizer.java*. Due to reasons that were already elaborated within a

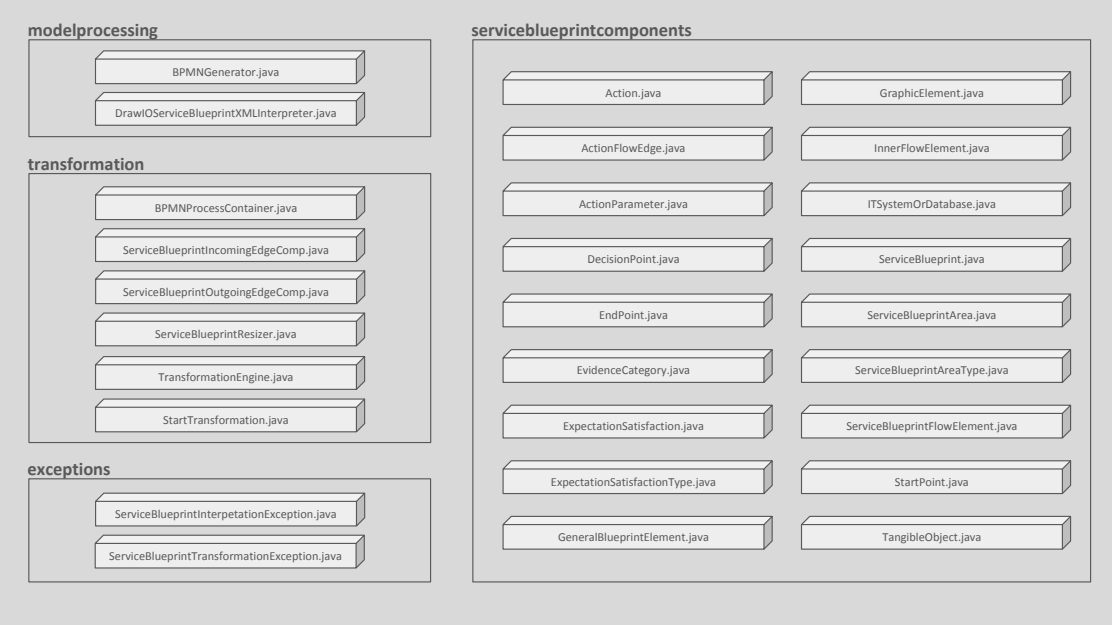


Figure 5.19: Package structure of the prototype developed in *Java*

previous section, this mechanism is applied to increase the size of the overall modeling canvas and rearrange all contained flow elements accordingly. This way the discussed density of the Service Blueprint can be decreased.

For this purpose, the corresponding start-up functionality is triggered while passing on the imported Service Blueprint (in form of the Service Blueprint *Java* object), as well as the factors for the horizontal and vertical resizing (in pixel). The first step inside the current class consists within the calculation of the density of the current model while applying the formula discussed in a previous section. Looking at the exemplary model, the indicator is set to 0.0127. This result is compared to the static value of 0.01, which was set after running several test model and perceived as a good threshold for determining whether a resizing is necessary or not. Of course, this value can be changed if deemed necessary. As one can see, for the given model, the logical comparison returns *true* and so the processing should continue.

The next step consists of the treatment of the *Start Point* of the Service Blueprint. After the horizontal and vertical repositioning is done, the recursive processing of the target nodes of all outgoing edges is triggered. As an additional input parameter, besides the factors applied for the shifting, the sub-method takes a counter which is used to determine for a specific node how far it is necessary to move the element on the horizontal axis. This makes it possible to create a wider spacing between the different shapes which is required to fit in helping notations like the *Catching and Throwing Events* of BPMN later on. Instead of maintaining a global counter, the parameter is handed down along the recursion to be able to keep the relative positioning of elements on different paths originating from

the same splitting point. This way, elements that are positioned vertically aligned within the Service Blueprint, will generally stay so even after the repositioning is completed.

Coming to the actual logical functionality of the current method, the first task is to check for the node if it was already handled within this context, which is possible to the nature of the mechanism. So for instance, looking at the model given within Appendix C, it will be the case that the *Action* "administer medication" is visited twice when the resizing is applied (one time for each outgoing path of the previous splitting point). The overall target should be to apply the shift that leads to the maximum value of the X coordinate. However, if the current step is the first visit of the node at hand, the next thing that has to be determined is whether the current situation is a case where a merging gateway is the direct successor of a splitting point or not. This is relevant since it has been observed that if so, even more additional space between the elements is required. To do so, if the current node has multiple incoming paths, for each corresponding source element it is assessed if it is a splitting point. To be able to use this information later on, a Boolean variable is set accordingly. When looking at the mentioned example, such a situation cannot be found.

Now, before the actual shift can be executed, the original value of the X coordinate is stored in a map using the element's id as a key, which is done or the usage later on. Afterward, the coordinates (i.e. X and Y) are amended while applying the factors and the shift counter for the horizontal dimension. If the Boolean value that was set in the last paragraph is true, the X coordinate is increased to a higher degree. The counter is also modified accordingly.

Since the current node was now subject to the resizing, it is added to the set of elements that are completed.

If the element was already treated within the context of another path through the model, the functionality of the recursive sub-method as discussed so far is not applicable. At this point, the shifting was concluded and the X and Y coordinates were altered. However, for the first one, it is possible that the current path is logically longer than the one that leads to the treatment before. In that case, another calculation would make it necessary to shift the node even further along the axis. At this point, the original value of the X coordinate that was stored previously is used to determine this new position. If it is found to be larger than the current value of the element, it is changed accordingly.

In any case, the last step executed within the context of the sub-method consists of the recursive method call for all targets of the outgoing paths of the current node. Of course, the general recursion stops when the end of the *Action and Communication Flow* is reached (i.e. the *End Points*).

What still remains is the amendment of the modeling canvas itself, which means the resizing of the Service Blueprint *Areas*. Up until now, without the corresponding step, the contained flow notation would exceed the borders of the surrounding modeling space. To target this issue, a sorted list of all the contained *Areas* is iterated and for each one, the

height, the width, and the Y coordinate are adapted. For the latter one, the horizontal positioning should be taken into account to avoid overlapping.

Once this final step is completed, the actual transformation can be initialized.

Transforming the Model

To be able to perform the main conversion, the previously resized Service Blueprint represented via the discussed *Java* object structure is passed on to the class *TransformationEngine.java* which contains the logical core of the transformation process. In contrast to the previous program part, the initialization of the corresponding functionality does not require any additional parameters. As it is common for such components, after the instantiation of the containing class, the procedure is initiated via a top-level method call.

Targeting this mechanism, the following paragraphs represent a step-by-step walkthrough of the respective *Java* program. For a better understanding, the detailed code is reviewed while maintaining a higher level of abstraction which offers a clearer focus on the actual functionality of the specific parts. Accordingly, not every command and sub-method call is explicitly mentioned, but in case of the latter one, if such a segment shall be discussed, a corresponding excursus is created to keep the logical sequence of the specific components intact. In general, this concerns the creation of the logical as well as the graphical elements of the model.

Creating the Top-Level Model Elements

The first step towards the desired output is the creation of the necessary elements representing a blank BPMN model within the *Java* structure provided by the *Camunda Model-API* [Ser18c]. Since, as discussed previously, it is not sufficient to only deal with the logical framework of the target environment, within this step also an empty diagram representation has to be put in place. Luckily, for this aspect, the mentioned library offers the corresponding counterparts as well. As a final generic step, the process plane is created (*org.camunda.bpm.model.bpmn.instance.bpmndi.bpmnPlane*) which will then be used to add the generated graphical shapes to the diagram. At this point, the general preparatory steps are concluded and all subsequent tasks are dedicated to the actual representation of the source model.

Creating the main BPMN Structure

Next up, a sub-method is called that is responsible for the creation of the logical structure as a result of the application of the transformation rules on the *Areas* contained within the Service Blueprint. This includes the required, for now empty, processes as well as the *Pools and Lanes* of the BPMN model. In detail, the provided functionality consists within a top-level loop that iterates over the given Blueprint *Areas* and determines which counterparts need to be generated and added to the BPMN output. Of course, this concerns the logical as well as the graphical component of the diagram. To do so, three major cases are distinguished in accordance with the necessary BPMN processes as a result of the transformation concept. The code segment differentiates if the current *Area* belongs

to the customer (i.e. *Customer Actions*), the direct actions of the organization (i.e. *Front-Stage Interactions*, *Back-Stage Interactions*, or *Support Processes*) or the organization's preoperative operations (i.e. *Preparative Processes*, or *Managerial Processes*).

In each case, if not already present, the corresponding process representation is created using the *Java* structure provided by the *Camunda Model-API*. For the customer's *Area*, nothing has to be added, since the corresponding process does only have one section which is considered as default. For the other two participants, at this point, it is necessary to create the *Java* objects for the logical and graphical part of the BPMN *Lanes*. Considering the diagram component of the processes, for the one of the customer, it is rather straightforward, since it is possible to directly apply the coordinates (X and Y), as well as the width and the height extracted of the source element contained within the Service Blueprint. However, when it comes to processes that have to incorporate different *Lanes*, small code segments are necessary that calculate the correct width and height. The first value is set to the maximum of all Service Blueprint *Areas* since within the context of the more abstract modeling language there is no restriction that dictates a unified length of the segments. The second one is of course calculated while summing up the heights of the partitions. One additional aspect that needs to be taken care of, is the incorporation of the label area of the BPMN structure elements positioned on the left side. As a last step, looking at the coordinates of the process representation, once more, due to the lack of a corresponding restriction, both values are set to the minimum of the contained *Areas*.

Looking at the example model within Appendix C, at this point the processes representing the customer, the organization as well as the *Preparative and Managerial Actions* were created. Furthermore, the necessary BPMN *Lanes* for the *Front-Stage Actions*, *Back-Stage Actions*, *Support Processes* and *Preparative Processes* were added.

Initializing the Transformation of the Action and Communication Flow

After the general layout of the BPMN model is created, the transformation rules can be applied on the *Action and Communication Flow*. Similar to the execution of the graphical preprocessing, the basic approach consists within a recursive method call that is used to walk through each branch of the Service Blueprint and generate an appropriate BPMN counterpart. The first step towards this goal is to deal with the *Start Point* of the source model. Since for this work, it is expected that the starting element of a service offering can be uniquely identified, the initiation of the ongoing functionality is done via a simple call of the recursive main method while passing on the *Java* object of the Service Blueprint *Start Point* as a parameter.

In general, the method takes two variables. The first one is the abstract *Camunda Model-API Java* object for a BPMN *Gateway*, which is intended as a possibility to pass on foreign splitting points that are additionally generated to propagate *Decision Points* over multiple processes (for more information please have a look at the corresponding discussion within Section 5.4.3). Since this is certainly not the case for every method call and especially not for the very first, the value can be set to *null*. The second parameter,

on the other hand, references the current node that shall be subject to the transformation and therefore needs to be present each time.

Looking at the contained program code, the first logical operation is used to determine the specific, category of the current Service Blueprint flow element. On its top level, the mechanism differentiates between the treatment of a *Start Point* and inner flow elements. In a later step the second category is spitted further into *Actions* and *Decision Points*. *End Points* are not a part of this distinction since they represent the end of a recursion and do not require additional handling. When the method is called the first time and a *Start Point* is detected, the logical next step is to create the actual BPMN counterpart.

Excursus: Centralized BPMN Flow-Node Creation
To handle each element's creation properly, another sub-method is used which works according to a *get or create* paradigm. This means that for a passed Service Blueprint element, it is checked if the corresponding BPMN representation was already created previously. If so, the method returns the result of the search. Otherwise, a new logical and graphical element is generated, stored and then passed on. This way the management of all actual flow objects has been centralized which makes maintaining and extending the contained functionality more convenient. However, along the way of the direct transformation of the flow elements, it is also necessary to take care of information and details tightly linked to the source object. This includes the indication of *Failure, Waiting and Specification Points* for *Actions*, as well as the handling of *Expectations and Satisfactions* for *Start* respectively *End Points*. Furthermore, this helper method calls another sub-method that takes care of the linked tangible objects. This code section is outsourced since it was necessary to reuse the same mechanism at another point within the overall program. Accordingly, it is possible, that a call of the *get or create* method results within the creation of several BPMN concepts as a result of the transformation of the given Service Blueprint element. Last but not least, the functionality also takes care of the correct assignment to the specific process and *Lane* of the target methodology.

Thinking about the example model provided in Appendix C, the very first element that is created in this regard is a new BPMN *Start Event* with the description {patient enters clinic}. Since the source model contains a set of imposed *Expectations*, for this counterpart a corresponding *Annotation* is created as well. Furthermore, looking at the source model, the current flow node is linked to a tangible object. Accordingly, an additional *Annotation* is added to represent this information.

Continuing the Transformation of the Action and Communication Flow
Coming back to the initial call of the recursive method used to transform the *Action and Communication Flow*, at this point, the BPMN *Start Event* mirroring the current Service Blueprinting node is created along with a BPMN *Annotation* illustrating potential *Expectations* and linked tangible objects. But this is only the first part of the current iteration of the functionality. As a next step, it has to be determined if the *Start Point* serves as a point of origin for the initiation of parallel paths.

Excursus: Handling Parallel Paths

Looking at the chosen notational concepts for the service-oriented methodology, such a splitting point is present if the flow node has more than one outgoing edge. Within the example provided in Appendix C the *Action* {arrange necessary steps} represents such a case. To deal with these situations, the class *TransformationEngine.java* uses a sub-method since this kind of functionality has to be applied for *Actions* as well as the *Start Point*. As parameters this new method expects the foreign *Gateway* that was passed to the recursive top-level method (but of course is *null* for *Start Points*), the current Service Blueprint flow node (e.g. the *Start Point*) and the corresponding BPMN representation (e.g. the *Start Event*).

When it comes to the contained program logic, the first step consists of the determination whether or not the treatment of parallel paths is even necessary at this point. Looking at the exemplary model, the mentioned *Action* is the only one which causes the method to execute further operations. In this case, a new local *Parallel Gateway* is created (i.e. positioned within the current process). Within this context, the propagation of the flow notation across multiple processes comes into play. For this purpose, in accordance with the considerations elaborated within Section 5.4.4, for the current splitting point, it is determined how many outgoing paths cross the borders of the local process. This is done by applying another recursive sub-method call that moves along the upcoming branches and returns a set of edges that are identified to be relevant at this point. If the set is empty or only contains one element, no foreign *Gateway* is necessary. So the graphical representation of the *Parallel Gateway* is created and linked to the *Start Event*. In that case, the method comes to an end.

However, if the set contains multiple edges, it is necessary to mirror the situation within the foreign process. Note, that due to the fact that only the customer's and the organization's process are participants relevant for the *Action and Communication Flow*, the target at this point can be easily identified. Now, the next step within this sub-section is the creation of the new foreign *Parallel Gateway* and the generation of the graphical representation. Due to the conflict potential of this element, this is not an easy task. The corresponding discussion can be found in Section 5.5.

Once the previous step is completed, another aspect that needs to be taken care of concerns the first parameter that is passed on to the handling of the parallel paths. If a foreign gateway is present in the remote process and is waiting to be connected to the further sequence flow, it needs to be dealt with. If this BPMN node is the representation of another local *Parallel Gateway*, it is simply connected to the new one, and the further processing contained within the context of handling the parallel paths continues. If not, it represents a local *Exclusive Gateway* (i.e. the logical counterpart of the *Decision Point*), which makes it necessary to introduce an additional message flow between the processes at this point. This is the case since the specific counterpart at this point is an *Event-based Gateway* which requires being followed by an element of the type *Event* (or *Message Task*). So this new message flow (i.e. a combination of local *Throw Event*, message flow, and foreign *Catch Event*) indicates the propagation of the local decision

that was made in the past [OMG11].

In any case, at this point, the necessary foreign *Gateway* is created. The next step consists of the treatment of the so-called unfinished business within the remote process. This term describes flow elements that are not directly or indirectly connected to a corresponding *End Event* and therefore are incomplete. As mentioned before, such elements are stored within a specific list within the respective process container at the time of their creation. If multiple elements have been identified to be relevant at this point, another *Gateway* has to be created and linked, since in BPMN a *Gateway* should not have multiple incoming and outgoing paths (i.e. it should not be used as a merging and a splitting point at the same time).

As the last section of this program part, another special case has to be considered. It is possible that the created foreign *Gateway* is the first element of the foreign process flow. However, every BPMN *Pool* needs to start with a corresponding *Start Event*. If such an element is missing, a generic default *Event* has to be created and linked to the brand new *Gateway*.

With the treatment of these last issues linked to the handling of the parallel paths, this step is completed and if a *Parallel Gateway* was necessary, it is used as new current BPMN element for the remaining part of the iteration of the recursive method dedicated to the transformation of the *Action and Communication Flow*.

Continuing the Transformation of the Action and Communication Flow

Within a valid Service Blueprint, for a *Start Point* there has to be at least one outgoing edge leading to the subsequent element of the service flow. What has to be done next is the treatment of these linkages and the corresponding target elements. Due to the possibility that there are multiple edges present, this has to be done via a looping functionality. Due to its relevance concerning the creation of the graphical representation, while doing so, the default sorting of the edge lists, which is a result of the XML interpretation of the source model, is not sufficient. Instead, implementations of the *Java* Comparator (i.e. *java.util.Comparator*) are used to enable the reorganization of lists containing incoming respective outgoing edges of a specific node according to the vertical positioning of the source respective target element (i.e. Y coordinate). This way, for the current case, it can be ensured that the first iteration of the loop takes care of the outgoing linkage to the element that is graphically positioned above the other ones. Doing so makes it possible to correctly position elements that are additionally necessary to create a valid BPMN file as shown within the result of the transformation in Appendix C.

As a next step, to be able to take care of the linkage between the *Start Event*, respectively the newly generated *Parallel Gateway* and the upcoming flow elements, the targets of the edges have to be actually created within the context of the target model. Once more the *get or create* methodology, as discussed above, is applied for the creation of the logical and graphical component of the necessary BPMN elements.

Furthermore, at this point, the fact that each edge could be marked with additional information needs to be considered. If such an annotation can be identified as being part

of the Service Blueprint *Data Flow* as specified within the context of this work (i.e. *[data element]*), a sub-method is called that resolves these situations.

Excursus: Handling the Data Flow
This functionality expects as parameters the textual content of the annotation, as well as the BPMN elements describing the source and the target of the corresponding edge. In general, it is a rather straightforward task, which creates the actual BPMN *Data Object* including the corresponding associations linking it to the passed flow nodes. This concerns the logical as well as the diagram components of the data flow. Due to the conflict potential at this point, the generation of the graphical elements is the most complicated part of this sub-method. At the end of the function call, the current annotation that was identified as being a *Data Flow* is excluded for the further processing (i.e. is set to *null*).

Continuing the Transformation of the Action and Communication Flow
After the potential data flow was handled, for the current iteration through the list of outgoing edges of the current *Start Point*, it is necessary to check whether the target object serves as a merging point for multiple incoming edges or not. If this is the case, the target of the BPMN sequence flow has to be a merging *Gateway*. For this purpose, a global storage is used to look if for the current target element, a corresponding *Gateway* was already created and can be reused. If such an object cannot be found, another encapsulated sub-functionality is used that takes care of this issue. This is necessary since it is important that not only a generic instance is created, but the right one.

Excursus: The Creation of merging Gateways
Within the context of the prototypical transformation, two categories are used to merge multiple paths together. Depending on the corresponding splitting point, the logical counterpart can be of type *Parallel Gateway* or *Exclusive Gateway*. The decision is made by a recursive backtracking of all edges that need to be joined together.

Once the common source element of all incoming paths was found, its type is determined and reused for the required decision at this point. However, if, due to some model constellation it was not possible to retrieve the desired splitting point, a *Complex Gateway* is created as a merging point, which has to be changed by the user when it comes to the manual revision of the transformation output.

Continuing the Transformation of the Action and Communication Flow
To proceed with the linkage of the *Start Event* to the target of the current outgoing edge if it is necessary to reuse the generated merging *Gateway* at a later point of the program, it is added to the central storage that was mentioned above. After the corresponding graphical representation is created, the actual sequence flow between the *Gateway* and the BPMN element mirroring the target of the edge from the Service Blueprint is put in place.

What is still missing to be able to conclude the current call of the recursive method is an end-to-end BPMN sequence flow between the two sides of the *Action and Communication Flow*. At this point, it is possible that the source element is either the *Start Event* or a *Parallel Gateway* due to the fact that the Service Blueprint *Start Point* has multiple

outgoing edges. In the latter case, the linkage between the initial BPMN flow element and the new splitting point was already created above. Furthermore, it is possible that the target is either the direct result of the transformation of the Service Blueprint element that is the target of the corresponding edge (e.g. an *Exclusive Gateway* representing a *Decision Point* or a *Task* representing an *Action*), or it is a merging *Gateway* that was necessary to join multiple incoming paths and was already connected to the upcoming element.

Independent of the current constellation, the program code needs to take care of this connecting step. Since this task is similar for many sections throughout the whole *Java* class, it is advisable to do so contained within a reusable sub-method. Depending on the situation at hand, the input parameters may vary from case to case. Accordingly, the layout that was detected above while transforming the *Start Point* is reflected in the method call.

Excursus: Creating the Action Flow Linkage

In general, the function *createActionFlowLinkage* takes the potential foreign *Gateway*, the content of the edge's annotation, the source, and target elements as well as an ordering number which is used to calculate the graphical ordering as parameters. Depending on the situation the first two may be set to *null* which prevents the corresponding code segments from being executed.

The initial step towards the creation of this part of the BPMN sequence flow consists within the determination whether the source resides within a different process than the target. If this is not the case, the required linkage is created via a BPMN flow step using a simple edge. However, if the two ends are not located within the same *Pool*, the current task requires a resolution using the sequence as well as the message flow of the target methodology. To begin with, a new *Message Throw Event* needs to be added to the local process which is used to propagate the domestic progress to the other participant (i.e. organization or customer). After the necessary notational element is created logically as well as graphically, some special cases need to be considered. First of all, it has to be checked if the ongoing path that spans over the whole BPMN diagram (i.e. including sequence and message flow) comes back to the local process at some point. If this is not the case, the newly generated *Message Throw Event* is the last element contained within the *Pool* and therefore has to be linked to a corresponding *End Event*. Since there is none present at this point, a new one has to be created. However, if the path does come back to the local process later on, the next issue that is targeted within the current sub-method is the question if it is necessary to create a new local merging point. This might be the case to reflect an upcoming one that is present in the foreign process and so needs to be put in place due to the alignment of both sequence flows. To determine if such a resolution is required, another recursive helper method is used that moves along the subsequent process elements and checks if such a foreign merging *Gateway* does exist. If this is not the case, the current *Message Throw Event* can be added to the unfinished business of the process which means that it is stored within a specific set to be resolved at a later time during the overall transformation. Otherwise, it has to be checked if a

corresponding local *Gateway* that is used to mirror the foreign one is already present. If not, the logical and graphical details have to be generated and added to the output model of the program. The positioning of the diagram part has to be aligned with the one of the foreign counterpart. Therefore the ordering of the corresponding incoming edges that are present in the other *Pool* needs to be taken into account. Afterward, the merging point is linked to the *Message Throw Event* and, if not already present, added to the unfinished business of the current process.

As a next step, the receiving constellation on the other side of the message flow has to be created. Within the target process, the logical counterpart consists within a *Message Catch Event* that is embedded inside the corresponding foreign sequence flow. Once again, there might be the situation, that this new element represents the very first object within its *Pool* and therefore cannot be linked to a required *Start Event*. If so, the newly generated *Event* is declared as such and put in place without any incoming edges. The calculation of the diagram information that will be used to generate the graphical element heavily depends on the situation that is present at the targeted position (e.g. *Gateway* present).

Afterward, the linkage of all the elements that are necessary to illustrate the current layout of the diagram flow can be set up. First of all, the BPMN source element is linked to the *Message Throw Event*. If an *Annotation* is present at this point, it is assumed that it is a simple text content that is used to mark the outgoing edges within the context of a *Decision Point* (respectively *Exclusive Gateway*). Note that a potential *Data Object* would have been identified in an earlier stage of the transformation and handled accordingly. Now the message flow between the *Throw Event* and the *Catch Event* can be created, which results within the bridge between the two distinct processes. As a last step directly related to the current exchange of information, the foreign *Catch Event* is linked to the target of the overall flow.

However, the transformation is not yet complete since, due to occurring special cases and loose ends as a result of prior process paths, there might be still some situations that need to be resolved to be able to ensure a valid transition between the specific objects. So, as a first step, the splitting *Gateway* that was potentially passed on to this sub-method as a parameter needs to be integrated into the current flow. The correct placement for the corresponding edge is between this element and the *Catch Event* within the remote process. Finally, to conclude the transformation of the actual *Action and Communication Flow* between the Service Blueprint source and target, the unfinished business present within the participant containing the target object needs to be handled. Again there might be a situation where a merging *Gateway* is necessary to be able to conclude the transition accordingly.

Finalizing the first iteration of the Transformation

At this point, the BPMN concepts required to correctly represent the Service Blueprint *Start Point*, the subsequent elements and the linkage between them were created. What now remains to finish the current procedure are the recursive calls of this core method using the target elements of the outgoing edges coming from the *Start Point*. Looking at

the example provided in Appendix C, the next element that has to be dealt with is the *Action* {express problem}.

Next iteration of the Transformation with an Action as source Element
To begin with, once more, the recursive method has to determine whether the current node is a *Start Point* or an inner flow element (i.e. *Action* or *Decision Point*). Now, the current element belongs to the second category. After the *get or create* functionality is used to retrieve the previously created element, it is assessed if the current BPMN node and its subsequent paths were already subject to the transformation. This is done via looking at the outgoing edges of the BPMN element. If there are none, this is not the case and the method should proceed. Due to their different nature, *Actions* and *Decision Points* should be treated separately and the functionality splits the contained code segments even further.

As we are currently dealing with the first one, if multiple outgoing paths are detected when looking at the corresponding Service Blueprint node, a parallel sequence flow should be set up which requires the usage of a *Parallel Gateway*. To resolve this situation, the same sub-functionality that was already used within the context of the *Start Point* is applied. If such a splitting point was necessary it will be used as the current BPMN node for the further transformation steps.

The treatment of the additional information related to Service Blueprint *Actions* was already conducted during the item's creation using the method as described above. So what remains at this point is the handling of the outgoing edges of the current node. This includes the creation of the logical and graphical component of the necessary BPMN target-counterpart as well as the necessary flow linkage (i.e. sequence and message flow). As it was the case for the *Start Point*, a list of the outgoing linkages that is sorted while applying the vertical positioning via the customized *Java* Comparator implementation, is used to iterate over its content.

Within the corresponding loop structure, the first task is to create the BPMN target element using the already known sub-method that also takes care of any additional information like linked tangible objects. Afterward, it is time to handle the potential *Data Flow* for the current edge. As it was done for the first section of this part of the program, the method *handleDataFlow* is invoked while passing on the annotation, as well as the current and the next BPMN node.

Once this task is concluded, another special case has to be dealt with. It is theoretically possible that the target of the current edge requires an *Exclusive Gateway* that has to be mirrored within the remote process to be able to propagate the splitting point. This is done via the application of an *Event-based Gateway* placed within the other process and *Message Catch Events* to receive the decision made. This is basically fine for the most situations except a case, where there is already another *Event-based Gateway* present that is waiting to be linked to the required subsequent *Events*. If the transformation proceeds without any additional steps, this would result in a direct linkage of the two foreign splitting points. But, since this type of *Gateway* has to be followed by an *Event*

(or *Message Receiving Task*), the model would not be valid [OMG11]. So, if this kind of situation has been detected it needs to be handled accordingly. To do so, an additional message flow constellation is put in place using a local *Message Throw Event* and a foreign *Message Catch Event*.

Afterward, similar to the treatment of the *Start Point* it has to be checked whether a merging *Gateway* is necessary on the target's side or not. If so, it has to be created (or simply retrieved if already present) and connected to the present elements.

The last two steps consist within the method call targeting the creation of the linkage between the source and the target of the edge as discussed above and the recursive invocation of the main transformation functionality using the target BPMN element as a parameter.

Next iteration of the Transformation with a Decision Point as source Element
Coming back to the used example in Appendix C, the next Service Blueprint element is also of type *Action* which means that the same program segment as discussed above would be applicable. However, as one can see, the one after that is a *Decision Point* named {decide what to do} which has two outgoing paths amended with the corresponding decision. If this element is passed on to the recursive functionality after it was detected that it is a member of the inner flow of the Service Blueprint, again it is checked whether the node was already visited or not. If it still needs to be processed and the object is identified as a *Decision Point* the last major code segment of the main transformation functionality is invoked.

First of all, since this splitting point was not already dealt with, it is not clear if the necessary local *Gateway* has to be reflected in the other process. So the first step is intended to take care of this uncertainty and resolve it accordingly. To be able to assess the situation correctly, a sub-method is used that recursively checks if the ongoing *Action and Communication Flow* crosses the border of the current process. As a result, the functionality returns a list of all Service Blueprint edges that were found to be subject to this definition. If the collection contains at least one entry, a foreign *Gateway* is necessary. Looking at the example in Appendix C, it can be observed that an outgoing path crosses over to the customer process with the target *Action* {sign approval}. So since the *Decision Point* does in fact have an impact on the other participant, it needs to be reflected.

In this case, to begin with, the *Event-based Gateway* located within the remote process is created. As it was discussed within the section targeting the generation of the graphical elements, the calculation of the diagram part at this point is one of the most complex due to the high potential to inflict conflicts with other model elements. However, once a satisfying positioning has been found, it is necessary to think about the unfinished business that might be present in the other process. If this is the case, it needs to be linked to the newly created splitting point. Note, that at this point an additional *Gateway* might be required to merge multiple incoming paths together. Along with the handling of such elements, there is the chance that another foreign *Gateway* is present that is

waiting to be connected. If so, it needs to be taken care of at this point. A last special case within this context is once more the situation that up until now, the other process may not contain any element, which means that a corresponding *Start Event* is missing and needs to be created and linked to the new *Gateway*.

At this point, the preparations targeting the current Service Blueprinting node, the *Decision Point*, and the additionally required elements are completed and the next tasks are focusing on the outgoing edges representing the specific decisions. Again, a list sorted according to the vertical ordering of its target elements is used as a basis for the loop behavior of this code segment.

The first step within this context is to create or retrieve the already present BPMN representation of the target of the current edge. Afterward, another special case needs to be handled, assuming that for the current situation a foreign *Gateway* is necessary to reflect the local decision since an upcoming path is crossing over. As discussed above, as a counterpart to the local *Exclusive Gateway* an *Event-based Gateway* in combination with *Message Catch Events* is used. If the current edge is part of a path that in fact has an impact on the other participant, we can expect that, at some point in the future, there will be a message flow with a corresponding *Catch Event* that propagates the decision. It will be created automatically while transforming the *Action and Communication Flow* between the source and the target of the specific edge. However, as it is the case within the shown example in Appendix C (decision *NOTHING*), there might be a path that does not inherit such a behavior and comes to an end without affecting the other process. This constitutes a problem since such situations would be missing in the foreign *Pool*, which would result in an incomplete BPMN model.

To handle these cases, after it is detected via a recursive sub-functionality, a new message flow combination using a local *Message Throw Event* and a remote *Message Catch Event* is put in place and linked to the corresponding *Gateways*. Now, for the ongoing processing at this point, there are three possibilities. First of all, it might be the case, that the current path that only has local implications comes to an end without rejoining the other ones. The second variant consists within the situation that it is in fact merged again and continues afterward, but the general flow does not cross over to the foreign process. Last but not least, the local path is joined together with the other ones and, after that, the ongoing *Action and Communication Flow* has an impact on both participants.

Looking at these variants, for the first two, it is not necessary to create a merging point within the foreign *Pool*. In contrast to that, for the last one, this is not applicable. So if this case is detected, a *Gateway* is required within the other process that needs to be linked to the newly created *Message Catch Event*. Depending on the sequence in which the outgoing edges of the *Decision Point* are handled, this merging point might already be present or needs to be set up within this context. If the current situation belongs to the first or second category, the *Catch Event* is simply linked to a new *End Event* that concludes the corresponding sequence flow.

Independent of the situation that was identified above, the further steps are quite similar

to the closing tasks for *Start Points* and *Actions*. First, it has to be determined whether a merging *Gateway* directly linked to the target element is necessary. If so and it does not already exist, a new one is created. Afterward, in accordance to the overall structure of the current point in the Service Blueprint, the sub-method responsible for the proper transformation of the linkage between the source and the target of the current edge is invoked. Once more, the final command consists within the call of the main transformation method with the target element as a parameter.

Transforming the Preparative and Managerial Actions

The paragraphs above discuss the last sections of the recursive function which is responsible for the transformation of the *Action and Communication Flow* including all the contained elements and the linked additional data like tangible objects. After each call has come to an end, looking at the example model, nearly every Blueprint *Area* including its content was transformed into the BPMN methodology. However, still missing at this point, are the *Preparative and Managerial Actions*, as well as the *IT Elements*.

To target this assessment, the first step is to take care of the missing, more general *Actions*. As discussed previously, due their independence of the actual execution phase of the service instance, a separate process is necessary. Accordingly, the task at this point is to create a new BPMN participant and *Pool* and to construct a new sequence flow. The dependencies that are reflected by the Service Blueprint are illustrated using the message flow of the target language.

If *Preparative and/or Managerial Actions* are in fact present within the source model, the first step consists of the iteration of the relevant Service Blueprint flow elements. For each object, the main recursive transformation method that was discussed above is invoked with the current node as a parameter. Since the functionality moves along the pathways of the current diagram part, it is also sufficient to transform this aspect of the source model. However, one amendment that had to be made to fit the current situation concerns the handling of the unfinished business. Looking, at the example model in Appendix C, it can be seen that the *Areas* concerning the *Preparative and Managerial Actions* do not have a *Start and End Point* as well as an end-to-end *Action and Communication Flow*. As a result, each inner flow node that was assessed as having no incoming and/or outgoing edges has to be added to the set of unfinished business. This way the linkage necessary for a valid *BPMN* structure can be added later on.

Once the corresponding *Actions*, the already available linkages, and the dependencies are created within the target model, the opening and concluding elements have to be added. For the model designer, it is not necessary to use both *Areas* within the Service Blueprint. Therefore, the code section tries to place the *Start and End Event* within the *Lane* representing the *Preparative Actions*. If this is not possible, the managerial part is used.

Depending on the number of BPMN flow nodes that do not have any incoming respective outgoing edges, it might be required to add splitting and merging *Parallel Gateways* linked to the *Start and End Event*. Depending on the *Lane* selection for the placement

above, the *Gateways* are positioned in the same way. Looking at the provided example, since there is only one *Action* present within this context, this is not necessary.

To complete this code section, at this point each element that is not part of a consistent sequence flow is linked to the *Start and End Event* respectively the corresponding splitting points.

Transforming the IT Elements
The second missing issue concerns the *Area* that can be found on the very bottom of the example Blueprint and represents the *IT Elements* that are linked to the current model instance. For this purpose, a corresponding sub-method iterates over a list contained in the source diagram using a loop structure. For each contained element a new BPMN *Data Store* is created and added to the logical component of the intended output. However, the graphical counterpart concerning this concept is an issue that occurred during the development. As indicated in the description of the *Camunda Model-API*, not all mechanisms mentioned within the official standard of BPMN are yet fully supported [Ser15a]. The diagram part necessary to represent *IT Elements* cannot be generated up until now using the mentioned library. As a result, the output BPMN model of the transformation is logically complete but does not show the objects created within this section when opened within a graphical model editor. However, as soon as the necessary aspects are published for the *Camunda Model-API* this point can be easily fixed.

Concluding the Transformation
As a very last step during the transformation of the whole source model, a function is invoked that looks for each detected process if there is still some unfinished business present that was not handled at this point. For each loose end that is detected this way, a new generic *End Event* is added and linked to the respective object.

With the execution of these final lines of code, the transformation should create a valid BPMN model that can then be passed on to the creation of a corresponding .bpmn file.

Generating the BPMN File

The output of the *Java* class *TransformationEngine.java* consists within an instance of the *Camunda Model-API* object structure representing a BPMN model. To be able to receive a file that can then be imported into another environment (e.g, graphical modeling tool like the *Camunda Modeler*) and further processed, a standardized file is necessary.

As mentioned before, within this context, the used library offers two very helpful functionalities. First of all, it contains an easy to use validation mechanism that ensures the user that the newly created model is compliant with the official specification of the target methodology. The second aspect consists of a method call that writes the transformation result into the desired *.bpmn* file. The *Java* class *BPMNGenerator.java* encapsulates the call of these tools and is used to take care of this last step within the context of the overall undertaking.

Considering the example model that was used for the corresponding discussions within this chapter, the result of the transformation can also be found in Appendix C.

At this point, the explanation of the actual transformation process is finished and the BPMN output file was created. Accordingly, these paragraphs constitute the last component of the chapter targeting the development of the prototypical implementation. As a next topic, it is time to focus on the conclusion of the overall work.

CHAPTER 6

Conclusion

After the discussions and the development targeting the prototypical implementation of the transformation between Service Blueprinting and BPMN, the last step contributing to the core issue of this thesis which was meant to serve as a proof of concept has been finalized. Now, as a closing statement, this chapter shall represent a conclusion summing up the main findings that were made within the context of this thesis.

The idea of the transformation between Service Blueprinting and BPMN could contribute to the modeling of services and facilitate the way how a corresponding process illustration is created. The main intention is to use an existing Service Blueprint as a source for the conversion process which then automatically generates a valid BPMN representation. Based on the fact that when considering the direction of this mechanism, the starting state inherits a higher level of abstraction than the target, if a comprehensive illustration is required afterwards, the direct result of the transformation has to be amended with some additional, finer grained details that were probably not a subject to the discussions during the design phase. So, although the intention is to automatically receive a valid BPMN model, there might still be the need to refine the output afterward.

This last statement also leads to an important requirement imposed on the result of the conversion. Besides the fact that the process modeling standard is a very well known and widely adopted tool-set maintained by the Object Management Group, one main advantage consists within the XML-file representation available since its version 2.0 [OMG11]. This way it is possible to further process the results while using an additional automatism or even a graphical editor that supports the given format.

Now, looking at the design process of the transformation that was the main aspect of this thesis, each consideration was made while following the direction of the conversion itself. Accordingly, the first major part concentrates on the examination of the source methodology Service Blueprinting. Due to the missing standardization, at this point of the overall work, the underlying literature research has been considered as a very important

task to be able to build up a solid foundation for the necessary definition concerning the actual source of the transformation. In general, the top-level classification of the service-oriented approach makes its distinction based on the illustration of the sequence flow throughout the diagram. Basically, it is possible to do this in an implicit way, while only using the graphical positioning of the contained elements on the canvas, or explicitly via the application of model edges similar to classic flow notations. The big difference between these two categories now consists within the ability to display more complex logical layouts of the action and communication flow between the specific operations. While the implicit approach only supports one dimensional and very simple service orchestrations, its counterpart tends to be more flexible in this regard. Furthermore, when combined with additional conceptional functionalities like decisions and parallel paths, a broader spectrum of service offerings can be covered. However, one important aspect of the source methodology is its comprehensibility, which is especially important for the collaborative design process with business owners that do not have extensive modeling skills. For this purpose its necessary to keep the palette that has to be applied to successfully create a valid Service Blueprint on an appropriate level. After an extensive research and the creation of several example models, the chosen set as discussed in Chapter 4.1.4 and graphically summarized in Figure 5.2 has been fixed. To get a better understanding of how to work with this modeling approach, during the corresponding examinations also a case study was conducted targeting the *Emergency Admission of a Patient at a Surgical Ward within an Austrian Hospital*. The results of the corresponding design process that was carried out as a collaborative approach can be seen in Chapter 3 and Appendix A.

At this point, it is important to note that, although the shown concepts offer more possibilities, most of them are merely optional and shall give the user the opportunity to specify the situation at hand to a satisfying degree. Accordingly, it is possible to create a valid Service Blueprint merely by applying a *Start and End Point*, at least one *Action* and some edges linking the elements together. Concerning the *Areas* which are a key aspect of the methodology, the application of the one for the customer and the *Front-Stage Interactions* of the organisation are sufficient.

After the characterization of the concepts that shall be available to the model designer in the domain of Service Blueprinting, since the notational set provided by the target of the transformation is already very well specified and documented within the official standard, the next logical step that had to be carried out was the design of the actual mapping between the two sides. However, due to its complexity, this cannot be considered as a straightforward task. To be able to come up with a viable solution for the conversion, one has to identify and examine the commonalities and gaps. For this purpose two papers of Simon K. Milton and Lester W. Johnson respectively Yahya Kazemzadeh, Simon K. Milton and Lester W. Johnson that deal with this kind of comparison have been used as a basis for the discussion [KMJ15] [MJ12]. Along the way, the potential coverage of one side by the other has been a core aspect. As a result, besides the mismatch between the available tool-sets (i.e. the available conceptual elements) and the accompanying

different levels of abstraction, the actual nature of the methodologies has been identified as an issue at this point. While Service Blueprinting is a graphical approach to design an organization's service offering from the customer's point of view with a strong focus on the interaction points, BPMN is a process modeling language applied from the perspective of the organization.

Now the transformation has the task to overcome this differences and so create a bridge between the two worlds. At this point, it has to be mentioned that, of course, such an undertaking likely does not have exactly one correct solution for the matching. Instead, there are several possible ways which inherit specific advantages and disadvantages how the converting step can be accomplished. The corresponding discussions were conducted in Chapter 4.1.4 which concludes with a clear recommendation that is also illustrated in Table 4.1 and applied for the remaining parts of this thesis. When it comes to the transformation of one specific concept it has proven to be not sufficient to only focus on the characteristic of this single point. Instead, it is also necessary to review the intended matching within the overall context, since the interdependencies between the individual rules have to be kept in mind as well. So for example, it has to be avoided to convert multiple concepts in a similar way which would cause ambiguity within the output of the mechanism afterward.

With the creation of the set of guidelines as shown in Table 4.1, the actual transformation concept has been completed to a degree that was achievable from a merely theoretical point of view. As mentioned already within the introductory chapter of this thesis, to confirm these considerations a prototypical implementation should be applied that provides the capability to read in an existing Service Blueprint and apply the given set of rules to generate the BPMN output model.

To begin with, during the initial discussions within Chapter 5, three main components have been identified that need to be a part of the planned environment. First of all, as a starting point, there has to be some kind of automatically processable file representation of the source methodology. After a functionality then takes care of its interpretation, the core components should use the extracted information and create a new BPMN model from scratch. As a final step, the task is to export the generated result within a standardized format that can then be processed while using other tools and mechanisms.

To approach the detailed specification of this rough layout and begin with the actual development, as a logical conclusion, the direction of the transformation has been kept in mind while initiating the necessary work. So at this point, the way how a Service Blueprint could be created and stored within a readable file structure has been examined. In general, for this purpose, some kind of computerized tool has to be utilized. Sadly, up until now, compared to well-structured languages like BPMN, the software support is not that well developed for this aspect. Although, some applications have been found during the corresponding research, as discussed within Section 5.2, they were not assessed to fulfill the requirements that have to be met to be applicable as a part of the transformation environment. To overcome this situation, basically there are two alternative approaches. First of all, it could be a possibility to look for an already existing framework that can

be amended and configured accordingly. If this is not a viable option either, as a last resort, a completely new tool designed for the creation of Service Blueprints could be developed. After some digging, the easy-to-use modeling environment *Draw.io* has been found. To satisfy the demands within this context, the tool enables the user to specify customized libraries where it is possible to use individualized shapes to create a new approach. Accordingly, the task was to introduce a new set of conceptual elements that match the specification made within the theoretical part of this thesis. But, of course, for the tool representing the source of the transformation, it is not enough to be able to fulfill this role. There is also the need for a file export of the created model that can then be interpreted afterward. For this purpose, *Draw.io* offers several different formats including an uncompressed XML version. Additionally, while assembling one's library, the corresponding style element of the specific concepts can be amended which positively influences the automatic interpretation of the model afterward.

The next aspect that has to be considered when looking at the technical set-up targets the other side of the transformation. Within this context, it is not necessary to look for a tool that creates the models since the conversion mechanism has to take care of this task. But, as indicated above, there is the requirement to have a BPMN output file that can be processed while applying additional tools. For this purpose since its version 2.0 the official standard references an XML specification that can be used at this point. Every tool that has the ability to read in such a *.bpmn* file can then be applied to edit the results of the transformation. The *Camunda Modeler* has been found to be compatible with this specification and was mainly used to test the generated outputs.

Considering the way how the prototype shall be actually created, what still is missing is the corresponding development environment. To accomplish this task, the programming language *Java* in combination with the *Eclipse IDE*, *Git* via *Bitbucket* as version management and *Maven* for the dependency management have been applied.

However, before the actual transformation can be implemented, from a technical point of view, it is necessary to consider the way how the desired file can be created. Within this context, the *Camunda Model-API* has been found to be very helpful, since it offers a predefined BPMN *Java* object structure and the possibility to export the resulting orchestrations into the desired format. So what has to be done, is the creation of a mechanism that interprets the customized Service Blueprint XML files provided by the *Draw.io* export and convert the outcome into the given BPMN *Java* representation. After that, the mentioned functionality that takes care of the required output generation can be invoked.

Another general consideration that has to be taken care of is the important point of the validation of the transformation results. Only if this aspect can be satisfied to a reasonable degree, it can be assessed whether the implemented prototype serves its purpose as a proof of concept or not. To target this demand, during the development of the implementation, three different methods have been applied. First of all, once more the integrated *Camunda Model-API* can be used to approach this topic. It offers a method that validates the newly generated logical BPMN model present within the *Java*

structure mentioned above, against the BPMN 2.0 specification[Ser15b]. As a result, every file that is successfully created via the transformation mechanism is compliant with the official standard. But of course, this can only be considered as a top-level requirement within this regard. What is still missing concerns the validity of the results when looking at the proposed matching rules between the two methodologies. To resolve this situation, during the earlier stages of the development, the focus was on the review of the generated XML background structure (i.e. the *.bpmn* file structure). To be able to assess if the output fulfills the requirements, the graphical editor *Camunda Modeler* has been applied to create a template matching the current test case. However, although this might be a practical approach to test smaller to medium sized test Blueprints when it comes to larger ones, it is not that useful. To overcome this issue, again the well known BPMN modeling tool has been used, but this time not for the generation of test samples, but for the actual graphical interpretation of the conversion results. Once the corresponding model has been successfully created and imported (i.e. without receiving error messages provided by the embedded library and the editor itself), the graphical orchestration can be used to check if the transformation rules were applied correctly. In total, about 70 different test models ranging from smaller to more comprehensive ones have been created to aid the development and the validation of the results.

However, due to some difficulties concerning the graphical representation of the BPMN model, the application of the *Camunda Modeler* to import the transformation output, as proposed for the third validation approach, was only possible after the majority of the core functionality has been implemented. In general, considering the corresponding XML structure, a BPMN file consists of two major sections. The logical part represents the conceptual elements and their functional dependencies and is basically sufficient to outline a valid BPMN model. However, this way it is only an option to use automated functionalities to process the results afterward. The usage of a graphical editor like the *Camunda Modeler* is not possible. To overcome this limitation, the diagram component needs to be created as well and linked to the first section of the model. Since a corresponding, already existing library has not been found, this task had to be implemented from scratch. In general, the processing of the graphical information while using a mechanism like the intended prototype is not an easy task since a multitude of aspects has to be considered like the conflict management. To cope with this complexity, the corresponding development steps have been initialized after the core functionality targeting the logical counterpart has been put in place.

But this was not the only critical aspect that has to be dealt with during the development of the prototype. A major issue already occurred during the step-by-step implementation of the transformation targeting the logical component of the resulting BPMN model. Within the context of the theoretical concept, as discussed in Chapter 4.1.4, the way how the areas as proposed by Service Blueprinting can be correctly transferred into the world of BPMN has been a key point. As shown in Table 4.1, the decision has been made to apply two main processes (i.e. *Pools*), one for the customer and one for the organization. *Lanes* are then applied to convert the remaining separators. Additionally, if *Preparative*

and/or Managerial Actions are present in the source model, a third process is added. This option has been chosen since it offers the best distinction between the two major participants of the service situation. However, when following this assumption, it has to be considered that the unified *Action and Communication Flow* as present within the conceptual set of the source methodology, this cannot be the case for the target environment. The usage of multiple *Pools* in BPMN requires that each process has an encapsulated sequence flow (including start and end nodes) and it is only possible for the participants to interact with each other while applying the provided message notation. Accordingly, during the development of the transformation mechanism, it was necessary to think about a way to split the given *Action and Communication flow* and distribute it across multiple BPMN *Pools*. This is not an easy undertaking since, besides the necessary synchronization of the contained flows, for some specific elements like *Decision Points* and parallel paths a new flow constellation has to be created automatically to be able to mirror the situation for all participants and so generate a valid BPMN structure that is able to reflect the actual service situation at hand. Since this aspect has a strong impact on the general applicability of the transformations rules, especially when considering model constellations that include different combinations of such critical situations, this issue was one of the major sources of complexity during the development of the prototype.

As indicated above, once the logical component had been dealt with to a satisfying degree, the creation of the graphical elements was approached. To be able to provide a BPMN diagram representation that inherits a similar look to the Service Blueprint and since the corresponding coordinates and measures had to be calculated by the newly designed mechanism, the graphical information that is a part of the XML export of *Draw.io* has been reused. While doing so, already after the review of the first transformation results, conflicting model constellations occurred. The reason for this can be traced back to the very nature of the two methodologies. While Service Blueprinting is a more abstract approach that tries to provide the user with a manageable core set of shapes, BPMN, on the other hand, tries to capture the underlying processes with an extended level of detail. Additionally, the requirement for the transformation to split the unified *Action and Communication Flow* and capture the situation while using at least two distinct sequence flows encapsulated within the corresponding *Pools*, has an impact on this point. As a result, the target methodology requires more conceptual elements to be able to represent the service, which of course, potentially stand in conflict to each other when applying the graphical details extracted from the source model.

After some experimenting, two approaches have been implemented to overcome this issue. The first logical step would be the introduction of some conflict resolution mechanisms. However, as it has been observed after various test runs, the encountered problems could not be solved to a satisfying degree. The reason for this lies within the basic cause of the conflicts which is the different amount of conceptual elements required for the creation of a valid model. Now since the transformation input is represented via Service Blueprinting and it cannot be expected from the users to consider this aspect during the design phase, the density of the source, which is indicated by the number of graphical elements that

are positioned inside the specific areas, proofs to be too high in most cases. Due to this fact, when it comes to the placement of the additional shapes as required by BPMN the target area tends to be too crowded to find a non-conflicting way even after the application of simple resolution mechanisms. As a countermeasure, the main idea was to use a preprocessing mechanism that reads in the Service Blueprint and decreases the measurable density of the model. To do so, a corresponding component of the prototype can be configured using two factors, one for each dimension, which is applied if the calculated indicator exceeds a specific threshold. As a result, the source model inherits enlarged areas with increased spacing between the contained elements. Now when it comes to the graphical transformation towards BPMN, while executing this preliminary mechanism, many potentially conflicting situations can be avoided. But nevertheless, there is still the need to perform some resolving steps within the context of the main transformation component to be able to take care of issues that occur despite the resizing of the original source model. With the combination of both approaches, it was finally possible to generate a satisfying graphical output of the mechanism.

At this point, the general concept of the transformation between Service Blueprints and BPMN has been discussed from a theoretical as well as more practical point of view. During the development of the proof of concept, the initial suggestion was extended with complementary aspects like the splitting of the unified *Action and Communication Flow*. Since the various test runs using the prototypical implementation led to results that have been assessed as valid according to the integrated validation mechanism and the checks for compliance against the set of designed rules, the conversion as proposed within the context of this thesis has been confirmed.

However, what still remains to be addressed concerns the definition of the target that was set in the introductory chapter. For the sake of a clear focus, when thinking about the work contained in this thesis, the following scientific questions have been stressed at the beginning.

- What is the common basis of the concepts Service Blueprinting and BPMN (especially with respect to the linkage between both methodologies)?
- What are the limitations for the creation of a semi-automatic transformation between the two sides?
- Is it necessary to introduce more formal characteristics or restrictions to Service Blueprints to be able to aid the transformation and if yes, what are they on a conceptual basis?
- When considering today's business process modeling, are there any enhancements that could be introduced to Service Blueprinting that might aid the overall development?

To target the first point, as an initial step, Chapter 2.5 concentrates on the related literature and discusses the corresponding findings. Due to the very specific application

(i.e. combination) of the two methodologies, this section is completed via the subsequent examination focusing on the additional aspects targeting the commonalities and differences with respect to the given conceptual and notational sets that were determined beforehand. In general, this part represents an important basis for the design of the transformation concept.

Going on to the second question which is about the limitations that have to be introduced to the environment to be able to successfully perform the planned conversion, the answers are a core component of the discussions concerning the transformation concept and the subsequent work targeting the proof of concept. Within this context, probably one of the most important aspects is related to the different level of abstraction inherited by the two modeling languages. Since the direction of the transformation can be considered as top-down, it is simply not possible to generate every desired detail within BPMN. As a result, the output of the mechanism may need some additional refinements to fulfill the demands of the model designer in the end.

Another limitation that shall be mentioned at this point is also related to the third scientific question and consists within the requirement that for Service Blueprinting it is necessary to have some kind of fixed conceptual palette that is applied while using a computerized tool for the creation of the models. Due to the missing standardization, there are several approaches that could be taken into account. However, for the context of this work, a specific set had to be applied. As mentioned above, a summarized overview can be seen in Figure 5.2.

Coming to the last scientific question, besides the well known notational elements of Service Blueprinting that were also present in the referenced literature, additional ones have been included. Within this context, the decision whether a certain extension should be introduced or not was based on shortcomings of the overall approach which were mentioned within the literature, or it was perceived as having the potential to positively contribute to the applicability of the language. As a result, the possibility to integrate the flow of data artifacts throughout the service while applying annotations to the flow edges is also a new part of the methodology's palette as the usage of customer expectations, satisfactions. Section 4.1.4 specifically focuses on these extensions and their transformation towards BPMN.

With these last paragraphs summarizing the findings targeting the scientific questions stated in the introductory chapter of this thesis, the general undertaking can be considered as complete. While taking certain prerequisites and constraints into account, it has been shown, that it is possible to perform the transformation between Service Blueprinting and the Business Process Model and Notation. However, one thing that should be targeted as a final part is the issue of potential further developments that could be introduced to develop the concept even further.

CHAPTER 7

Future Developments

Having reached the end of the steps necessary for the development of the first transformation from Service Blueprinting towards BPMN, along the way several aspects have been identified that could also contribute to the relevance and applicability of the overall idea but did not fit the scope of this work. The following paragraphs shall be used to discuss these points.

First of all, the work performed within the context of thesis had a strong focus on the design of the conversion and the creation of proof that the approach is applicable while using the intended prototype. Now that this undertaking can be considered as being successful, within a further step it has to be examined if the proposed transformation rules are also perceived as the best alternatives when it comes to other real-life situations besides the one illustrated within the case study of this work. For this purpose, it is advisable to select several service situations of different areas and with different levels of customer interaction and customization and apply the designed modeling environment.

Another aspect that could be targeted as a further development concerns the level of abstraction of the source methodology of the transformation process. Within the context of this thesis, it was assessed, that it is necessary to take a very specific set of notational elements into account that forms the basis for the transformation rules. While doing so, the preference was more on a conceptual set that is based on well-known flow charting approaches which makes it possible to integrate more details if necessary to designing enhanced service flow constellations. However when looking at the more classical approaches of Service Blueprinting as proposed within the literature [LW11] [BOM08] [KMJ14], it could be also interesting to add the possibility to interpret such an abstract conceptual set to the conversion mechanism as well. As a result, the output BPMN model would also inherit fewer details and require a greater effort of the model designer to finalize the process representation afterward. Nevertheless, since there is no fixed standard for this source language, the option to rely on different notational sets that can be used for the transformation, could positively contribute to the versatility of the overall idea.

The next topic that shall be mentioned within this context concerns the way the source model is actually created before it is passed on to the conversion mechanism. For the context of this thesis, the free-to-use modeling tool *Draw.io* is extended with a customized library that enables the creation of valid Service Blueprints. However, when compared to more specialized software, like the *Camunda Modeler*, the application does not restrict the user when it comes to the placement and linkage of the notational elements. As a result, it offers a quite extensive freedom during the design phase which leaves it to the interpretation component targeting the necessary input files to highlight potential errors afterward. Thinking about the usability of the whole environment, and not just the modeling tool, it could be a good idea to introduce some restrictions already during the creation of the Service Blueprint. Thinking about the next steps, it has to be discussed and investigated if this topic should be considered and to which degree. After all, one aspect that can be seen as a source of the methodologies popularity directly refers to its ability to be applied in collaborative and creative environments. A limitation that proves to be too extreme might, therefore, stand in conflict to the general applicability.

Targeting this aspect of the general environment, another approach is theoretically possible and could be subject to ongoing research. When it comes to the area of service and process design, independent of the actual methodology that is applied, as a first approach the utilization of whiteboards and sticky notes is very common [SS16]. Although a next step often consists within the usage of some kind of modeling application, it could be a convenient step to automatically transfer the information captured while using such an analog method towards the digital counterpart. This could be accomplished via image recognition software that interprets a picture of the corresponding service draft and automatically generates a computerized model. Due to its nature, this approach can be considered as preliminary to the actual transformation between Service Blueprinting and BPMN.

Now, when considering the other side of the conversion, also for BPMN it is possible to think about aspects that could be discussed and may contribute to the value of the overall environment. Due to the fact that the process modeling language is strongly focused on the illustration of given situations from the organizations perspective, it can be argued that there is a shortcoming when it comes to the representation of characteristics critical for the customer. In general this may not be a big issue for the methodology itself, however when looking at the embedding within the discussed service design framework, one could think about ways to extend the approach. Since BPMN is a fixed, official standard, this might not be easy and have to be done with great care to ensure the alignment of any additional concepts with the already existing ones.

The last remark that shall be noted within this context concerns the actual transformation mechanism itself. As already mentioned in the introduction of this thesis, the intention of this work was to develop a prototype that shall sever as a proof of concept. Although the program itself was developed to fit most model constellations and also emphasize extensibility and maintainability, if it comes to the introduction of the component into a real-life application it could be a good idea to rethink some technical decisions made

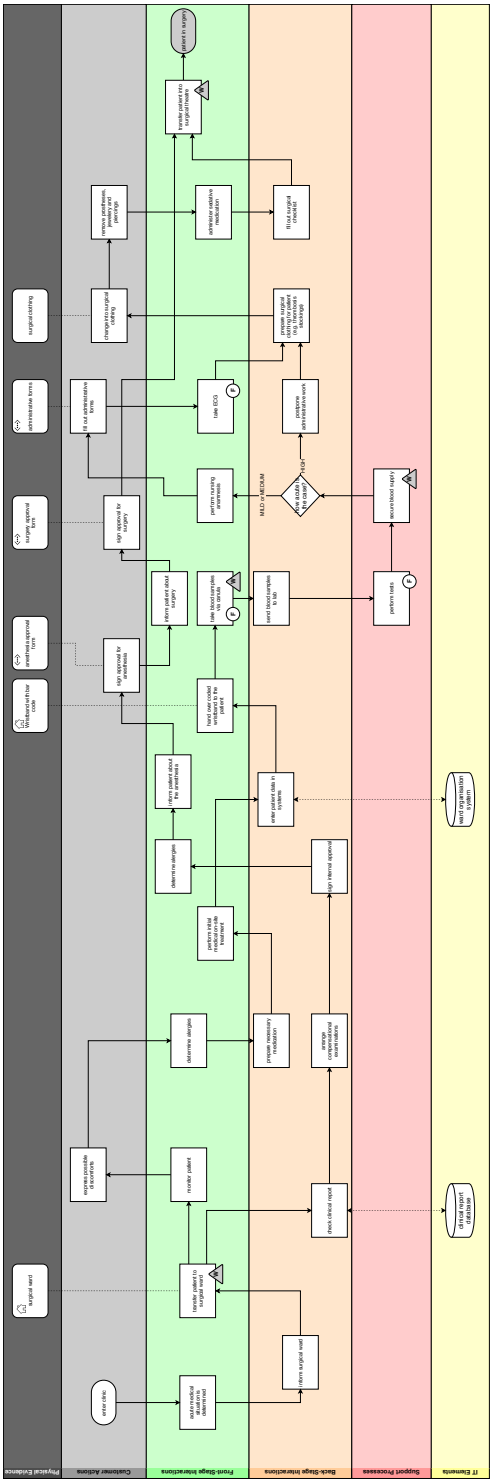
along the way. One aspect that shall be stressed at this point concerns the way how the actual transformation rules are handled directly within the code. Up until now, the corresponding instructions are embedded within the very core element of the prototype. Basically, it is possible to think of a way to elevate the rule definition on a higher level which would make it easier to maintain the set in the long run. However, whether such a solution is feasible with respect to the faced complexity concerning the transformation of the *Action and Communication Flow* and the creation of the diagram part of the final output or not has to be assessed first.

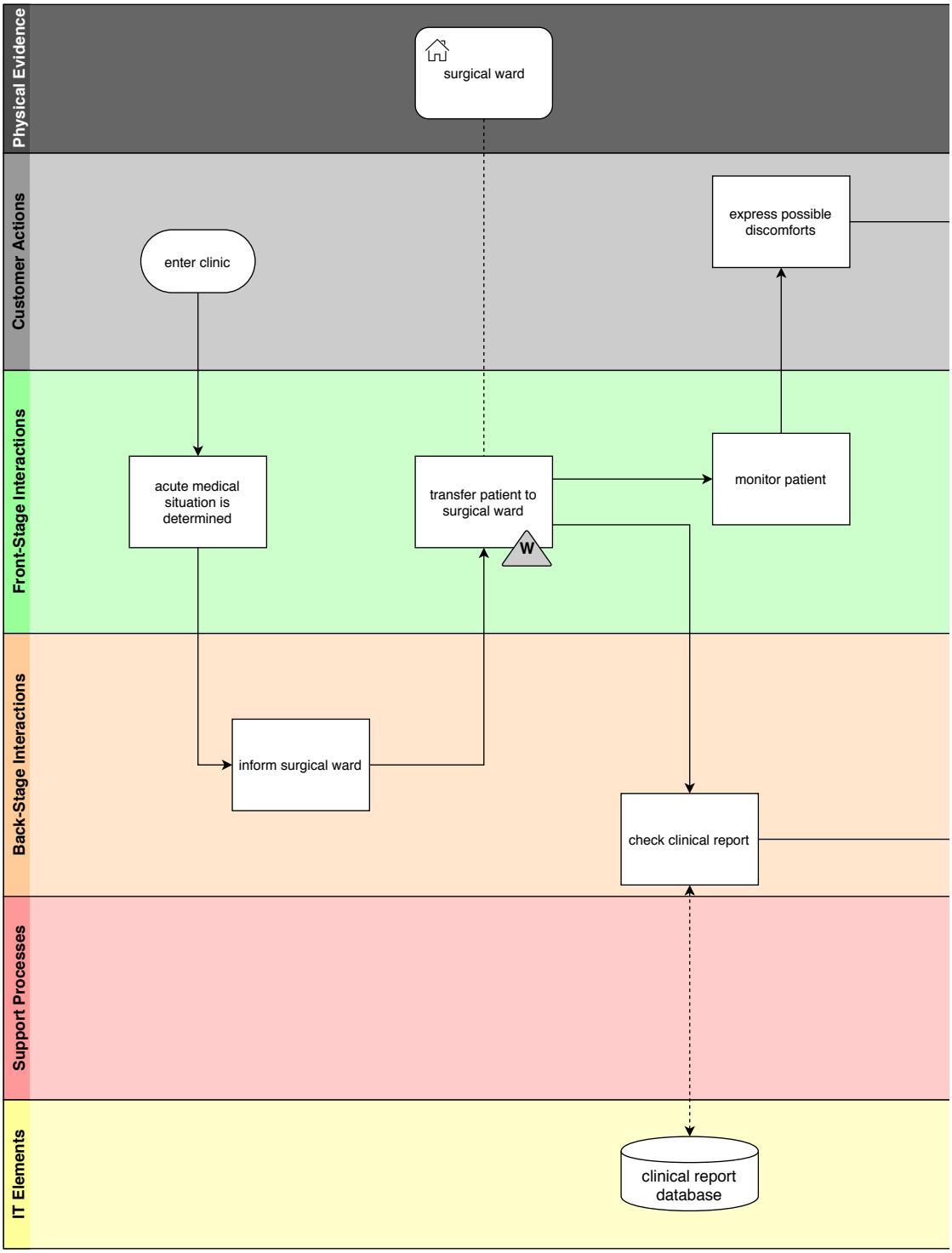
The discussion concerning the possible further steps that could be performed to extend and enhance the combination of Service Blueprints and the Business Process Model and Notation builds up the final component of this theses. This is now complete as well and so this work comes to an end.

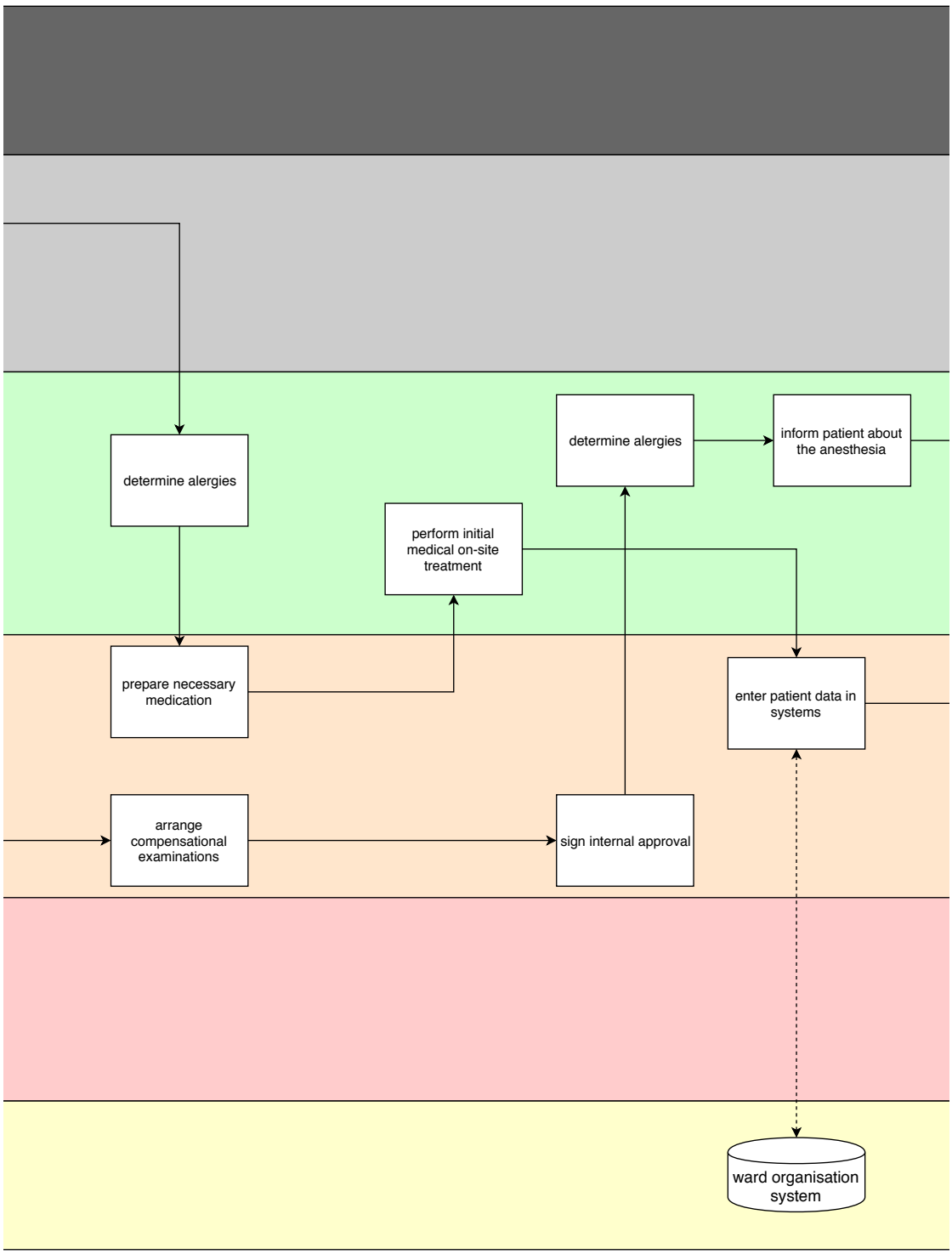
APPENDIX A

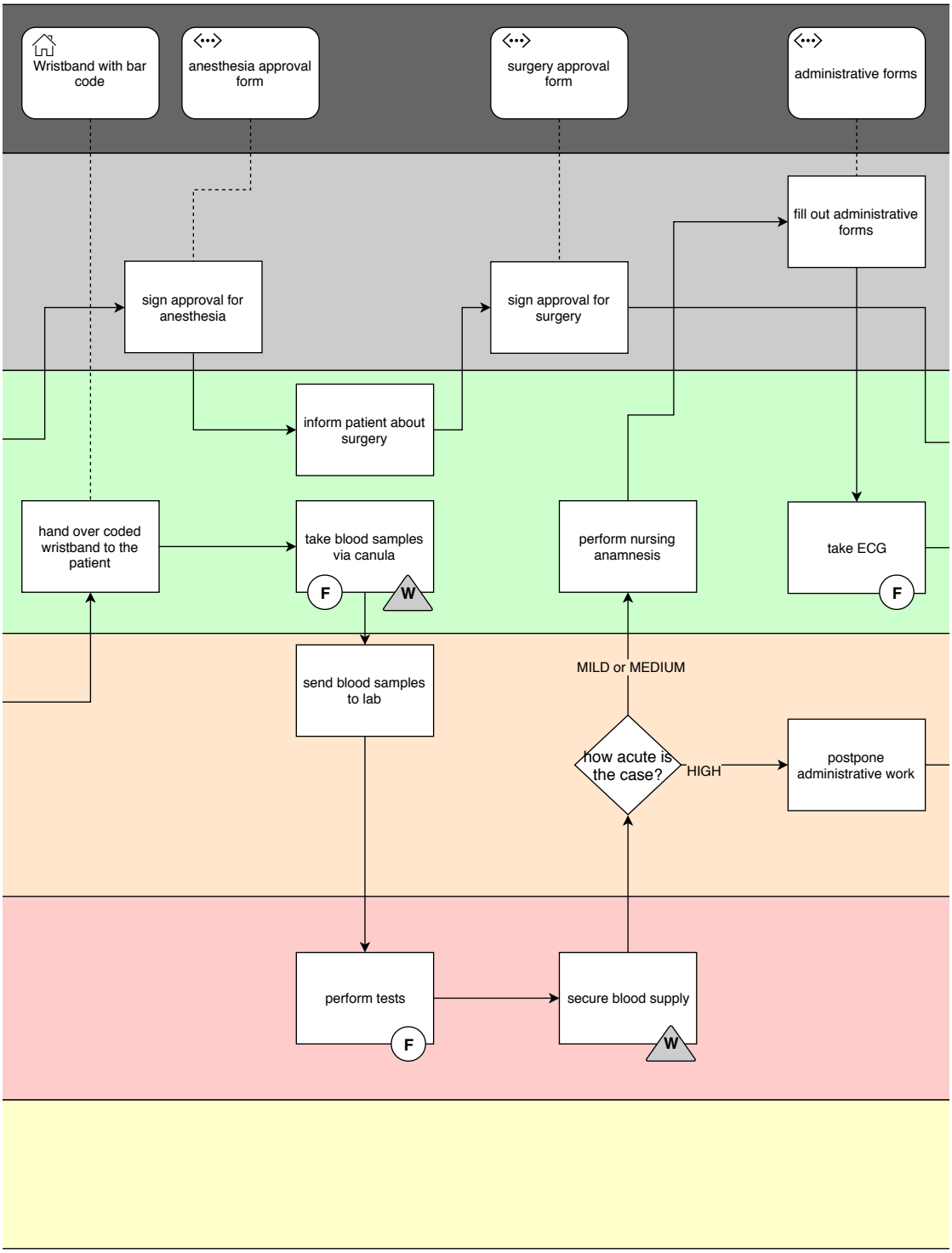
Case Study: Emergency Admission of a Patient at a Surgical Ward within an Austrian Hospital

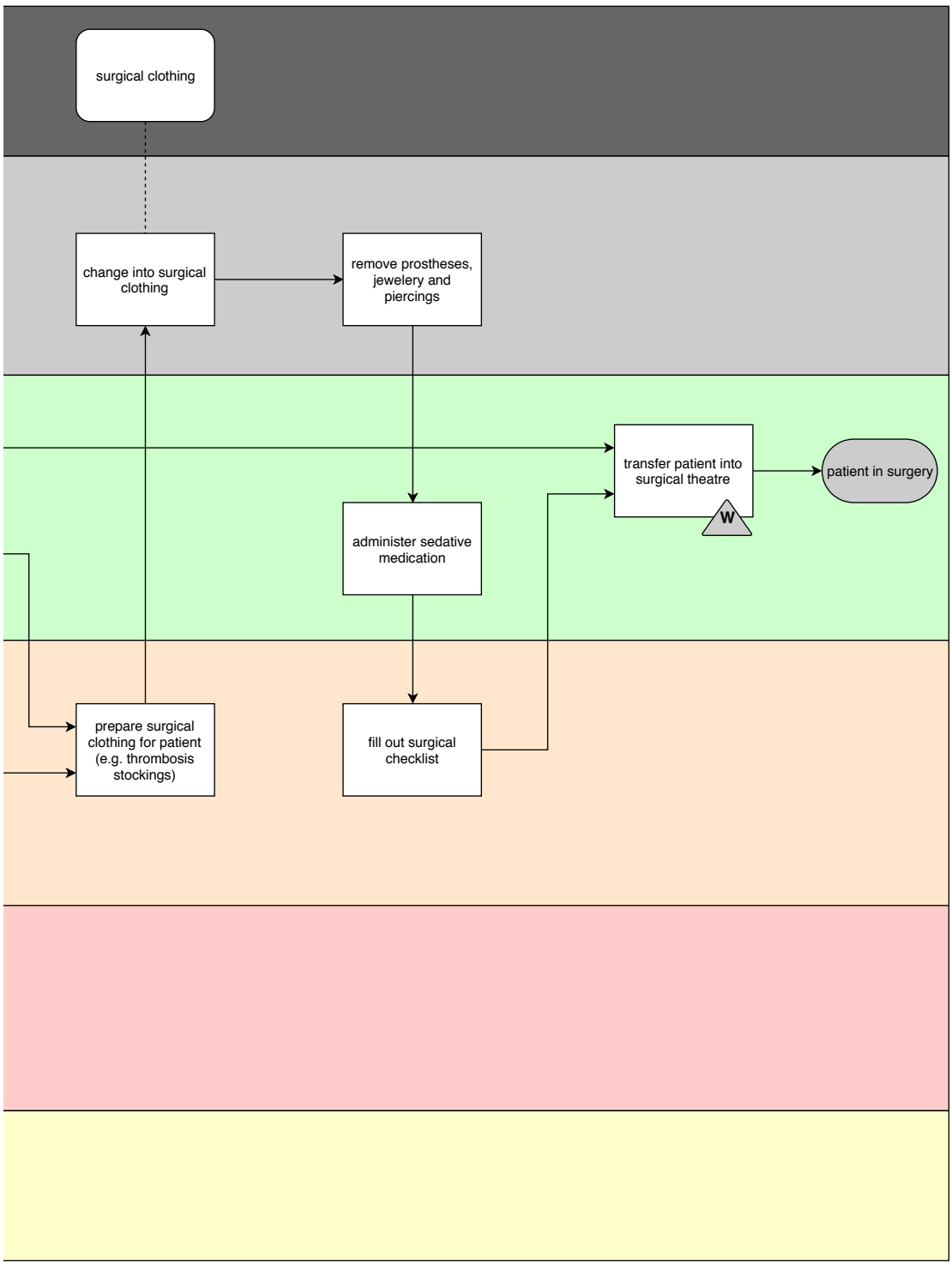
A.1 Abstract Service Blueprint



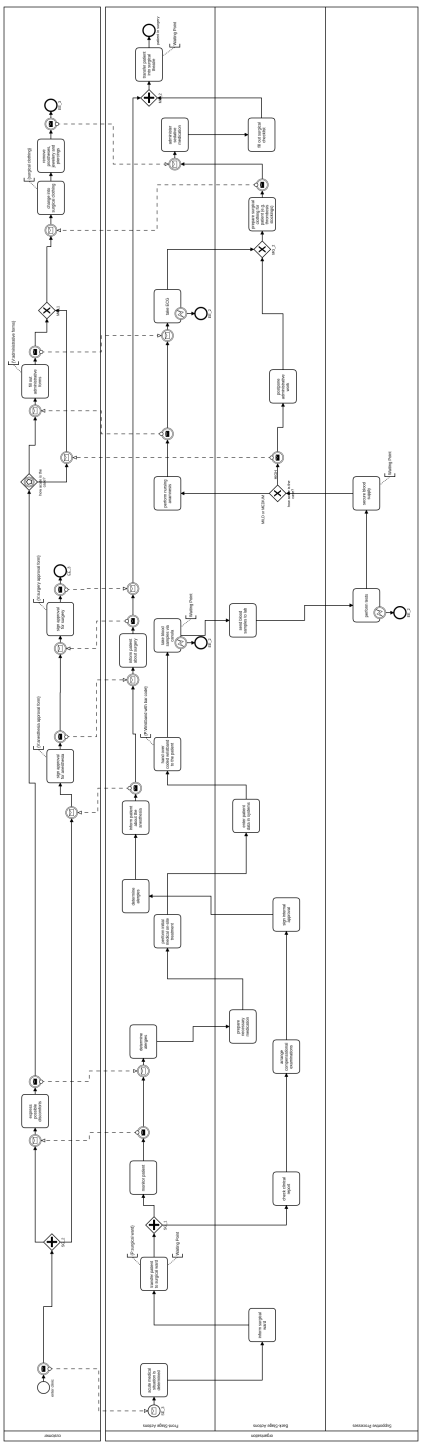

Physical Evidence
surgical ward
Customer Actions
enter clinic
express possible discomforts
Front-Stage Interactions
acute medical situation is determined
transfer patient to surgical ward
W
monitor patient
Back-Stage Interactions
inform surgical ward
check clinical report
Support Processes
IT Elements
clinical report database


determine alergies
determine alergies
inform patient about the anesthesia
perform initial medical on-site treatment
prepare necessary medication
enter patient data in systems
arrange compensational examinations
sign internal approval
ward organisation system

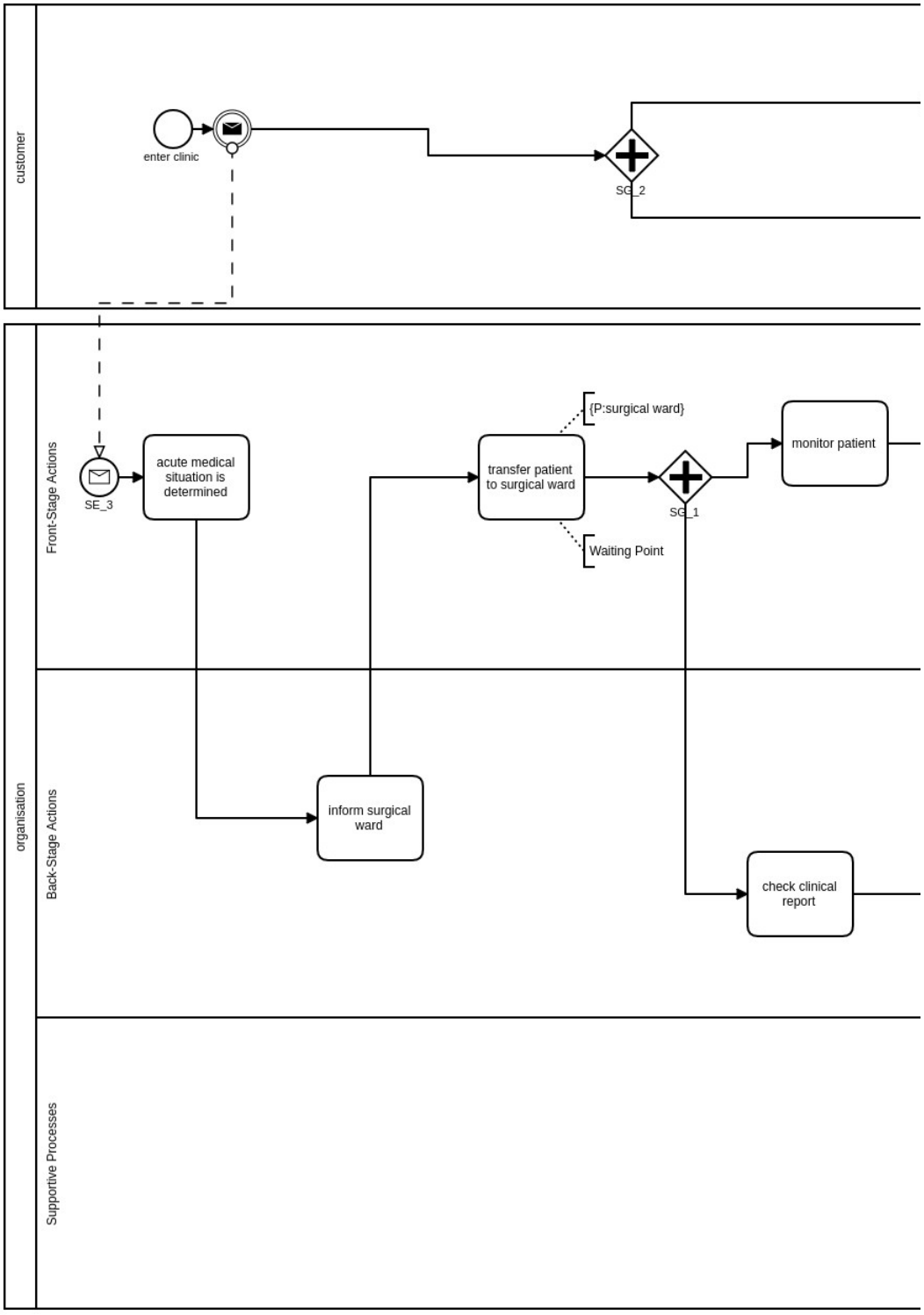

Wristband with bar code
anesthesia approval form
surgery approval form
administrative forms
fill out administrative forms
sign approval for anesthesia
sign approval for surgery
inform patient about surgery
hand over coded wristband to the patient
take blood samples via canula
F
W
perform nursing anamnesis
take ECG
F
send blood samples to lab
MILD or MEDIUM
how acute is the case?
HIGH
postpone administrative work
perform tests
F
secure blood supply
W

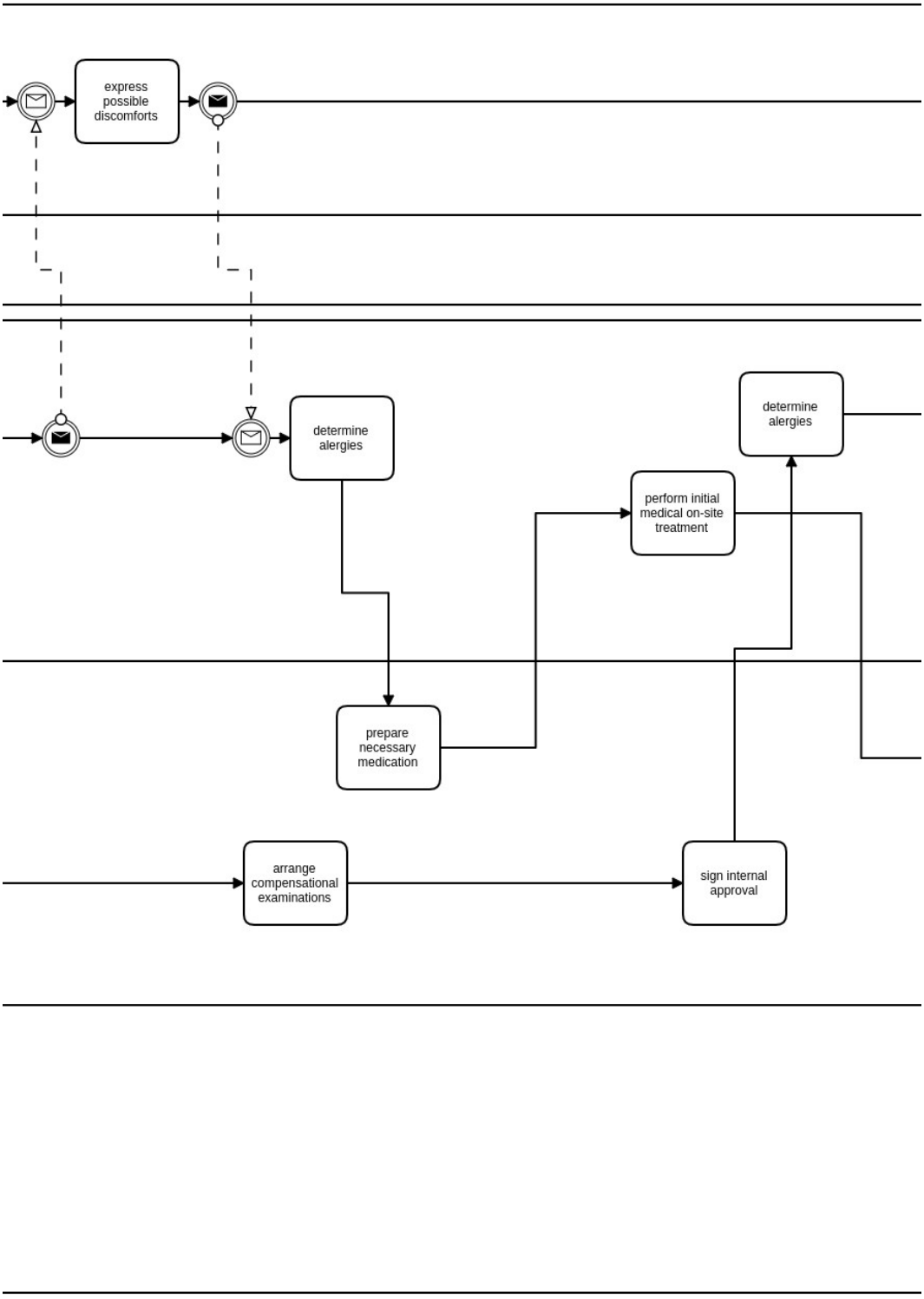

surgical clothing
change into surgical clothing
remove prostheses, jewelery and piercings
transfer patient into surgical theatre
W
patient in surgery
administer sedative medication
prepare surgical clothing for patient (e.g. thrombosis stockings)
fill out surgical checklist

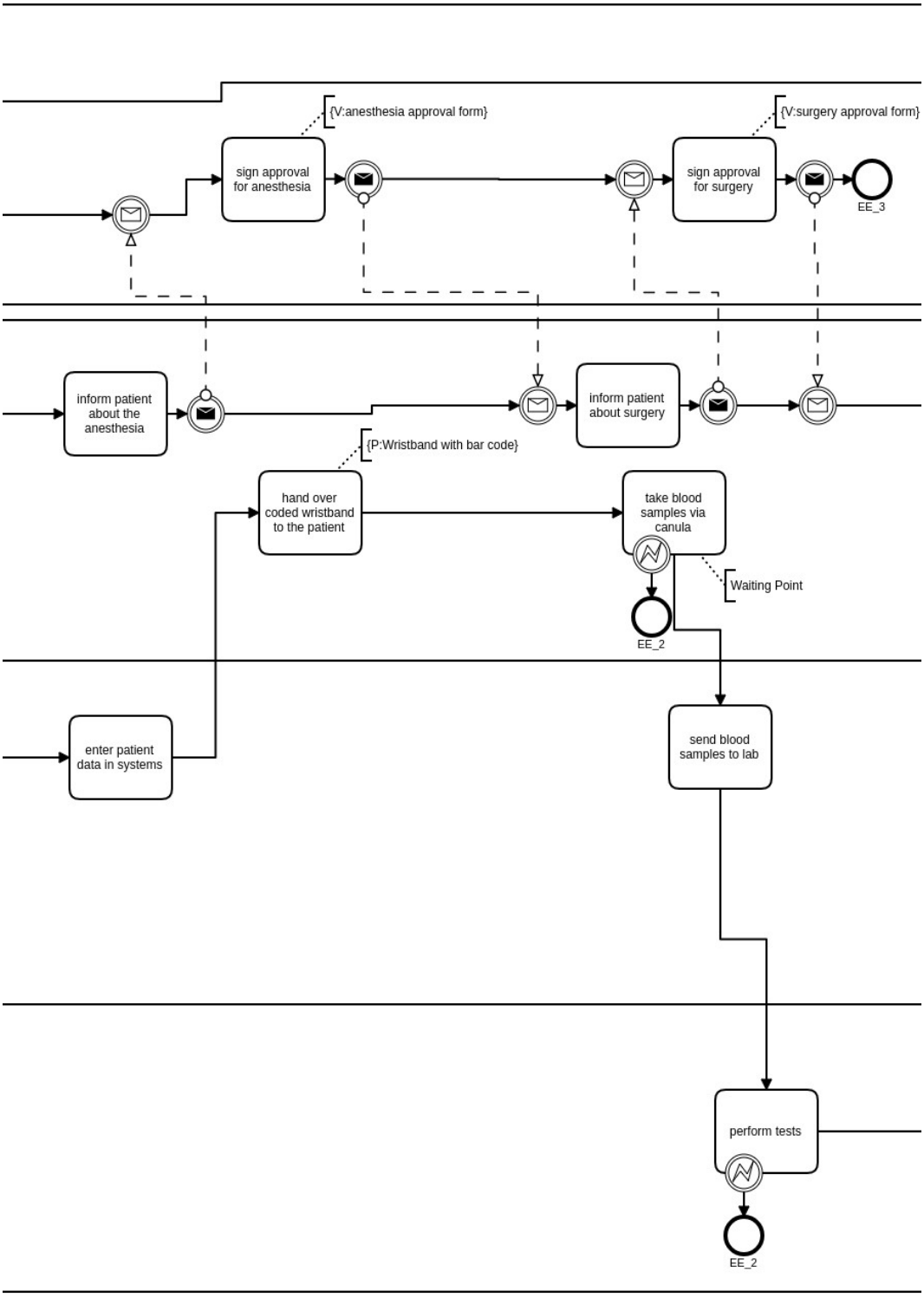
A.2 Transformation Result of the Abstract Model[1]

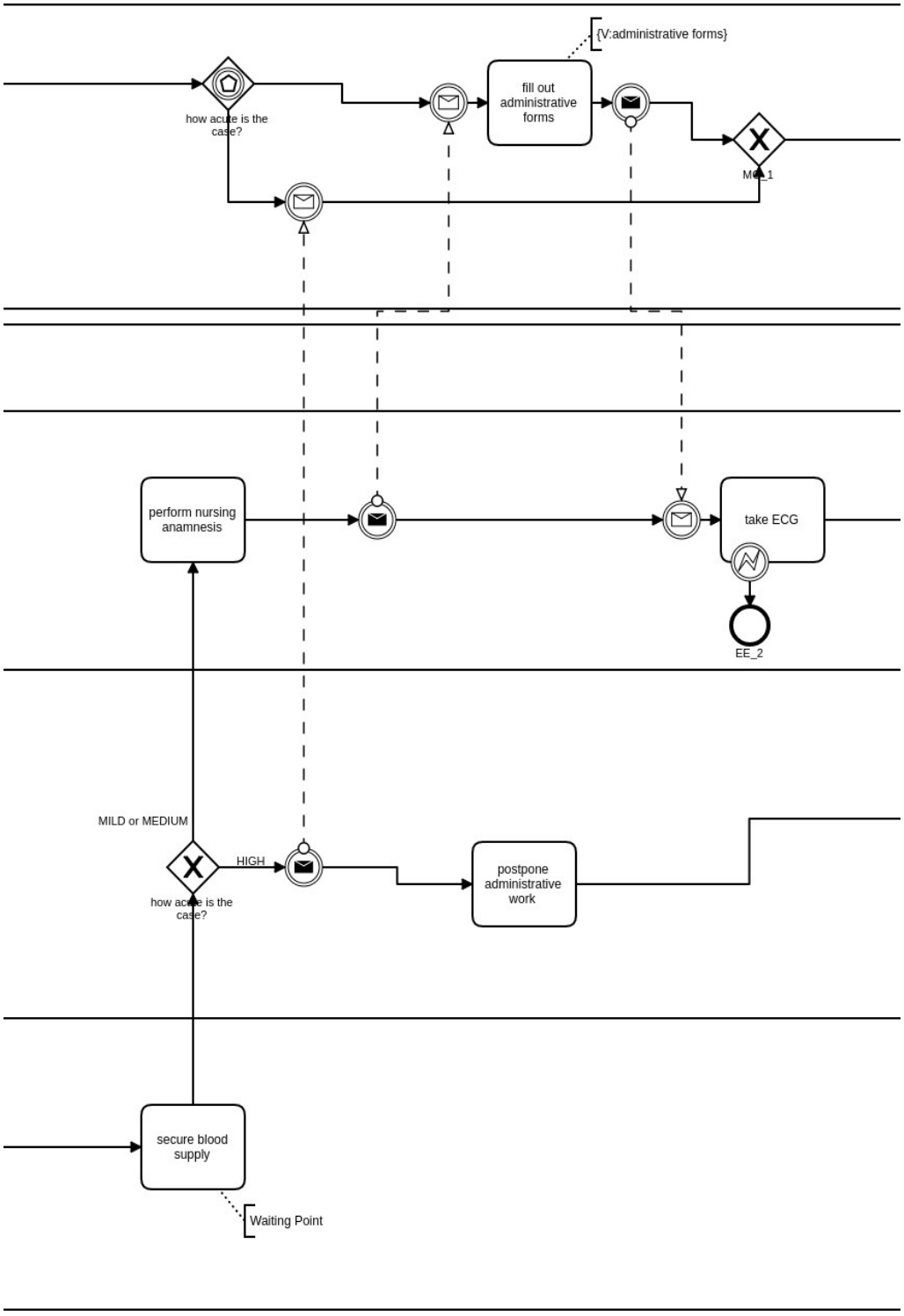


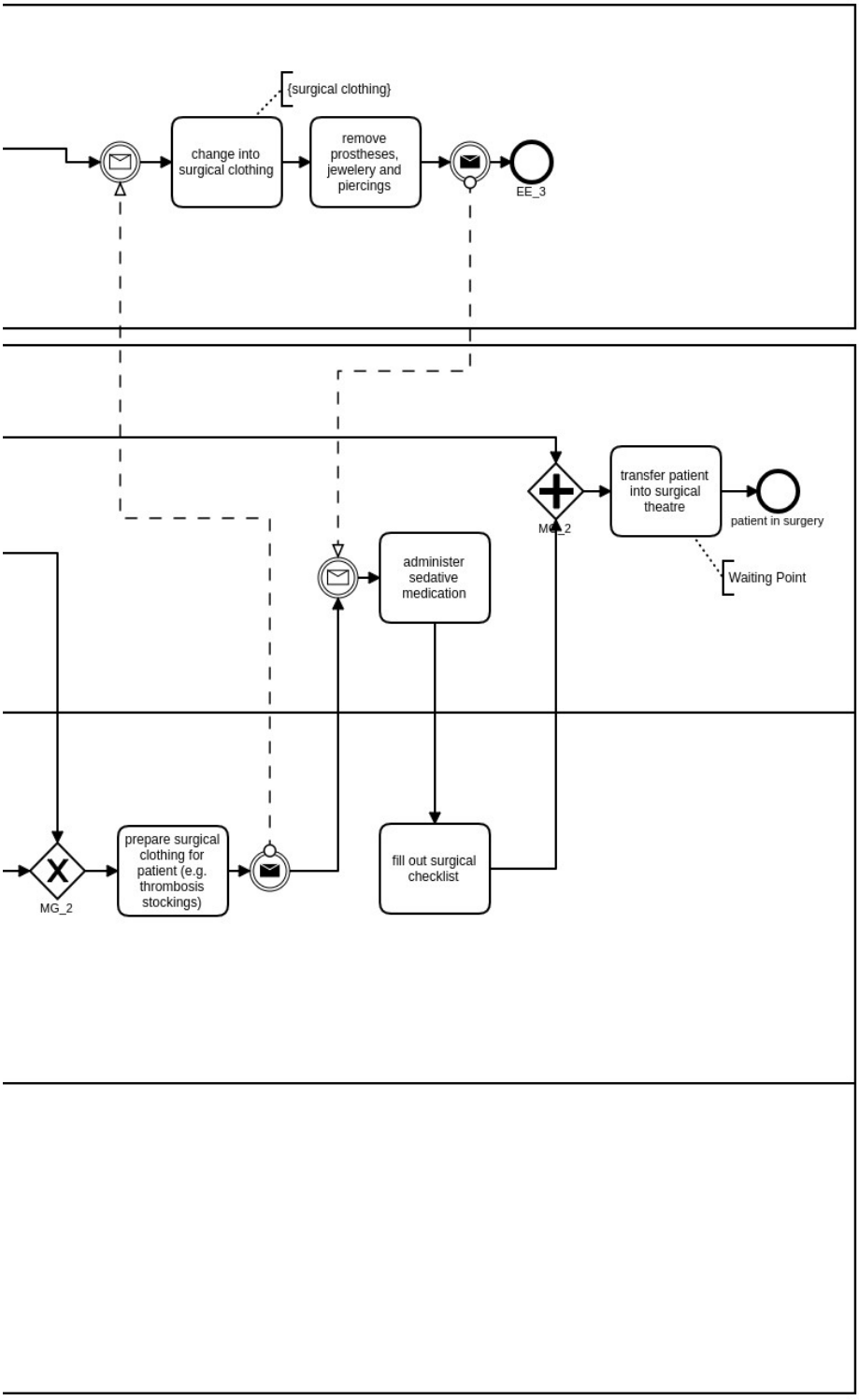
[1]Please note that the illustration was slightly manually edited to fit the layout of these pages


customer
enter clinic
SC_2
organisation
Front-Stage Actions
SE_3
acute medical situation is determined
{P:surgical ward}
transfer patient to surgical ward
SC_1
monitor patient
Waiting Point
Back-Stage Actions
inform surgical ward
check clinical report
Supportive Processes

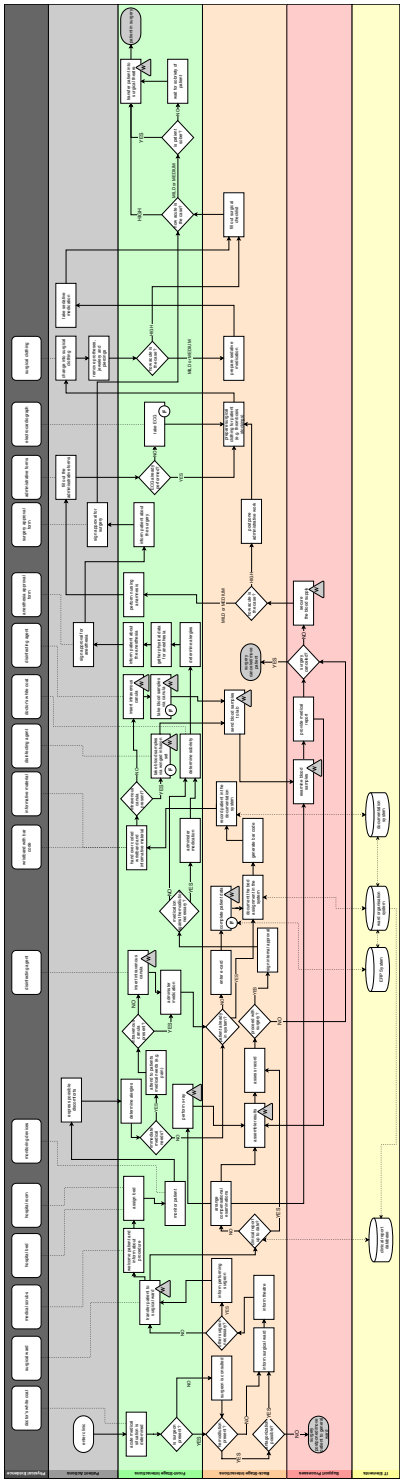

express possible discomforts
determine alergies
determine alergies
perform initial medical on-site treatment
prepare necessary medication
arrange compensational examinations
sign internal approval

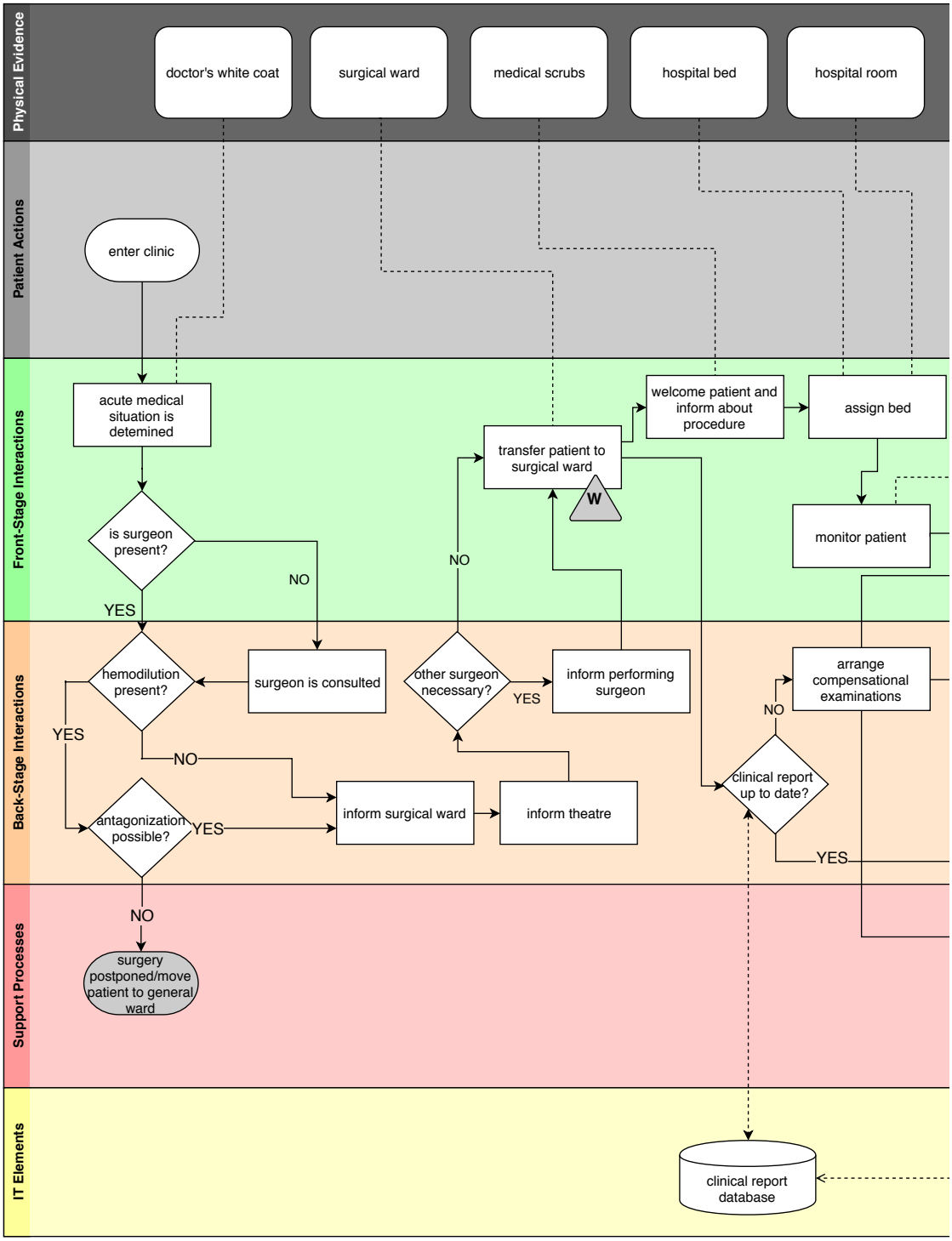

{V:anesthesia approval form}
sign approval for anesthesia
{V:surgery approval form}
sign approval for surgery
EE_3
inform patient about the anesthesia
inform patient about surgery
{P:Wristband with bar code}
hand over coded wristband to the patient
take blood samples via canula
Waiting Point
EE_2
enter patient data in systems
send blood samples to lab
perform tests
EE_2

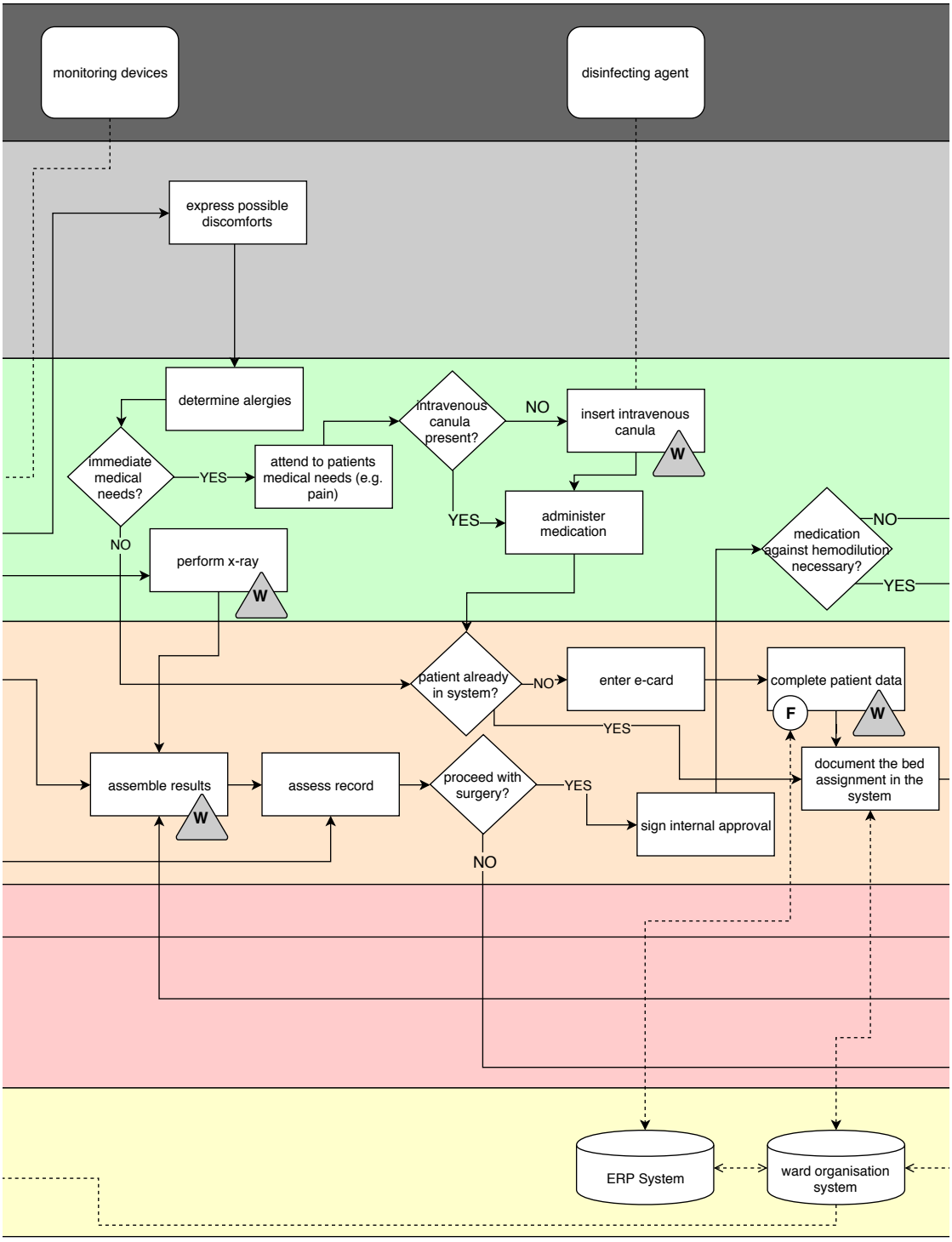

{V:administrative forms}
fill out administrative forms
how acute is the case?
perform nursing anamnesis
take ECG
EE_2
MILD or MEDIUM
HIGH
how acute is the case?
postpone administrative work
secure blood supply
Waiting Point

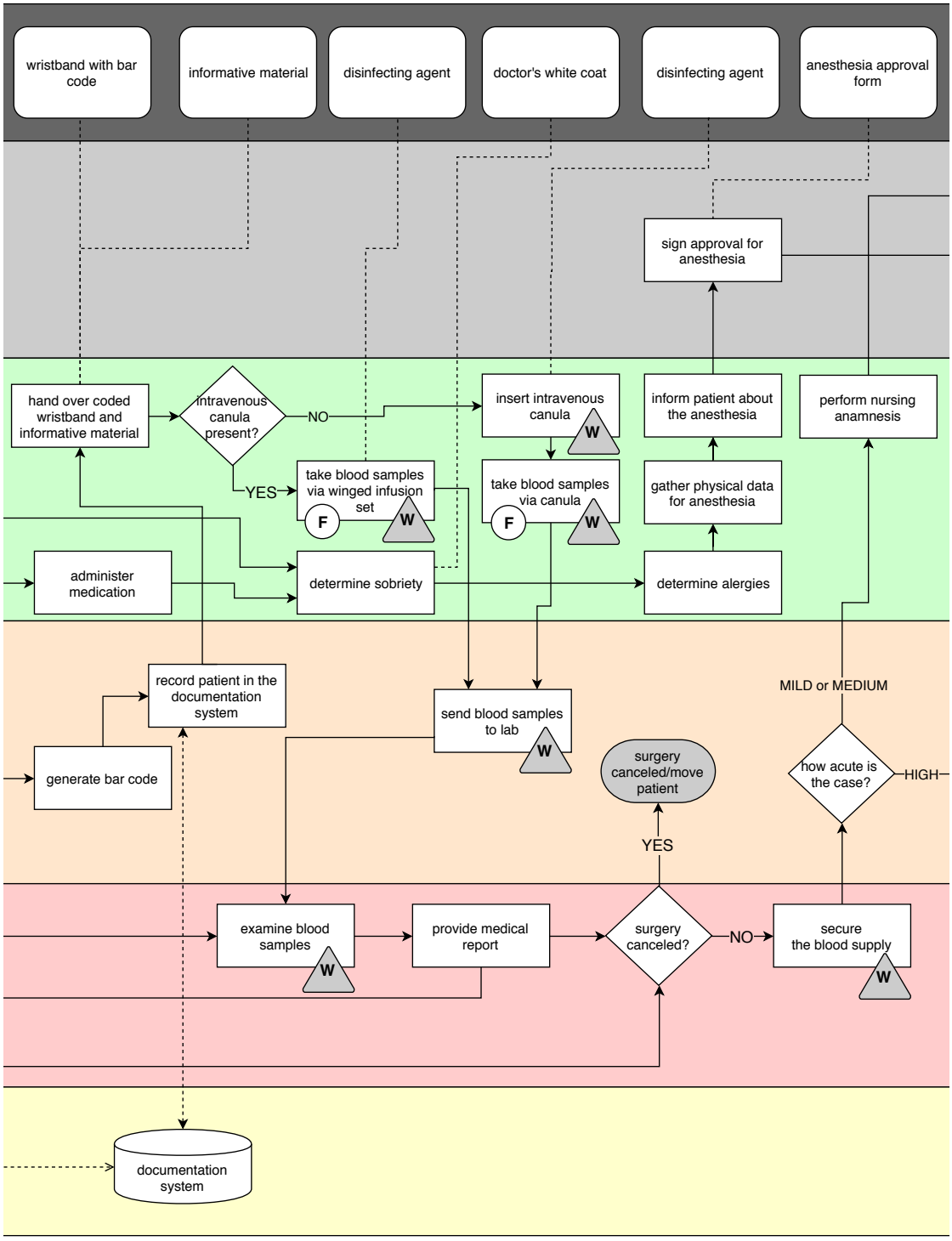

{surgical clothing}
change into surgical clothing
remove prostheses, jewelery and piercings
EE_3
transfer patient into surgical theatre
patient in surgery
Waiting Point
administer sedative medication
prepare surgical clothing for patient (e.g. thrombosis stockings)
MG_2
fill out surgical checklist

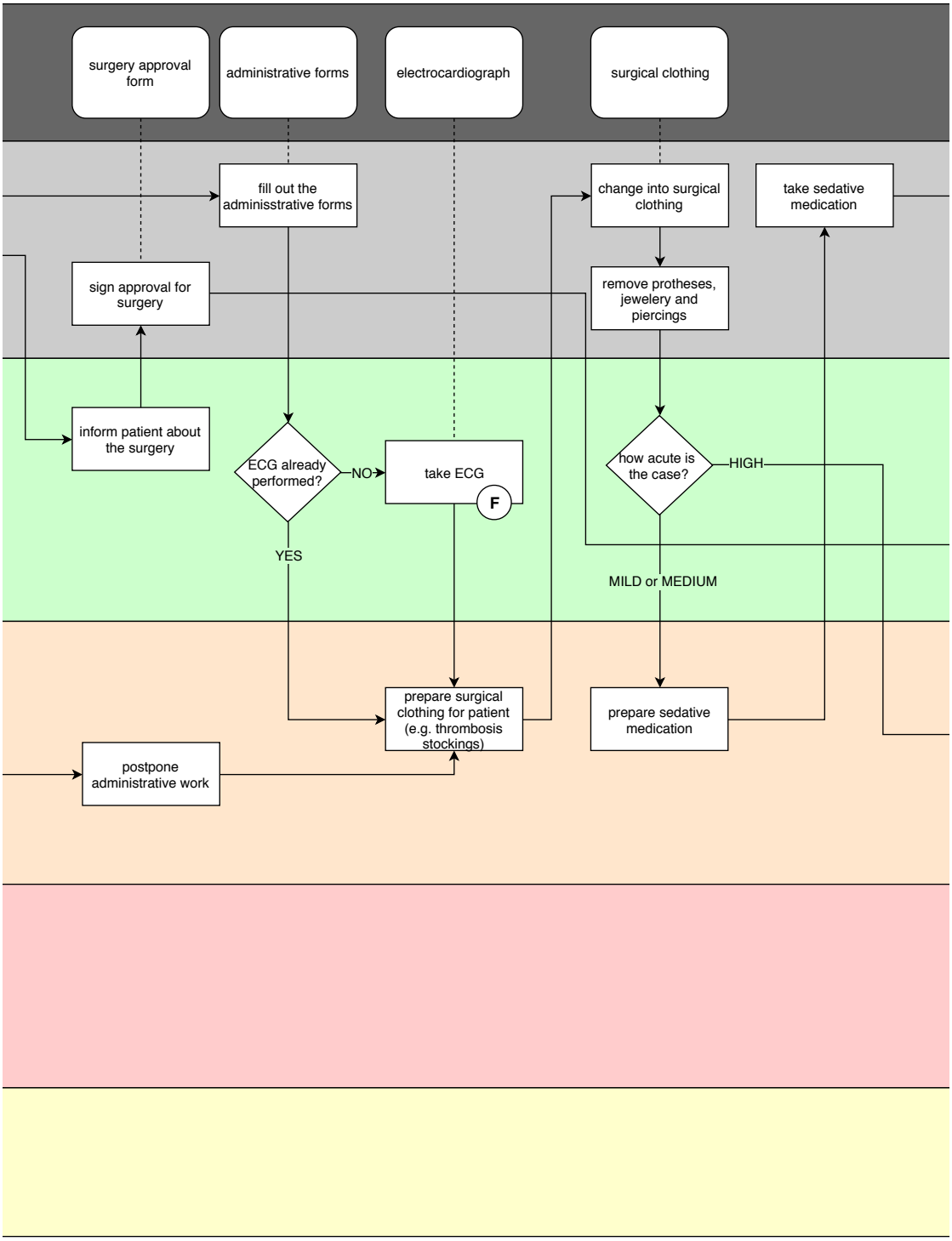
A.3 Detailed Service Blueprint

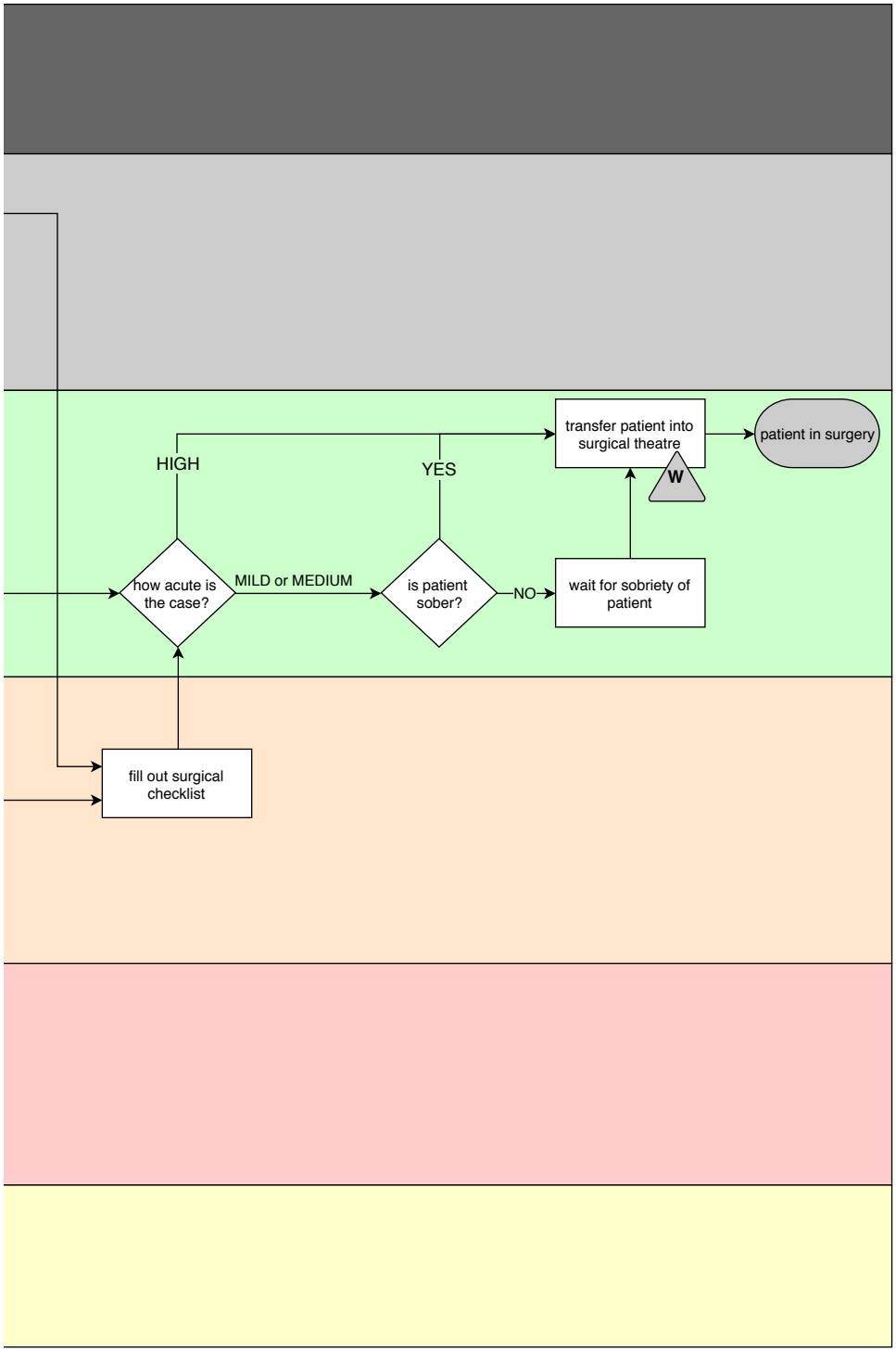



Physical Evidence
doctor's white coat
surgical ward
medical scrubs
hospital bed
hospital room
Patient Actions
enter clinic
Front-Stage Interactions
acute medical situation is detemined
is surgeon present?
NO
YES
transfer patient to surgical ward
W
welcome patient and inform about procedure
assign bed
monitor patient
NO
Back-Stage Interactions
hemodilution present?
surgeon is consulted
other surgeon necessary?
YES
inform performing surgeon
arrange compensational examinations
NO
YES
NO
antagonization possible?
YES
inform surgical ward
inform theatre
clinical report up to date?
NO
YES
Support Processes
surgery postponed/move patient to general ward
IT Elements
clinical report database


monitoring devices
disinfecting agent
express possible discomforts
determine alergies
immediate medical needs?
YES
attend to patients medical needs (e.g. pain)
NO
perform x-ray
W
intravenous canula present?
NO
insert intravenous canula
W
YES
administer medication
medication against hemodilution necessary?
NO
YES
patient already in system?
NO
enter e-card
complete patient data
F
W
YES
assemble results
W
assess record
proceed with surgery?
YES
sign internal approval
document the bed assignment in the system
NO
ERP System
ward organisation system


wristband with bar code
informative material
disinfecting agent
doctor's white coat
disinfecting agent
anesthesia approval form
sign approval for anesthesia
hand over coded wristband and informative material
intravenous canula present?
NO
YES
insert intravenous canula
W
inform patient about the anesthesia
perform nursing anamnesis
take blood samples via winged infusion set
F
W
take blood samples via canula
F
W
gather physical data for anesthesia
administer medication
determine sobriety
determine alergies
record patient in the documentation system
send blood samples to lab
W
MILD or MEDIUM
generate bar code
surgery canceled/move patient
how acute is the case?
HIGH
YES
examine blood samples
W
provide medical report
surgery canceled?
NO
secure the blood supply
W
documentation system


surgery approval form
administrative forms
electrocardiograph
surgical clothing
fill out the adminisstrative forms
change into surgical clothing
take sedative medication
sign approval for surgery
remove protheses, jewelery and piercings
inform patient about the surgery
ECG already performed?
NO
take ECG
F
how acute is the case?
HIGH
YES
MILD or MEDIUM
prepare surgical clothing for patient (e.g. thrombosis stockings)
prepare sedative medication
postpone administrative work


transfer patient into surgical theatre
patient in surgery
HIGH
YES
W
how acute is the case?
MILD or MEDIUM
is patient sober?
NO
wait for sobriety of patient
fill out surgical checklist

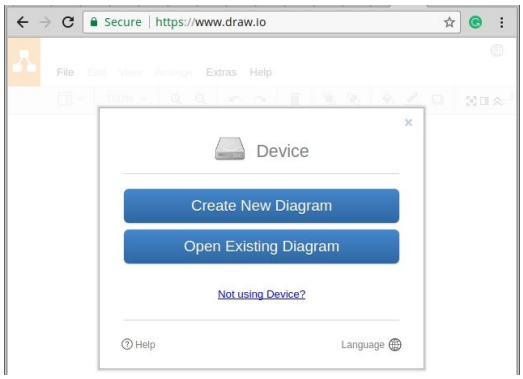
APPENDIX B

Tutorial: Service Blueprinting with *Draw.io*

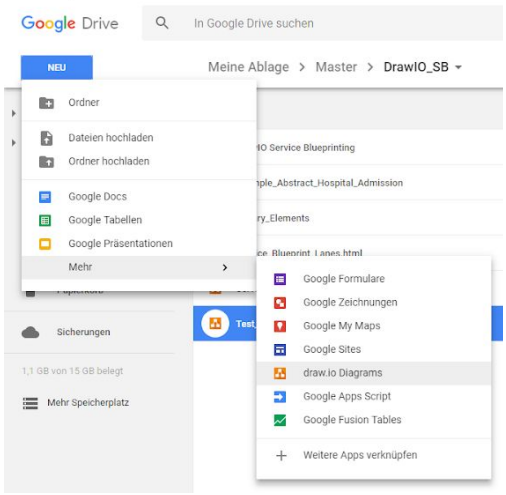
Service Blueprinting with draw.io

By Matthias Winkelhofer

Accessible via https://www.draw.io/



Or Google Drive



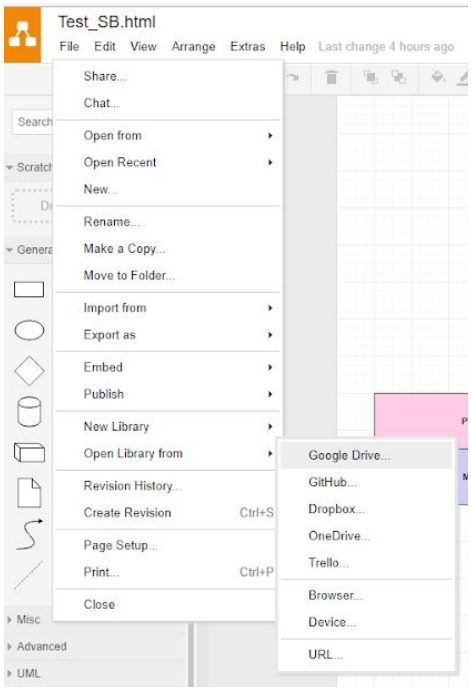
Service Blueprint draw.io Library

For the creation of Service Blueprints in draw.io the customized library *Service_Blueprinting.xml* can be imported and used.

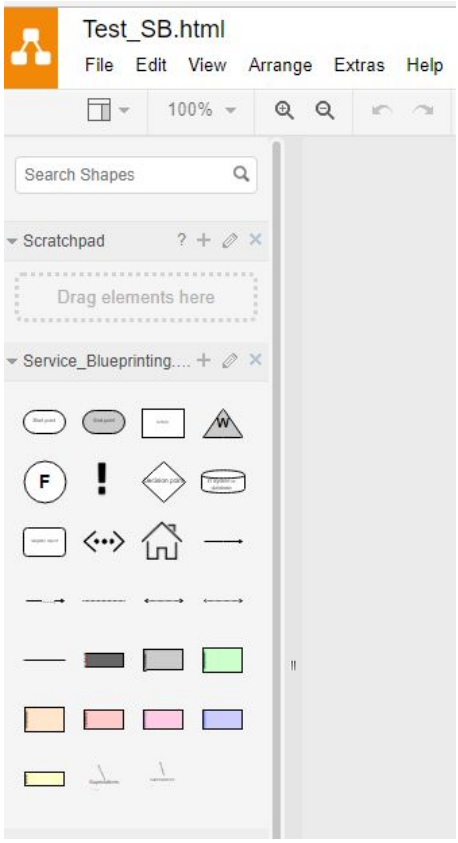
Note: The library was created for the development of a prototypical modeling environment and therefore is not perfect ;).

Import

1. Create new diagram
2. Click *File -> Open Library from* and select XML-File

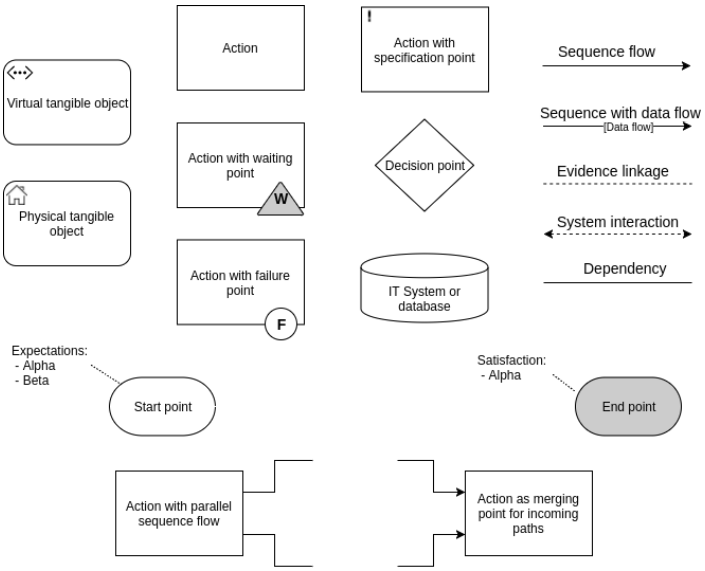


3. Afterwards the corresponding elements can be selected from the toolbar on the left side of the screen (drag&drop)

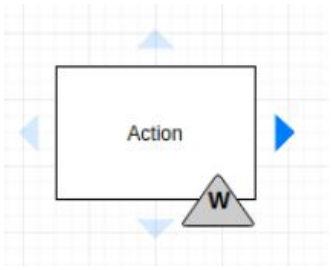


Selectable Elements

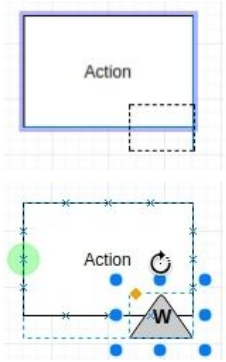
Note: For a better understanding of the applicability of the elements please have a look at the provided example diagram (*Example_Abstract_Hospital_Admission*).



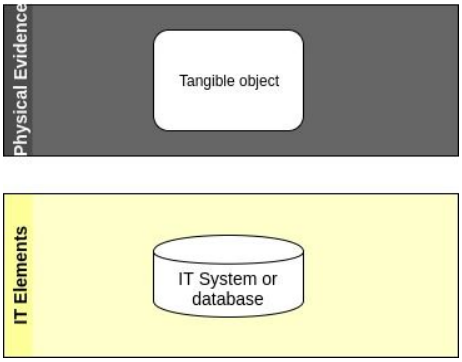
Note: In contrast to the usage of the static elements as shown above (i.e. actions, IT systems, tangible objects, start and end points), the corresponding edges do not have to be selected from the toolbar. It is ok to use the integrated linking option provided by draw.io as long as the specific directions (i.e. the arrows) are maintained, as shown above.



Note: Marker like the Waiting and Failure Point or the indication of tangible objects can be applied by dropping the corresponding icon on the underlying element (e.g. Action).
If placed correctly, a wrapping container (blue edging) will take care of the cohesion if the base element is moved on the canvas.



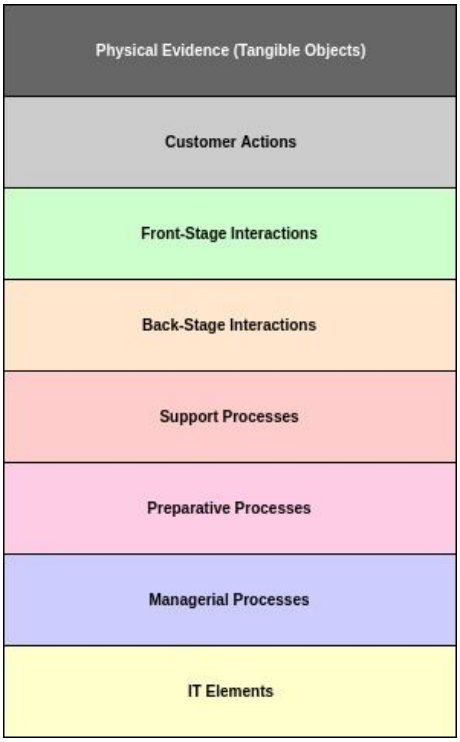
Note: In contrast to the other flow elements assigned to the main areas (i.e. customer actions, front-stage, back-stage and supportive actions), tangible objects and IT Systems may only be placed within the corresponding areas.



Note: Actions placed within the areas *Preparative and Managerial Processes* are not part of the direct value generation of the service and therefore not a part of the main sequence flow. They are only linked to depending actions to display additional interdependencies (see also *Example_Abstract_Hospital_Admission*).

Service Blueprint areas

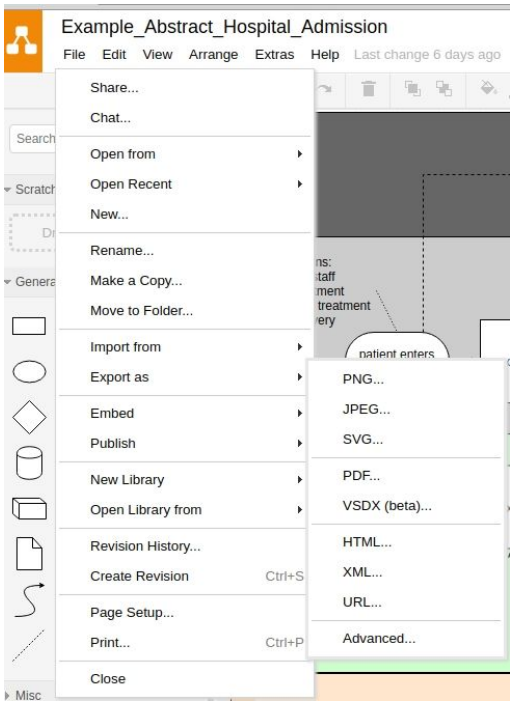
As it is custom for Service Blueprinting, the following order should be applied when it comes to the specific areas. Of course, if a certain area is not necessary for the models purpose it can be omitted.



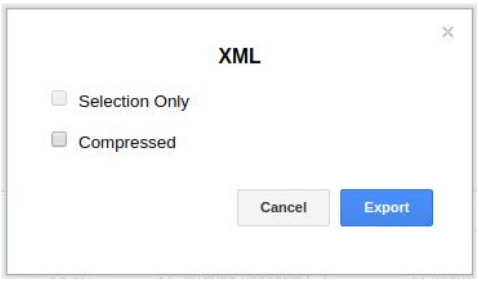
Extract a Service Blueprint XML from draw.io

In order to use a created Service Blueprint for further processing (i.e. transformation towards BPMN) a XML file has to be created.

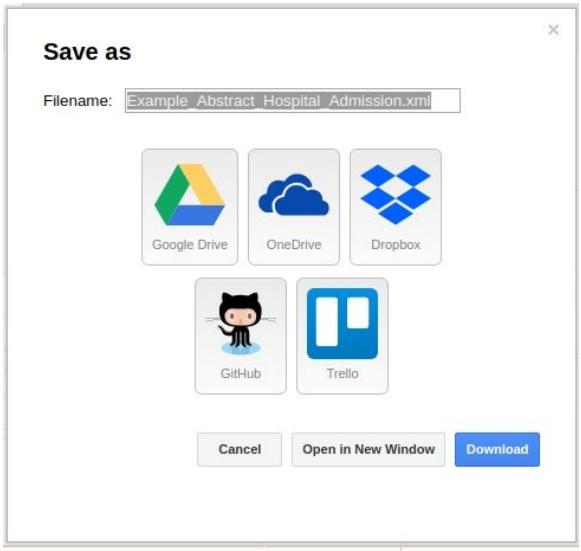
1. In draw.io click *File -> Export as -> XML...*



2. Deselect *compress* (otherwise the XML structure cannot be interpreted)

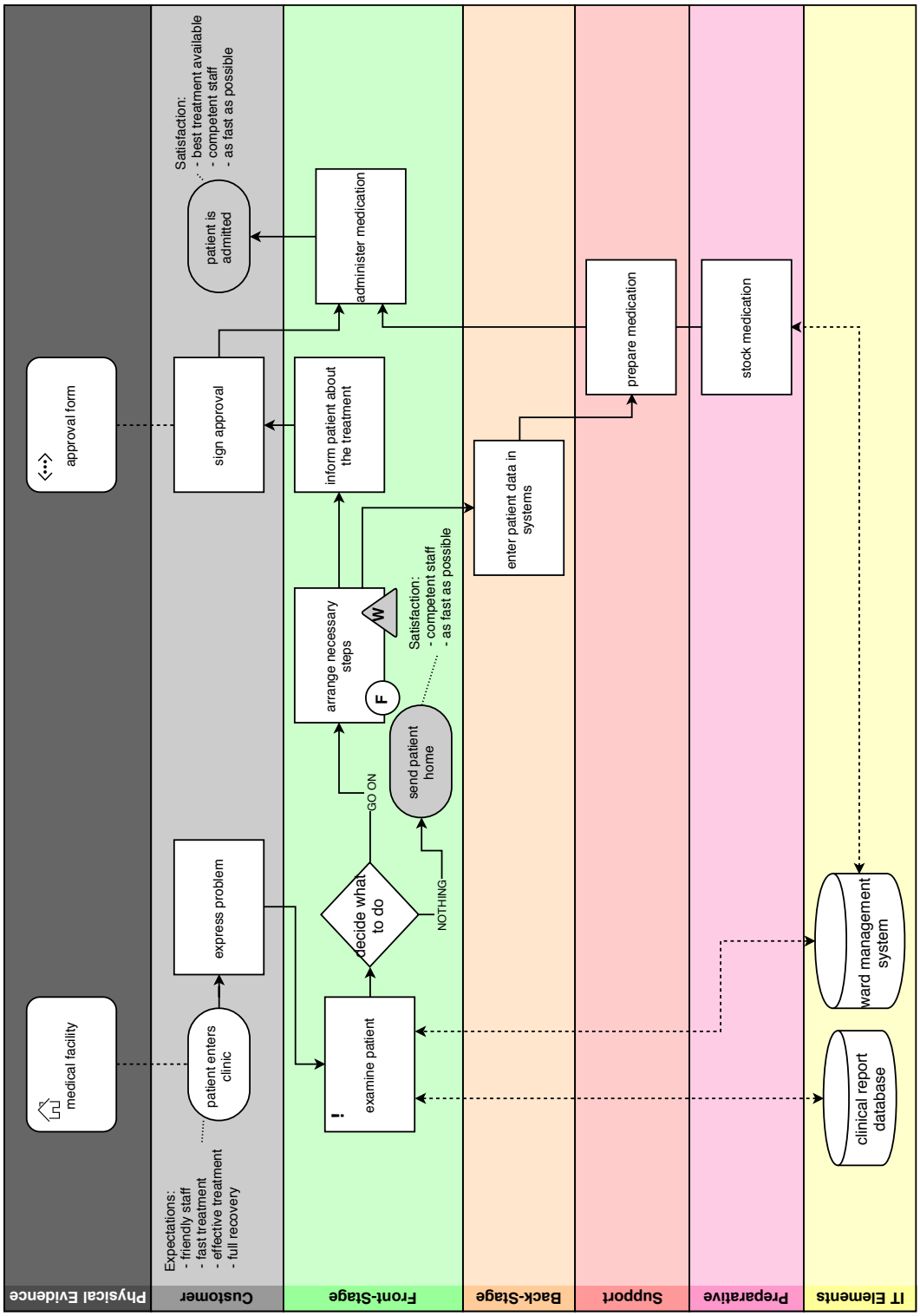


3. Choose the destination or simply click *Download*

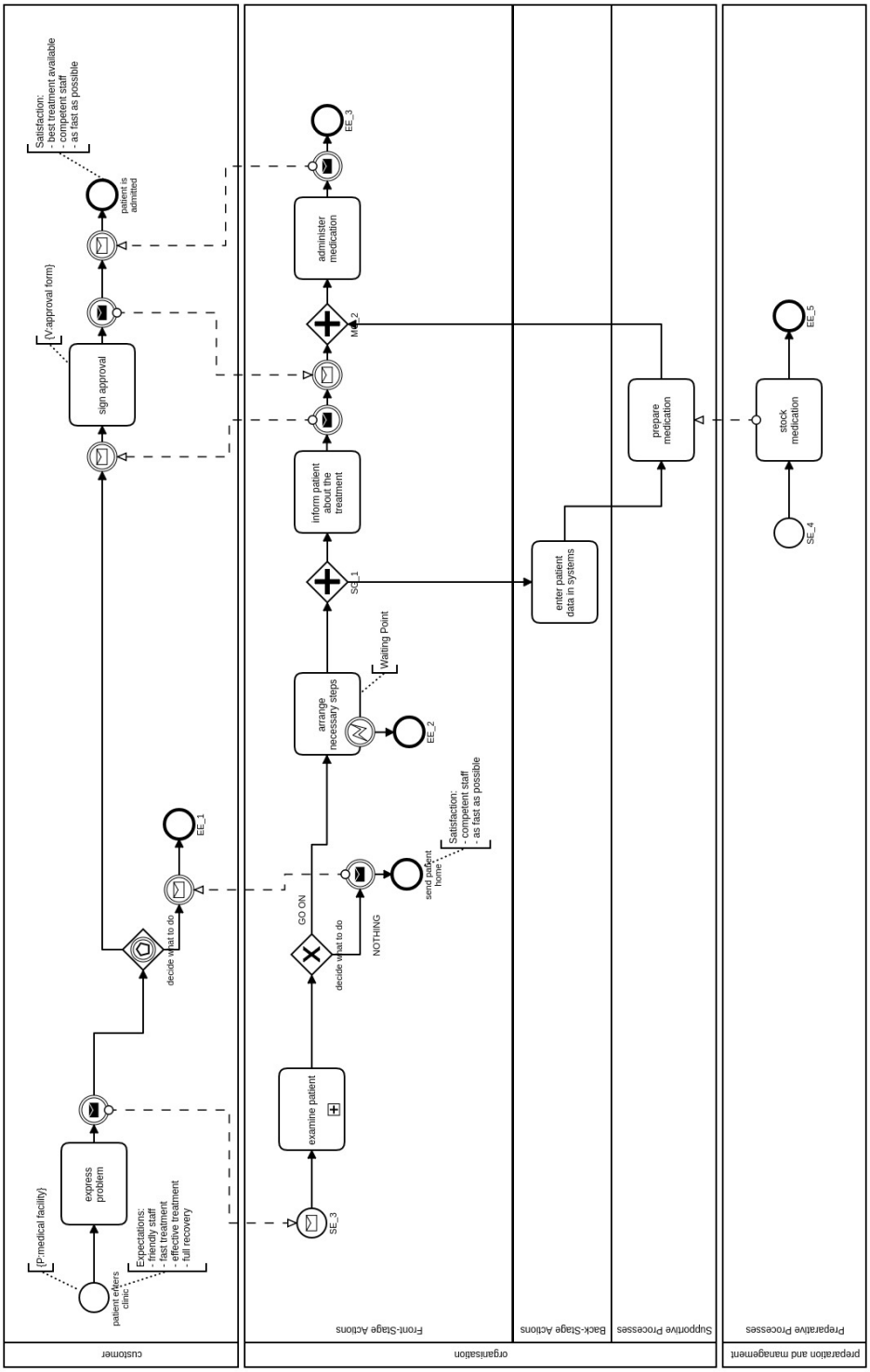


APPENDIX C

Example: Simplified Hospital Admission



Service Blueprint *Simplified Hospital Admission* created with *Draw.io*



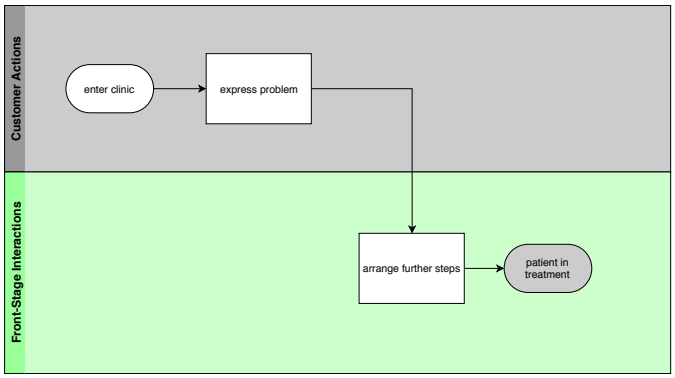
BPMN Model *Simplified Hospital Admission* (result of the transformation)[1]

[1]Please note that the illustration was slightly manually edited to fit the layout of this page

APPENDIX D

XML File Examples

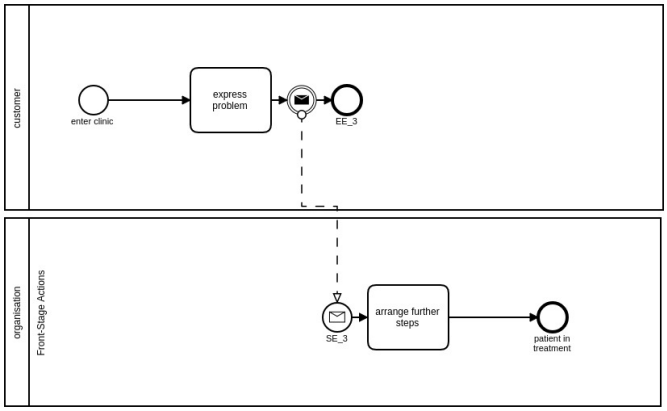
D.1 Service Blueprint created via the customized library in *Draw.io*




```
<?xml version="1.0" encoding="UTF-8"?>
<mxGraphModel dx="1426" dy="748" grid="1" gridSize="10" guides="1" tooltips="1"
connect="1" arrows="1" fold="1" page="1" pageScale="1" pageWidth="1169" pageHeight="826"
background="#ffffff" math="0" shadow="0">
   <root>
      <mxCell id="0" />
      <mxCell id="1" parent="0" />
      <mxCell id="7453095b2c57516d-1" value="Customer Actions"
      style="sbType=area_customer_actions;swimlane;html=1;horizontal=0;
      swimlaneFillColor=#CCCCCC;swimlaneLine=0;align=center;verticalAlign=middle;
      labelPosition=center;verticalLabelPosition=middle;glass=0;fillColor=#999999;
      strokeColor=#000000;rounded=0;comic=0;labelBackgroundColor=none;fontSize=14;
      fontColor=#000000;" parent="1" vertex="1">
         <mxGeometry x="120" y="190" width="760" height="190" as="geometry" />
      </mxCell>
      <mxCell id="60445726eb1ce408-5" style="edgeStyle=orthogonalEdgeStyle;rounded=0;
      html=1;entryX=0;entryY=0.5;jettySize=auto;orthogonalLoop=1;" edge="1"
      parent="7453095b2c57516d-1" source="4c8f399520ffc351-3"
      target="60445726eb1ce408-4">
         <mxGeometry relative="1" as="geometry" />
      </mxCell>
      <mxCell id="4c8f399520ffc351-3" value="enter clinic" style="sbType=start_point;
      shape=start_point;rounded=1;html=1;whiteSpace=wrap;align=center;
      gradientColor=none;glass=0;shadow=0;comic=0;labelBackgroundColor=none;
      strokeColor=#000000;strokeWidth=1;fontSize=12;fontColor=#000000;arcSize=50;"
      parent="7453095b2c57516d-1" vertex="1">
         <mxGeometry x="70" y="67.5" width="100" height="55" as="geometry" />
      </mxCell>
      <mxCell id="60445726eb1ce408-3" value="" style="sbType=action_container;
      shape=action_container;strokeColor=none;fillColor=none;rounded=0;comic=0;
      labelBackgroundColor=none;fontSize=14;fontColor=#000000;align=center;html=1;"
      vertex="1" connectable="0" parent="7453095b2c57516d-1">
         <mxGeometry x="230" y="55" width="120" height="80" as="geometry" />
      </mxCell>
      <mxCell id="60445726eb1ce408-4" value="express problem" style="sbType=action;
      shape=action;rounded=0;html=1;whiteSpace=wrap;align=center;
      labelBackgroundColor=none;strokeColor=#000000;strokeWidth=1;fillColor=#FFFFFF;
      gradientColor=none;fontSize=12;fontColor=#000000;comic=0;" vertex="1"
      parent="60445726eb1ce408-3">
         <mxGeometry width="120" height="80" as="geometry" />
      </mxCell>
      <mxCell id="7453095b2c57516d-2" value="Front-Stage Interactions"
      style="sbType=area_front_stage_interactions;swimlane;html=1;horizontal=0;
      swimlaneFillColor=#CCFFCC;swimlaneLine=0;align=center;verticalAlign=middle;
      labelPosition=center;verticalLabelPosition=middle;glass=0;fillColor=#99FF99;
      rounded=0;comic=0;labelBackgroundColor=none;strokeColor=#000000;strokeWidth=1;
      fontSize=14;fontColor=#000000;" parent="1" vertex="1">
         <mxGeometry x="120" y="380" width="760" height="230" as="geometry" />
```

```
        </mxCell>
        <mxCell id="7453095b2c57516d-7" value="" style="sbType=action_container;
        shape=action_container;strokeColor=none; fillColor=none;rounded=0;comic=0;
        labelBackgroundColor=none;fontSize=14;fontColor=#000000;align=center;html=1;"
        parent="7453095b2c57516d-2" vertex="1" connectable="0">
            <mxGeometry x="404.5" y="70" width="120" height="80" as="geometry" />
        </mxCell>
        <mxCell id="7453095b2c57516d-8" value="arrange further steps"
        style="sbType=action;shape=action;rounded=0;html=1;whiteSpace=wrap;align=center;
        labelBackgroundColor=none;strokeColor=#000000;strokeWidth=1;fillColor=#FFFFFF;
        gradientColor=none;fontSize=12;fontColor=#000000;comic=0;"
        parent="7453095b2c57516d-7" vertex="1">
            <mxGeometry width="120" height="80" as="geometry" />
        </mxCell>
        <mxCell id="7453095b2c57516d-14" style="edgeStyle=orthogonalEdgeStyle;rounded=0;
        html=1;entryX=0;entryY=0.5;labelBackgroundColor=none;startArrow=none;startFill=0;
        endArrow=classic;endFill=1;jettySize=auto;orthogonalLoop=1;strokeWidth=1;
        fontSize=25;fontColor=#000000;" parent="7453095b2c57516d-2"
        source="7453095b2c57516d-8" target="4c8f399520ffc351-2" edge="1">
            <mxGeometry relative="1" as="geometry">
                <mxPoint x="560" y="110" as="targetPoint" />
            </mxGeometry>
        </mxCell>
        <mxCell id="4c8f399520ffc351-2" value="patient in treatment"
        style="sbType=end_point;shape=end_point;rounded=1;html=1;whiteSpace=wrap;
        align=center;gradientColor=none;glass=0;shadow=0;comic=0;
        labelBackgroundColor=none;strokeColor=#000000;strokeWidth=1;fontSize=12;
        fontColor=#000000;arcSize=50;fillColor=#CCCCCC;" parent="7453095b2c57516d-2"
        vertex="1">
            <mxGeometry x="570" y="82.5" width="100" height="55" as="geometry" />
        </mxCell>
        <mxCell id="60445726eb1ce408-6" style="edgeStyle=orthogonalEdgeStyle;
        rounded=0;html=1;entryX=0.5;entryY=0;jettySize=auto;orthogonalLoop=1;"
        edge="1" parent="1" source="60445726eb1ce408-4" target="7453095b2c57516d-8">
            <mxGeometry relative="1" as="geometry" />
        </mxCell>
    </root>
</mxGraphModel>
```

D.2 Simple Process Model as *.bpmn*




```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI"
xmlns:dc="http://www.omg.org/spec/DD/20100524/DC"
xmlns:di="http://www.omg.org/spec/DD/20100524/DI"
xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="definitions_dd80ea6a-14f6-47db-89ef-104be4577c7e"
targetNamespace="http://bpmn.io/schema/bpmn" exporter="Camunda Modeler"
exporterVersion="1.11.3">
  <collaboration id="collaboration_dcbf1c22-0a9a-4d47-a000-2a63ee0fb29f">
    <participant id="customer" name="customer" processRef="customer-process" />
    <participant id="organisation" name="organisation"
    processRef="organisation-process" />
    <messageFlow id="intermediateThrowEvent_e436c246-0a90-4488-8ecd-265a9903dfd1-
    startEvent_648ec9e7-5beb-4417-af70-5f1b51a016d2"
    sourceRef="intermediateThrowEvent_e436c246-0a90-4488-8ecd-265a9903dfd1"
    targetRef="startEvent_648ec9e7-5beb-4417-af70-5f1b51a016d2" />
  </collaboration>
  <process id="customer-process" isExecutable="false">
    <task id="sb-60445726eb1ce408-4" name="express problem">
      <incoming>sb-4c8f399520ffc351-3-sb-60445726eb1ce408-4</incoming>
      <outgoing>sb-60445726eb1ce408-4-
      intermediateThrowEvent_e436c246-0a90-4488-8ecd-265a9903dfd1</outgoing>
    </task>
    <sequenceFlow id="sb-4c8f399520ffc351-3-sb-60445726eb1ce408-4"
    sourceRef="sb-4c8f399520ffc351-3" targetRef="sb-60445726eb1ce408-4" />
    <intermediateThrowEvent
    id="intermediateThrowEvent_e436c246-0a90-4488-8ecd-265a9903dfd1">
      <incoming>sb-60445726eb1ce408-4-
      intermediateThrowEvent_e436c246-0a90-4488-8ecd-265a9903dfd1</incoming>
      <outgoing>intermediateThrowEvent_e436c246-0a90-4488-8ecd-265a9903dfd1-
      endEvent_44420c43-0d70-4bab-9df5-464ea0320bbb</outgoing>
      <messageEventDefinition
      id="messageEventDefinition_95832a63-0c08-4685-86cd-a720da747a3d" />
    </intermediateThrowEvent>
    <endEvent id="endEvent_44420c43-0d70-4bab-9df5-464ea0320bbb" name="EE_3">
      <incoming>intermediateThrowEvent_e436c246-0a90-4488-8ecd-265a9903dfd1-
      endEvent_44420c43-0d70-4bab-9df5-464ea0320bbb</incoming>
    </endEvent>
    <sequenceFlow id="intermediateThrowEvent_e436c246-0a90-4488-8ecd-265a9903dfd1-
    endEvent_44420c43-0d70-4bab-9df5-464ea0320bbb"
    sourceRef="intermediateThrowEvent_e436c246-0a90-4488-8ecd-265a9903dfd1"
    targetRef="endEvent_44420c43-0d70-4bab-9df5-464ea0320bbb" />
    <sequenceFlow id="sb-60445726eb1ce408-4-
    intermediateThrowEvent_e436c246-0a90-4488-8ecd-265a9903dfd1"
    sourceRef="sb-60445726eb1ce408-4"
    targetRef="intermediateThrowEvent_e436c246-0a90-4488-8ecd-265a9903dfd1" />
    <startEvent id="sb-4c8f399520ffc351-3" name="enter clinic">
```

```
      <outgoing>sb-4c8f399520ffc351-3-sb-60445726eb1ce408-4</outgoing>
    </startEvent>
  </process>
  <process id="organisation-process" isExecutable="false">
    <laneSet id="organisation-laneset">
      <lane id="sb-7453095b2c57516d-2" name="Front-Stage Actions">
        <flowNodeRef>sb-7453095b2c57516d-8</flowNodeRef>
        <flowNodeRef>startEvent_648ec9e7-5beb-4417-af70-5f1b51a016d2</flowNodeRef>
        <flowNodeRef>sb-4c8f399520ffc351-2</flowNodeRef>
      </lane>
    </laneSet>
    <task id="sb-7453095b2c57516d-8" name="arrange further steps">
      <incoming>startEvent_648ec9e7-5beb-4417-af70-5f1b51a016d2-
      sb-7453095b2c57516d-8</incoming>
      <outgoing>sb-7453095b2c57516d-8-sb-4c8f399520ffc351-2</outgoing>
    </task>
    <startEvent id="startEvent_648ec9e7-5beb-4417-af70-5f1b51a016d2" name="SE_3">
      <outgoing>startEvent_648ec9e7-5beb-4417-af70-5f1b51a016d2-
      sb-7453095b2c57516d-8</outgoing>
      <messageEventDefinition
      id="messageEventDefinition_66e929b0-e72c-4818-8991-7cffbe47ccd6" />
    </startEvent>
    <sequenceFlow id="startEvent_648ec9e7-5beb-4417-af70-5f1b51a016d2-
    sb-7453095b2c57516d-8" sourceRef="startEvent_648ec9e7-5beb-4417-af70-5f1b51a016d2"
    targetRef="sb-7453095b2c57516d-8" />
    <sequenceFlow id="sb-7453095b2c57516d-8-sb-4c8f399520ffc351-2"
    sourceRef="sb-7453095b2c57516d-8" targetRef="sb-4c8f399520ffc351-2" />
    <endEvent id="sb-4c8f399520ffc351-2" name="patient in treatment">
      <incoming>sb-7453095b2c57516d-8-sb-4c8f399520ffc351-2</incoming>
    </endEvent>
  </process>
  <bpmndi:BPMNDiagram id="diagram">
    <bpmndi:BPMNPlane id="processplane"
    bpmnElement="collaboration_dcbf1c22-0a9a-4d47-a000-2a63ee0fb29f">
      <bpmndi:BPMNShape id="BPMNShape_dfbfede3-396a-4805-b624-2376505a7e70"
      bpmnElement="customer">
        <dc:Bounds x="120" y="190" width="880" height="290" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="BPMNShape_faad355f-deef-451b-b0fa-83ce617ddc06"
      bpmnElement="organisation">
        <dc:Bounds x="120" y="490" width="870" height="330" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="BPMNShape_90a96ab5-dedb-40b5-bf39-0ebded807ce0"
      bpmnElement="sb-7453095b2c57516d-2">
        <dc:Bounds x="150" y="490" width="850" height="330" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="BPMNShape_6a4782a0-f61b-4997-bba2-d2ecef9686b0"
      bpmnElement="sb-4c8f399520ffc351-3">
        <dc:Bounds x="212" y="300" width="36" height="36" />
        <bpmndi:BPMNLabel>
          <dc:Bounds x="202" y="336" width="56" height="12" />
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="BPMNShape_fde745f3-13bd-4919-89dc-db05e36d44c4"
      bpmnElement="sb-60445726eb1ce408-4">
        <dc:Bounds x="350" y="278" width="100" height="80" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNEdge id="BPMNEdge_75c6a367-02c0-4d89-b0f3-47c3277a5226"
      bpmnElement="sb-4c8f399520ffc351-3-sb-60445726eb1ce408-4"
      sourceElement="BPMNShape_6a4782a0-f61b-4997-bba2-d2ecef9686b0"
      targetElement="BPMNShape_fde745f3-13bd-4919-89dc-db05e36d44c4">
        <di:waypoint xsi:type="dc:Point" x="248" y="318" />
        <di:waypoint xsi:type="dc:Point" x="299" y="318" />
        <di:waypoint xsi:type="dc:Point" x="299" y="318" />
        <di:waypoint xsi:type="dc:Point" x="350" y="318" />
        <bpmndi:BPMNLabel>
          <dc:Bounds x="269" y="308" width="90" height="20" />
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNShape id="BPMNShape_cf6bee69-00b2-475b-8489-58094de63481"
      bpmnElement="sb-7453095b2c57516d-8">
        <dc:Bounds x="569.5" y="593" width="100" height="80" />
      </bpmndi:BPMNShape>
```

```
      <bpmndi:BPMNShape id="BPMNShape_32605f6b-c7df-4c38-b75f-45cd028b6aa6"
      bpmnElement="intermediateThrowEvent_e436c246-0a90-4488-8ecd-265a9903dfd1">
        <dc:Bounds x="470" y="300" width="36" height="36" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="BPMNShape_8467b476-5f31-4ad3-9e78-0c670a02f0ae"
      bpmnElement="endEvent_44420c43-0d70-4bab-9df5-464ea0320bbb">
        <dc:Bounds x="526" y="300" width="36" height="36" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNEdge id="BPMNEdge_257aaee3-362c-433e-b929-4d7078aeabcc" bpmnElement="intermediateThro
      endEvent_44420c43-0d70-4bab-9df5-464ea0320bbb"
      sourceElement="BPMNShape_32605f6b-c7df-4c38-b75f-45cd028b6aa6"
      targetElement="BPMNShape_8467b476-5f31-4ad3-9e78-0c670a02f0ae">
        <di:waypoint xsi:type="dc:Point" x="506" y="318" />
        <di:waypoint xsi:type="dc:Point" x="516" y="318" />
        <di:waypoint xsi:type="dc:Point" x="516" y="318" />
        <di:waypoint xsi:type="dc:Point" x="526" y="318" />
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNShape id="BPMNShape_57ca7ece-b087-4f12-92ab-7d7584564c7c" bpmnElement="startEvent_648
        <dc:Bounds x="513.5" y="615" width="36" height="36" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNEdge id="BPMNEdge_b8d222b5-407d-4795-83fd-e4539c74724c"
      bpmnElement="sb-60445726eb1ce408-4-
      intermediateThrowEvent_e436c246-0a90-4488-8ecd-265a9903dfd1"
      sourceElement="BPMNShape_fde745f3-13bd-4919-89dc-db05e36d44c4"
      targetElement="BPMNShape_32605f6b-c7df-4c38-b75f-45cd028b6aa6">
        <di:waypoint xsi:type="dc:Point" x="450" y="318" />
        <di:waypoint xsi:type="dc:Point" x="460" y="318" />
        <di:waypoint xsi:type="dc:Point" x="460" y="318" />
        <di:waypoint xsi:type="dc:Point" x="470" y="318" />
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="BPMNEdge_edfb5f46-7ccf-4603-8417-7557763eb21d"
      bpmnElement="intermediateThrowEvent_e436c246-0a90-4488-8ecd-265a9903dfd1-
      startEvent_648ec9e7-5beb-4417-af70-5f1b51a016d2"
      sourceElement="BPMNShape_32605f6b-c7df-4c38-b75f-45cd028b6aa6"
      targetElement="BPMNShape_57ca7ece-b087-4f12-92ab-7d7584564c7c">
        <di:waypoint xsi:type="dc:Point" x="488" y="336" />
        <di:waypoint xsi:type="dc:Point" x="488" y="475.5" />
        <di:waypoint xsi:type="dc:Point" x="531.5" y="475.5" />
        <di:waypoint xsi:type="dc:Point" x="531.5" y="615" />
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="BPMNEdge_c34d3dca-fb5f-4da5-8b10-ed0ea297b8f4"
      bpmnElement="startEvent_648ec9e7-5beb-4417-af70-5f1b51a016d2-
      sb-7453095b2c57516d-8"
      sourceElement="BPMNShape_57ca7ece-b087-4f12-92ab-7d7584564c7c"
      targetElement="BPMNShape_cf6bee69-00b2-475b-8489-58094de63481">
        <di:waypoint xsi:type="dc:Point" x="549.5" y="633" />
        <di:waypoint xsi:type="dc:Point" x="559.5" y="633" />
        <di:waypoint xsi:type="dc:Point" x="559.5" y="633" />
        <di:waypoint xsi:type="dc:Point" x="569.5" y="633" />
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNShape id="BPMNShape_01c2cc35-754f-48e2-9c1d-c8f9f0c7b6ab"
      bpmnElement="sb-4c8f399520ffc351-2">
        <dc:Bounds x="780" y="615" width="36" height="36" />
        <bpmndi:BPMNLabel>
          <dc:Bounds x="774" y="651" width="49" height="24" />
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNEdge id="BPMNEdge_b3930e09-9e52-46be-978b-8f9930c22dcb"
      bpmnElement="sb-7453095b2c57516d-8-sb-4c8f399520ffc351-2"
      sourceElement="BPMNShape_cf6bee69-00b2-475b-8489-58094de63481"
      targetElement="BPMNShape_01c2cc35-754f-48e2-9c1d-c8f9f0c7b6ab">
        <di:waypoint xsi:type="dc:Point" x="670" y="633" />
        <di:waypoint xsi:type="dc:Point" x="724.75" y="633" />
        <di:waypoint xsi:type="dc:Point" x="724.75" y="633" />
        <di:waypoint xsi:type="dc:Point" x="780" y="633" />
        <bpmndi:BPMNLabel>
          <dc:Bounds x="694.75" y="623" width="90" height="20" />
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNEdge>
    </bpmndi:BPMNPlane>
  </bpmndi:BPMNDiagram>
</definitions>
```

List of Figures

Bibliography

[AF15] Sabah Al-Fedaghi. Alternative approach to service blueprinting. In M. Surendra Prasad Babu and Li Wenzheng, editors, *Proceedings of 2015 IEEE 6th International Conference on Software Engineering and Service Science*, pages 54–61. Piscataway, September 2015.

[Ama17] Amazon. The Amazon Marketplace. `https://services.amazon.de/programme/online-verkaufen/so-funktionierts-pro.html?ld=SEATSOAADGog-Branded_amazon-marketplace_p_213086498592_c`, 2017. Accessed: 2018-05-06.

[And12] Andreas Naef. Übersicht BPMN Prozessmodellierungs-Tools. `http://prozessmanagement-blog.ch/post/32587959467/uebersicht-bpmn-prozessmodellierungs-tools`, September 2012. Accessed: 2018-05-06.

[Art00] William B. Arthur. Cognition: The Black Box of Economics. In David Colander, editor, *The Complexity Vision and the Teaching of Economics*, pages 51–57. Edward Elgar Publishing, 2000.

[Art16] Daniel Arthursson. How Millennials Are Defining the Sharing Economy. `https://www.entrepreneur.com/article/275802`, June 2016. Accessed: 2018-05-06.

[Atl18] Atlassian. Bitbucket. `https://de.atlassian.com/software/bitbucket/`, 2018. Accessed: 2018-04-09.

[Bar11] Eric Barker. Does creativity require freedom or constraints? `http://www.businessinsider.com/does-creativity-require-freedom-or-constraints-2011-8?IR=T`, August 2011. Accessed: 2018-05-06.

[Bau10] Ramona Baureis. Basic rules of EPC modelling. `http://www.ariscommunity.com/users/rbaureis/2010-03-22-basic-rules-epc-modelling/`, March 2010. Accessed: 2018-05-06.

[BCOR11] Ana P. B. Barquet, Vitor P. Cunha, Maicon G. Oliveira, and Henrique Rozenfeld. Business model elements for product-service system. In Jürgen Hesselbach and Christoph Herrmann, editors, *Proceedings of the 3rd CIRP International Conference on Industrial Product Service Systems*, pages 332–337. Springer-Verlag, May 2011.

[BCW12] Marco Brambilla, Jordi Cabot, and Manuel Wimmer. *Model-Driven Software Engineering in Practice*. Morgan & Claypool Publishers, 1. edition, 2012.

[Ber15] BPM Offensive Berlin. BPMNPoster. `http://www.bpmb.de/index.php/BPMNPoster`, 2015. Accessed: 2018-05-06.

[Bha17] Hitesh Bhasin. Intangibility of services. `http://www.marketing91.com/intangibility-in-services/`, December 2017. Accessed: 2018-05-06.

[BK] Matthew Barsalou and Heinz G. Kehl. Grundlagen der Fehlermöglichkeits- und Einfluss-Analyse. `https://www.qz-online.de/qualitaets-management/qm-basics/methoden/fmea/artikel/grundlagen-der-fehlermoeglichkeits-und-einfluss-analyse-903982.html/`. Accessed: 2018-05-06.

[BOM08] Mary Jo Bitner, Amy L. Ostrom, and Felicia N. Morgan. Service Blueprinting: A Practical Technique for Service Innovation. *California Management Review*, 50(3):66–94, 2008.

[BPM12] BPMN Forum. What is the Business Process Modeling Notation (BPMN)? `http://bpmnforum.com/bpmn-faq/#What_is_BPMN`, 2012. Accessed: 2018-05-06.

[Bus18] BusinessDictionary. Services Definition. `http://www.businessdictionary.com/definition/services.html`, 2018. Accessed: 2018-05-06.

[BY05] Nabil Boughnim and Bernard Yannou. Using Blueprinting Method for Developing Product Service Systems. In Andrew Samuel and William Lewis, editors, *Proceedings of the 5th International Conference on Engineering Design*. Design Society, August 2005.

[Cam17] Camunda. Cawemo Features. `https://cawemo.com/`, 2017. Accessed: 2017-10-26.

[Can] Canvanizer. Service BLueprint Canvas. `https://canvanizer.com/new/service-blueprint-canvas`. Accessed: 2018-05-06.

[Chu07] Pao-Tiao Chuang. Combining Service Blueprint and FMEA for Service Design. *The Service Industries Journal*, 27(2):91–104, 2007.

[Com17] Aris Community. Event-driven process chain (EPC). http://www.ariscommunity.com/event-driven-process-chain/, 2017. Accessed: 2018-05-06.

[Dij16] Geke Van Dijk. Design Ethnography: Taking Inspiration from Everyday Life. In Marc Stickdorn and Jakob Schneider, editors, *This Is Service Design Thinking*, pages 108–115. BIS Publishing, 2016.

[Dog17] Artyom Dogtiev. Uber Revenue and Usage Statistics 2017. http://www.businessofapps.com/data/uber-statistics/#3, September 2017. Accessed: 2018-05-06.

[Dra] Draw.io. Tool for online modeling. https://www.draw.io/. Accessed: 2018-05-06.

[DRK16] Geke Van Dijk, Bas Raijmakers, and Luke Kelly. What are the tools of Service Design? In Marc Stickdorn and Jakob Schneider, editors, *This Is Service Design Thinking*, pages 147–215. BIS Publishing, 2016.

[FK04] Sabine Fließ and Michael Kleinaltenkamp. Blueprinting the service company: Managing service processes efficiently. *Journal of Business Research*, 57(4):392–404, 2004.

[FLM04] Sabine Fließ, Britta Lasshof, and Monika Meckel. Möglichkeiten der Integration eines Zeitmanagements in das Blueprinting von Dienstleistungsprozessen. Diskussionspapier 362, FernUniversität in Hagen / Fachbereich Wirtschaftswissenschaft, June 2004.

[Flo15] Flokzu's Team. What is BPM in plain English? Explained to Small and Medium Enterprises. http://www.flokzu.com/blog/en/smes/what-is-bpm/, April 2015. Accessed: 2017-10-18.

[Fou17] Eclipse Foundation. Eclipse BPMN Modeler. https://www.eclipse.org/bpmn2-modeler/, 2017. Accessed: 2018-04-09.

[Fou18] The Apache Software Foundation. Apache Maven. https://maven.apache.org/, 2018. Accessed: 2018-04-09.

[Gre13] Gregor Polančič. Common BPMN Modeling Mistakes and Best-Practices: Basic Events. http://blog.goodelearning.com/bpmn/common-bpmn-modeling-mistakes-best-practices-basic-events/, February 2013. Accessed: 2018-05-06.

[Gui12] Management Study Guide. Services Marketing - Definition and its Importance. http://managementstudyguide.com/services-marketing.htm, 2012. Accessed: 2018-05-06.

[Hav05] Michael Havey. *Essential business process modeling.* O'Reilly, 1. edition, 2005.

[HPN08] Janelle B. Hill, Massimo Pezzini, and Yefim V. Natis. Findings: confusion remains regarding BPM terminologies. *Gartner Research*, 501(G00155817), 2008.

[JGr18] JGraph. mxGraph 3.9.3. `https://jgraph.github.io/mxgraph/`, March 2018. Accessed: 2018-05-06.

[Jua16] Juan J. Moreno. What are the strengths BPMN for a method for modeling process and why is better than others methods? What are the weakness of BPMN? `https://www.quora.com/What-are-the-strengths-BPMN-for-a-method-for-modeling-process-and-why-is-better-than-others-methods-What-are-the-weakness-of-BPMN`, January 2016. Accessed: 2018-05-06.

[KB89] Jane Kingman-Brundage. The ABCs of service system blueprinting. In Mary Jo Bitner, editor, *Designing a winning service strategy*, pages 30–33. Amer Marketing Assn, 1989.

[KB91] Jane Kingman-Brundage. Technology, Design and Service Quality. *International Journal of Service Industry Management*, 2(3):47–59, December 1991.

[KB93] Jane Kingman-Brundage. Service mapping: gaining a concrete perspective on service system design. In Eberhard E. Scheuing and William F. Christopher, editors, *The service quality handbook*, page 148–163. Amacom, 1993.

[KB95] Jane Kingman-Brundage. Service mapping: back to basics. In William J. Glynn and James G. Barnes, editors, *Understanding Services Management: Integrating Marketing, Organizational Behaviour, Operations and Human Resource Management*, page 119–142. John Wiley and Sons, 1995.

[Kim16] Lucy Kimbell. Narketing: Connecting with People, Creating Value. In Marc Stickdorn and Jakob Schneider, editors, *This Is Service Design Thinking*, pages 46–51. BIS Publishing, 2016.

[KLL09] Ryan K.L. Ko, Stephen S.G. Lee, and Eng W. Lee. Business process management (BPM) standards: a survey. *Business Process Management Journal*, 15(5):744–791, 2009.

[KMJ14] Yahya Kazemzadeh, Simon K. Milton, and Lester W. Johnson. An explication of three service business process modeling approaches. In Abu N. M. Wahid and Carmen R. Amaro, editors, *Proceedings: Australian Academy of Business and Social Sciences Conference 2014*, pages 40–53. Australian Academy of Business and Social Science, August 2014.

[KMJ15] Yahya Kazemzadeh, Simon K. Milton, and Lester W. Johnson. Service Blueprinting and Business Process Modeling Notation (BPMN): A Conceptual Comparison. *Asian Social Science*, 11(12):307–319, 2015.

[Lov14] Christopher Lovelock. *Services Marketing: An Asia-Pacific Perspective*. Pearson Australia, 6. edition, 2014.

[LW11] Christopher Lovelock and Jochen Wirtz. *Services Marketing: People, Technology, Strategy*. Pearson Australia, 8. edition, 2011.

[Men13] Tom Mens. Model Transformation: A Survey of the State of the Art. In Jean-Philippe Babau, Mireille Blay-Fornarino, Joël Champeau, Sylvain Robert, and Antonio Sabetta, editors, *Model-Driven Engineering for Distributed Real-Time Systems: MARTE Modeling, Model Transformations and their Usages*, pages 1–19. ISTE Ltd, 2013.

[Mie16] Satu Miettinen. Product Design: Developing Products with Service Applications. In Marc Stickdorn and Jakob Schneider, editors, *This Is Service Design Thinking*, pages 56–67. BIS Publishing, 2016.

[MJ12] Simon K. Milton and Lester W. Johnson. Service blueprinting and BPMN: a comparison. *Managing Service Quality: An International Journal*, 22(6):606–621, 2012.

[Mky08] Mkyong. How to read XML file in Java – (DOM Parser). `https://www.mkyong.com/java/how-to-read-xml-file-in-java-dom-parser/`, December 2008. Accessed: 2018-04-09.

[MML10] Jochen Meis, Philipp Menschner, and Jan M. Leimeister. Modellierung von Dienstleistungen mittels Business Service Blueprinting Modeling. In Oliver Thomas and Markus Nüttgens, editors, *Dienstleistungsmodellierung 2010*, pages 39–64. Physica-Verlag HD, 2010.

[Mor16] Blake Morgan. The Evolution Of Customer Service. `https://www.forbes.com/sites/blakemorgan/2016/04/18/the-evolution-of-customer-service/#ebda0c24428e`, April 2016. Accessed: 2018-05-06.

[ODA+09] Chun Ouyang, Marlon Dumas, Wil M. P. Van Der Aalst, Arthur H. M. Ter Hofstede, and Jan Mendling. From Business Process Models to Process-oriented Software Systems: The BPMN to BPEL Way. *ACM transactions on software engineering and methodology (TOSEM)*, 19(1):41–78, August 2009.

[OMG11] OMG. Business Process Model and Notation (BPMN) Version 2.0. Standard, Object Management Group (OMG), January 2011.

[Pol04] Przemyslaw Polak. Model-Driven Program Transformation of a Large Avionics Framework. In Gabor Karsai and Eelco Visser, editors, *International Conference on Generative Programming and Component Engineering*, pages 361–378. Springer-Verlag Berlin Heidelberg, October 2004.

[Pol13] Przemyslaw Polak. BPMN Impact on Process Modeling. In Marite Kirikova, Gundega Lazdane, Janis Grabis, and Ksenija Lace, editors, *Proceedings of the 2nd International Business and Systems Conference BSC 2013*, pages 26–32. Riga Technical University, November 2013.

[Pol14] Gregor Polančič. BPMN 2.0 Message Events Vs. Message Tasks. `http://blog.goodelearning.com/subject-areas/bpmn/message-events-vs-message-tasks/`, May 2014. Accessed: 2018-04-09.

[Pri] Princeton University. Portrait of G. Lynn Shostack. `https://news.princeton.edu/uploads/243/image/Shostack_G%20Lynn.jpg`. Accessed: 2018-05-06.

[Rea] Realtime Board. Tool for Service Blueprinting. `https://realtimeboard.com/app/`. Accessed: 2018-05-06.

[SA16] Daniel Strüber and Anthony Anjorin. Comparing Reuse Mechanisms for Model Transformation Languages: Design for an Empirical Study. In Harald Storrle, Michel R. V. Chaudron, Vasco Amaral, and Miguel Goulao, editors, *Second International Workshop on Human Factors in Modeling*, pages 27–32. ACM, October 2016.

[SDSB09] Maik Seyring, Utz Dornberger, Alfredo Suvelza, and Trevor Brynes. *Service Blueprinting Handbook*. International SEPT Program, May 2009. Universität Leipzig, `http://www.vgu.edu.vn/fileadmin/pictures/studies/MBA/Handbook_Service_Blueprinting.pdf`, Accessed: 2017-03-25.

[Ser15a] Camunda Services. Camunda Model-API. `https://docs.camunda.org/manual/7.8/user-guide/model-api/bpmn-model-api/`, 2015. Accessed: 2018-04-09.

[Ser15b] Camunda Services. Create a Model. `https://docs.camunda.org/manual/7.8/user-guide/model-api/bpmn-model-api/create-a-model/`, 2015. Accessed: 2018-04-09.

[Ser18a] Camunda Services. Camunda Modeler. `https://camunda.com/products/modeler/`, 2018. Accessed: 2018-04-09.

[Ser18b] Camunda Services. Camunda Products. `https://camunda.com/products/`, 2018. Accessed: 2018-04-09.

[Ser18c] Camunda Services. Package org.camunda.bpm.model.bpmn.instance. https://docs.camunda.org/javadoc/camunda-bpm-platform/7.8/?org/camunda/bpm/model/bpmn/instance/package-summary.html/, 2018. Accessed: 2018-04-09.

[Sha10] Arash Shahin. Service Blueprinting: An Effective Approach for Targeting Critical Service Processes – With a Case Study in a Four-Star International Hotel. *Journal of Management Research*, 2(2):116–132, 2010.

[Sho84] G. Lynn Shostack. Designing Services That Deliver. *Harvard Business Review*, 84115(1):132–139, 1984.

[Sho11] Doug Short. Charting The Incredible Shift From Manufacturing To Services In America. http://www.businessinsider.com/charting-the-incredible-shift-from-manufacturing-to-services-in-america-2011-9?IR=T, September 2011. Accessed: 2018-05-06.

[Sof18] Alfresco Software. Quick Start Guide. https://www.activiti.org/quick-start/, 2018. Accessed: 2018-04-09.

[Spr14] Springer Gabler Verlag. Geschäftsprozessmanagement. https://wirtschaftslexikon.gabler.de/definition/geschaeftsprozessmanagement-53196/version-184601, July 2014. Accessed: 2018-05-06.

[SS] Marc Stickdorn and Jakob Schneider. Customer Journey Map Example. http://thetoolkitproject.webflow.io/tool/customer-journey-canvas#sthash.PKfhSe62.dpbs. Accessed: 2018-05-06.

[SS16] Marc Stickdorn and Jakob Schneider. *This Is Service Design Thinking.* BIS Publishing, 6. edition, 2016.

[ST05] August-Wilhelm Scheer and Oliver Thomas. Geschäftsprozessmodellierung mit der ereignisgesteuerten Prozesskette. *Das Wirtschaftsstudium*, 34(8-9):1069–1078, August 2005.

[Sti16a] Marc Stickdorn. 5 Principles of Service Design Thinking. In Marc Stickdorn and Jakob Schneider, editors, *This Is Service Design Thinking*, pages 34–45. BIS Publishing, 2016.

[Sti16b] Marc Stickdorn. The Iterative Process of Service Design Thinking. In Marc Stickdorn and Jakob Schneider, editors, *This Is Service Design Thinking*, pages 124–135. BIS Publishing, 2016.

[Ube17] Uber. Uber Service. https://www.uber.com, 2017. Accessed: 2017-08-05.

[vdAtHW03] Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, and Mathias Weske. Business process management: A survey. In Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, and Mathias Weske, editors, *Proceedings of the 1st International Conference on Business Process Management*, pages 1–12. Springer-Verlag, June 2003.

[Whi05] Stephen White. Using BPMN to model a BPEL process. *BPTrends*, 3(3):1–18, March 2005.

[Whi12] Stephen A. White. New Capabilities for Process and Interaction Modeling in BPMN 2.0. In Layna Fischer, editor, *BPMN 2.0 Handbook*, pages 17–32. Future Strategies, 2012.

[Wik18] Wikipedia. Comparison of Business Process Modeling Notation tools. `https://en.wikipedia.org/wiki/Comparison_of_Business_Process_Modeling_Notation_tools`, April 2018. Accessed: 2018-05-06.

[Wil02] James M. Wilson. Gantt charts: A centenary appreciation. *European Journal of Operational Research*, 149(2003):430–437, 2002.

[Wit04] Ole Wittko. *Die konzeptionellen Grundlagen des ServiceBlueprint*. FernUniverstät Hagen, 2004.

[Woo97] Julia T Wood. *But I Thought You Meant...: Misunderstandings in Human Communication*. McGraw-Hill, 1. edition, 1997.

[WvdAD+06] Petia Wohed, Wil M. P. van der Aalst, Marlon Dumas, Arthur H. M. ter Hofstede, and Nick Russell. On the Suitability of BPMN for Business Process Modelling. In Schahram Dustdar, José Luiz Fiadeiro, and Amit P. Sheth, editors, *Proceedings of the 1st International Conference on Business Process Management*, pages 161–176. Springer-Verlag, September 2006.

[YS10] JianHong Ye and Wen Song. Transformation of BPMN Diagrams to YAWL Nets. *Journal of Software*, 5(4):396–404, April 2010.

[ZBG06] Valarie A. Zeithaml, Mary Jo Bitner, and Dwayne D. Gremler. *Services Marketing: Integrating Customer Focus Across the Firm*. McGraw-Hill, 4. edition, 2006.