Chapter 11
Persuasive Technologies and Behavior Modification Through Technology: Design of a Mobile Application for Behavior Change

Andreas Hamper, Isabella Eigner, and Alexander Popp

11.1 Introduction

As of 2017, smartphone usage is still rapidly increasing, even in developed countries. Almost two-thirds of the entire US population is expected to own a smartphone at the end of the year, in comparison to 2012, where the penetration rate was below 40% (Statista: Smartphone Market 2016). Even though, only 2.97% of all available apps at the Apple App Store are in the Health & Fitness category (Statista: App Store Categories 2016), about 60% have downloaded at least one of them (Krebs and Duncan 2015). The reasons for using these kinds of apps are the adoption of a healthier lifestyle or tracking numbers, like weight or calories (Dennison et al. 2013).

However, there are two problems with the usage of health and fitness applications. On the one hand, there is a significant proportion of users (45.7%) who discontinue using the downloaded apps (Krebs and Duncan 2015). Making people reopen the app for a long period of time is not only desirable for the developer but also the only way to support a long-term behavior change, like a healthier lifestyle (Dennison et al. 2013).

On the other hand, only about 30% of users reported that their health improved considerably (Krebs and Duncan 2015). Although there are currently over 75,200 apps in the Health & Fitness category at the App Store (Apple: Health and Fitness Apps 2016), most of them have the same features as well as the same functionality. Additionally, this majority is developed only for people that are already physically active and does not support a long-term behavior change (Hofer 2016).

The two problems can be summarized to a low usage of downloaded health and fitness apps as well as limited effects for the majority of users, which lead to the

A. Hamper (✉) · I. Eigner · A. Popp
University Erlangen-Nuremberg, Institute of Information Systems, Nuremberg, Germany
e-mail: andreas.hamper@fau.de; isabella.eigner@fau.de

N. Wickramasinghe, J. L. Schaffer (eds.), *Theories to Inform Superior Health Informatics Research and Practice*, Healthcare Delivery in the Information Age,
https://doi.org/10.1007/978-3-319-72287-0_11

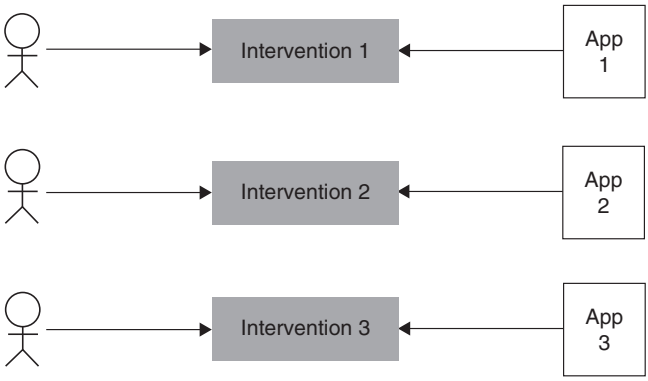

Fig. 11.1 Matching users with applications

following research question: How can behavioral interventions, in form of fitness apps, be tailored to specific users to increase their usage as well as effects?

The goal is to develop a process to identify apps for individual users at a specific point of time to support a long-term behavior change.

In order to achieve this, there has to be a methodology in place to describe and discover characteristics of users that are connected to influencing factors of behavior change. Furthermore, multiple applications have to be analyzed regarding the same influencing factors. This analysis can be built on the already existing examination of 29 different fitness apps regarding their characteristics of the transtheoretical model (TTM) and Fogg's Behavior Model (FBM) (Hofer 2016).

By translating these common factors to interventions for behavior change, they provide a method to match users with applications. The image below outlines this principle (Fig. 11.1).

Finally, a prototype will be built to demonstrate this process. It will analyze the user and match him with a suitable health and fitness service. Previous studies, though, discovered a lack of applications for certain characteristics (Hofer 2016), so the missing functionality will also be implemented to complete this demonstration.

11.2 Influencing User Behavior for Health and Fitness

Behavioral interventions appear to be more effective if they are individually tailored to the target (Bock et al. 2001), as well as perceived more positively and with a greater impact (Spittaels et al. 2007). However, tailoring behavioral interventions for each target individually leads to an exponentially increasing effort. There are numerous factors altering and affecting the behavior which make it challenging to limit or control their correlations. As a consequence, the exponential effort can be confined by clustering all possible targets into target groups. These represent a distinct set of properties and characteristics to which interventions, to a certain

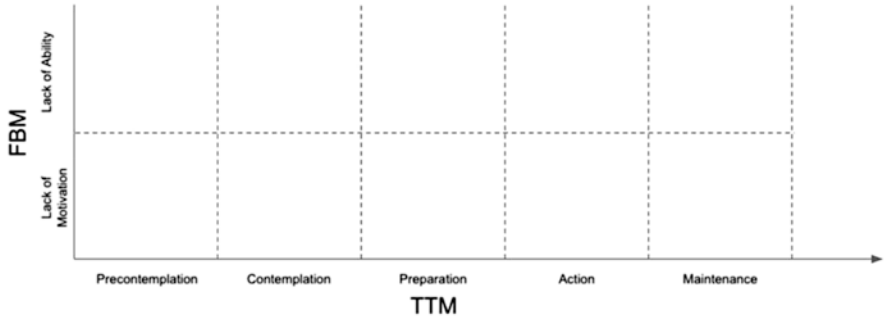

Fig. 11.2 Sociodemographic target groups

degree, can be tailored to. It might be necessary for some use cases to have extremely individual behavioral interventions, which can be achieved by increasing the number of target groups up to the number of original targets. For the results of this work, it is sufficient to have ten different groups.

There are many studies developing or elaborating new models for target groups. Most of them differ on the applications or purposes they are intended for. However, Rütten et al. (2007) take one model into account, the transtheoretical model (TTM), which focuses on behavior change. It describes a process with five different stages (precontemplation, contemplation, preparation, action, and maintenance), which individuals must go through to change and maintain a certain behavior (Glanz et al. 2008). The TTM has been used in multiple studies and can be used to develop and apply behavioral interventions (Fig. 11.2).

Nevertheless, these five stages are not enough to tailor precise interventions since there is no distinction between different types of people within one stage. TTM and FBM can be seen as a two-dimensional matrix. The FBM lists different personal factors, which influence human behavior. It helps to understand how and why people move from one stage of the TTM to the next (Fogg 2009). To simplify the resulting target groups, participants will be differentiated only by their combined core motivators and simplicity factors. These two factors will be referenced as "lack of motivation" and "lack of ability".

The combination of the TTM and FBM results into a matrix with ten different values (Fig. 11.3). Each of these values represents one sociodemographic target group. In this work, the term "sociodemographic target group" is defined by a target group that can be identified by exactly one TTM stage and a set of components from the FBM.

In order to decide how to classify participants into sociodemographic target groups, psychological questionnaires can be used:

The first one identifies the TTM stage by asking questions about the past, present, and future of the intended behavior. It is based on the staging algorithm, described by the Psychologisches Institut Freiburg (2001), which exploits the fact that the stages are linked in a timely manner (Prochaska et al. 1992).The second one

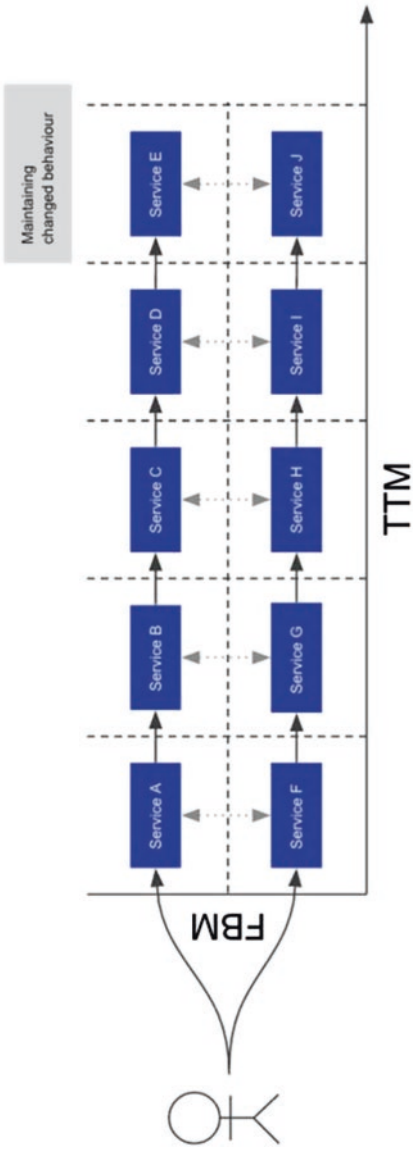

Fig. 11.3 Services for target groups

simply asks for the presence and significance of elements of motivation and elements of simplicity from Fogg's Behavior Model (Fogg 2009).

As a result, ten sociodemographic target groups, associated with processes of change from the TTM and behavior influencing factors from the FBM, can be differentiated. Furthermore, with the two questionnaires, there is a reliable and repeatable method to categorize people into these target groups.

Since the goal of this work is to change a behavior and preferably also maintain it, people have to get to the last stage of the TTM. This is where the desired behavior is already achieved and "people are working to prevent relapse" (Prochaska and Velicer 1997, p. 39). To get there, according to the TTM, the processes of change can be utilized to help targets proceed to the next stages. According to Prochaska and Velicer (1997, p. 39), "processes of change provide important guides for intervention programs." As explained above, tailored interventions are more effective; thus a tailored intervention to a specific target group helps them to move to the next stage and ultimately change and maintain a certain behavior.

Derived from the processes of change, this work will use services to create and apply interventions. As a result, one or more services that incorporate a process of change can be applied to progress to another stage.

Furthermore, these services should not only be tailored to the TTM stage but also to the influencing factors of the FBM to fully match with the target groups. The classification of services into target groups has been demonstrated by Hamper et al. (2016) with a Delphi study. By analyzing applications regarding their usefulness for each TTM stage and appealing to each Fogg factor, they can "be placed inside a two-dimensional matrix" (Hamper et al., 2016, p. 3353), which in turn represents all sociodemographic target groups.

One important difference between the two dimensions of the target groups is that the categorization into the TTM is temporally based (Prochaska and Velicer 1997), in contrast to the FBM. The influencing factors of the FBM do not necessarily have to be static, but they can be (Fogg 2009). The image below demonstrates the progress of a user through the different stages with the help of different services. For each target group, there is one individually tailored service.

However, it is quite complex to differentiate and work with so many services, especially if the vertical axis may be broken down further into more fine-grained FBM factors. To achieve an additional layer of standardization and abstraction, the term "service archetype" will be used for services that are in one TTM stage. It is based on the definition by the openEHR Foundation (2007) which states that archetypes are "expressions of a domain content model," "are all expressed in the same formalism," and are "defined for a wider reuse." In this work, a service archetype is an abstract structure, consisting of at least one service, which has an objective and several manifestations that explain how to achieve that objective.

11.2.1 Technical Requirements for Tailored Interventions

To translate the tailored interventions from the section before into actual software, it is necessary to develop concrete requirements to "reap the benefits of reduced integration and test costs, higher software reliability and maintainability, and more user-responsive[ness]" (Boehm 1984, p. 75) and, of course, simply to understand what it has to do and how (Wiegers and Beatty 2013). This section will first describe the theoretical background of requirements and then combine it with the service archetypes. The final result will be detailed requirements that can be used to actually implement this concept.

Software requirements describe a necessity for something, often combined with a prioritization and time specification. Nevertheless, there are many different levels and types of requirements for multiple purposes like project management or quality assurance. In order to keep it as clear and simple as possible, the focus will be on two very specific types that are used to describe systems: functional and nonfunctional requirements (Wiegers and Beatty 2013). Since the differentiation is of great importance for an exact tailoring, this section will introduce and describe both types.

Functional requirements describe "the necessary task, action or activity that must be accomplished" (Lightsey 2001, p. 36). It is not about in what way or manner this can be achieved but only about what it "must be able to do." The Institute of Electrical and Electronics Engineers (1984, p. 20) tries to summarize functional requirements into specifying the "inputs, processing, and outputs" of the software so that it can behave like a "finite state machine." It basically means that the system behaves solely, but also repeatable and predictable, based on its input and past behavior (Institute of Electrical and Electronics Engineers 1984). To summarize, functional requirements specify a concrete task, its necessary information, and possible results.

Nonfunctional requirements, on the other hand, are not related to the functionality of the software (Chung and do Prado Leite 2009). Although there are many different definitions and there is "still no consensus in the requirements engineering community what non-functional requirements are" (Glinz 2007, p. 21), the most common interpretation is that they are "attribute[s] of or constraint[s] on a system" (Glinz 2007, p. 25). They describe nonfunctional characteristics and general attributes of the software, like "efficiency, human engineering or understandability" (Chung and do Prado Leite 2009, p. 365).

A visualization by Glinz (2007) demonstrates the different classifications of requirements (Fig. 11.4). It indicates that functional as well as nonfunctional requirements are part of the system requirements. The important conclusion of this visualization is that it is necessary to define both types, functional and also nonfunctional, to properly design and develop a whole system.

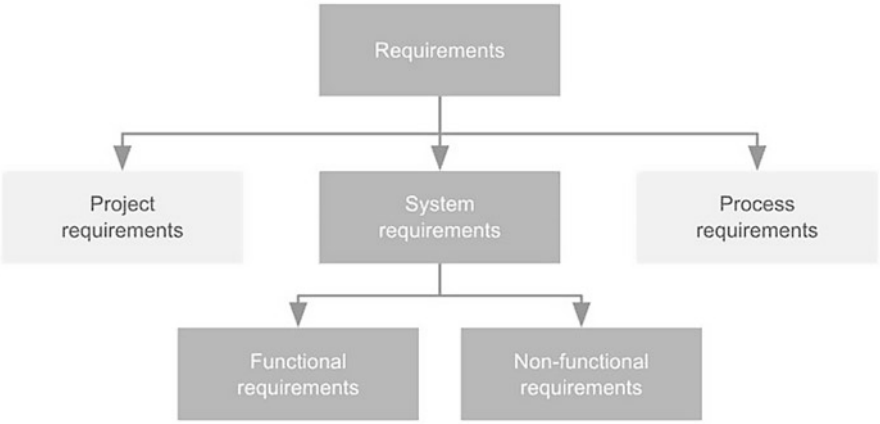

Fig. 11.4 Classification of requirements

11.2.2 Defining Requirements for Services

Concrete functional, as well as nonfunctional requirements, can now be derived from the service archetypes. Five different archetypes with multiple manifestations are necessary to change and maintain a certain behavior.

Due to the fact that each service archetype has an exact "objective," it can be matched with the definition of functional requirements, which can be described as a "concrete task" with a result. Furthermore, the manifestations of service archetypes "explain how to achieve that objective," have multiple constraints, and are individually tailored to the FBM. Considering the definition of nonfunctional requirements by Glinz (2007, p. 25), which states that they are "attribute[s] of or [a] constraint[s] on a system," it can be matched with the different manifestations.

As a consequence, the objective of a service archetype is determined by the TTM stage and represents functional requirements. The manifestations, in turn, are determined by the FBM factors and influence nonfunctional requirements. Figure 11.5 documents these correlations.

Consecutively, the requirement derivations for one service archetype can be applied to Fig. 11.6. This results into five service archetypes with distinct functional but a common set of two different nonfunctional requirements.

The next step is to define concrete requirements for each archetype, which can be used in the following sections to implement it into a prototype.

Functional requirements are already established by Prochaska et al. (1992) by means of the processes of change within the TTM. These are defined as "activities people use to get through stages" and "provide guides for intervention programs" (Glanz et al. 2008, p. 101), hence they directly relate to the definition of functional requirements. Moreover, there are "systematic relationships between people's stages and the processes" (Glanz et al. 2008, p. 105). Considering this and the fact

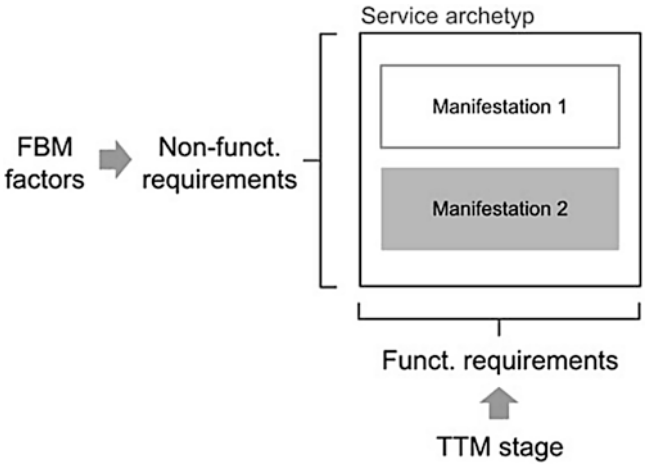

Fig. 11.5 Requirement derivation for a service archetype

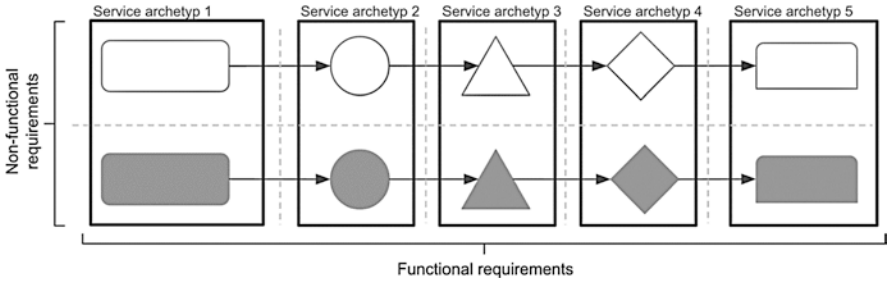

Fig. 11.6 Requirements for service archetypes

that each service archetype is placed at exactly one stage, the processes of change can be mapped to the five different archetypes.

- *Service archetype 1 (risk and fitness)*

- Raising consciousness for a particular behavior and its "causes, consequences, and cures" (Glanz et al. 2008, p. 101).
- *Service archetype 2 (coaching and advice)*
- Combining "increased emotional experiences" with an "anticipated relief" as soon as the desired behavior has taken place (Glanz et al. 2008, p. 101).
- *Service archetype 3 (progress monitoring)*
- Illustrating, cognitively as well as affectively, effects and consequences of the desired behavior to one's self-image (Glanz et al. 2008).
- *Service archetype 4 (rewards)*
- Reconfirming and supporting the "belief that one can change and the commitment and recommitment to act on that belief" (Glanz et al. 2008, p. 101).
- *Service archetype 5 (social and competition)*

- Adding signals to carry out the desired behavior as well as removing or avoiding signals for unwanted behaviors (Glanz et al. 2008).

In contrary to the section above, nonfunctional requirements have to be developed manually, because they are not directly related to the FBM factors.

Since there are many different nonfunctional requirements, the first step is to detect which ones can actually influence FBM factors. Distinct keywords, so-called indicator terms, in a text can be used to identify nonfunctional requirements. By applying the top 15 indicator terms of this study to the description of the FBM factors by Fogg (2009), the relevant requirements can be extracted.

Especially two requirements, "look and feel" and "usability," have an actual influence on multiple FBM factors. Another finding is that some of the factors cannot be influenced at all by nonfunctional requirements.

The next step is to specify the influence of "look and feel" as well as "usability" on both types of manifestations of service archetypes. They can be described as "lack of motivation" and "lack of ability," which in turn means that either elements of motivation or elements of ability have to be supported. This can be achieved by comparing characteristics of existing software with both types of FBM elements and the corresponding nonfunctional requirements. A list of 29 different health-related services, assembled by Wendt and Hofer (Hofer 2016), served as a base for this examination. Subsequently, a set of features can be described for both manifestations, lack of ability and lack of motivation, based on the, previously discovered, influencing requirements.

11.3 Prototypical Implementation of Tailored Interventions for Mobile Devices

Based on the requirements of the previous section, technical guidelines for the implementation as a mobile application will be described in the following section. The structure of the software, as well as the interactions with the user, is also laid out and explained. Furthermore, several rules and guidelines for organizing and building up the code base are mentioned along with the underlying reasons. The result does not only provide basic knowledge for other developers to further improve this prototype but also as a demonstration of how the previously described theoretical model can actually be implemented.

As already described, the platform has two tasks: firstly, it has to analyze the user and, secondly, it has to recommend suitable services. These recommendations have the form of simple links or redirections to existing services from other developers. This leads to two external parties. The use case diagram below (Fig. 11.7) demonstrates the connections and relations between the platform and the external actors.

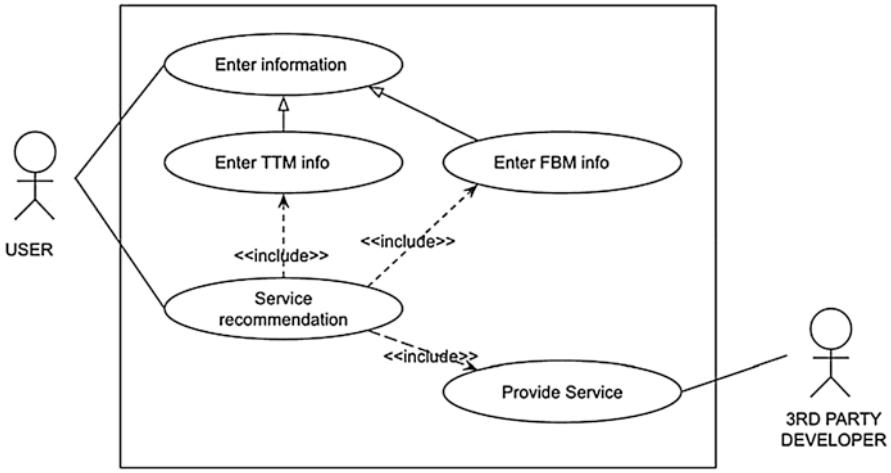

Fig. 11.7 Use case diagram for the platform

The first external actor is the developer who provides the necessary services. There may be multiple developers, but they can be regarded as one entity, especially when the Google Play Store is taken into account.

The second external actor is the user, who has two different interactions. On the one hand, he can enter some information about himself, which consists of data about his influencing FBM factors and current TTM stage. And on the other hand, he has access to service recommendations. These are based on the personal information about himself as well as the available third-party services.

We create a so-called feed. Since the classification into one TTM stage is not completely reliable, there won't be just one service recommendation. Instead, one service for each stage will be presented and arranged according to its relevance. As a result, there are five sorted "ServiceCards," one for each TTM stage (Prochaska and Velicer 1997), which act as links to services and have additional information, like the name of the service or the relevance. These are displayed within the feed and represent the service archetypes from Sect. 11.3.2. This structure fits perfectly to the role as a prototype. It does not focus on the user experience but on the demonstration of this concept. Future enhancements should tackle this issue and move the focus from a prototype to a release version. The image below (Fig. 11.8) visualizes this concept.

There are five ServiceCards, sorted by their relevance according to the TTM stage, the user is currently in, with multiple links. As stated in the requirements, there might be the case that there is only one link with an adapting service. This scenario is demonstrated for "ServiceCard 3" at Fig. 11.8. Another scenario, as displayed for "ServiceCard 5," is two links: one for lack of motivation and one for lack of ability.

Nevertheless, it is possible that the user simply does not like the recommended service. For this scenario, there will be another adaption. Some ServiceCards do not

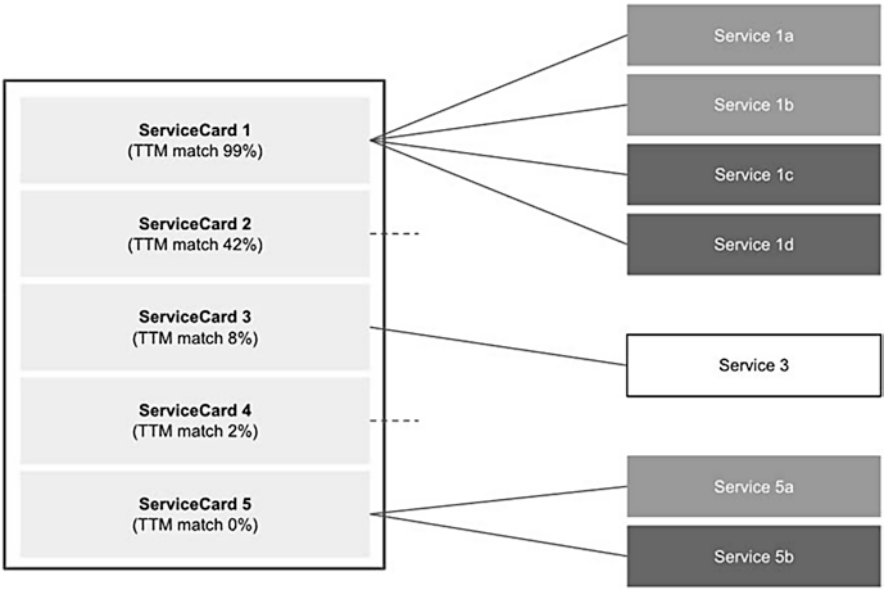

Fig. 11.8 Feed with five ServiceCards

just have one alternative but a list of equally matching services, the user can choose from ("ServiceCard 1").

11.3.1 Software Design Pattern and Abstraction

To "reduce system complexity" and build "reusable software," the goal is to use appropriate design patterns to structure the code. However, in this prototype, they have been used on two different tiers but for the same above- described purposes.

On a very abstract level they ensure "high reusability and interoperability" (Cortez and Vazhenin 2013, p. 132) of the code itself. Correspondingly, they have nothing to do with the concrete requirements but only with general code quality. Examples for this are the "abstract factory" or the "chain of responsibility."

On a more detailed level, these patterns are not designed on a class level but on a software component level, which means they are modeled out of entire fully functional components. The platform itself is built out of three software components: the data layer, logic layer, and presentation layer. It is important not to confuse this division into three with the model-view-controller pattern, which can rather be classified into the previously described, abstract type of patterns (Cortez and Vazhenin 2013). Instead, each of these three components can consist of multiple other patterns; they are only named after their main responsibility (Fig. 11.9).

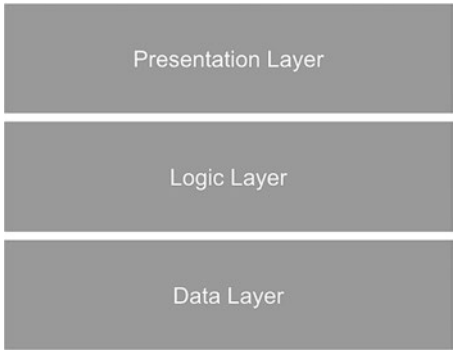

Fig. 11.9 Overview of the platform system architecture

The main task of the data layer is to provide the FBM and TTM related data. This includes retrieving it from the user as well as storing and keeping it up to date. Based on the information, provided by the data layer, the logic layer will generate a list of five ServiceCards. These have to be filled with content and sorted according to the current TTM stage of the user. Additionally, the corresponding services have to be filtered by the influencing FBM factors. The main responsibility of the presentation layer is to render these ServiceCards properly on the user interface. Furthermore, it has to provide the necessary information and ensure a smooth redirecting of the user to the external services.

The following three sections will give a more detailed insight into how these components are structured as well as their concrete tasks, interfaces, and relations.

Previous sections already indicated that the system of linking to other services has one flaw. Necessary applications with matching FBM factors for the right TTM stage might not exist. According to Hofer (2016), this problem is especially serious for the first two stages, since there are almost no applications tailored for the precontemplation and contemplation stage. In contrast, there are a huge number of services for the later stages. As a consequence, this section will outline the structure, design, and implementation of a service, tailored for the first stage of the TTM. Additionally, this service will adapt automatically to match to the influencing FBM factors of the user. To illustrate the software design, the following section describes the design of services for the precontemplation stages in technical detail based on software design patterns.

11.3.2 Implementation

People are usually in the precontemplation stage "because they are uninformed or underinformed about the consequences of their behavior" (Prochaska and Velicer 1997, p. 39). They can be moved to the next stage by raising consciousness and awareness of the consequences and effects of their lifestyle. They need to understand that there is a problem that has to be solved, which can be done with education as

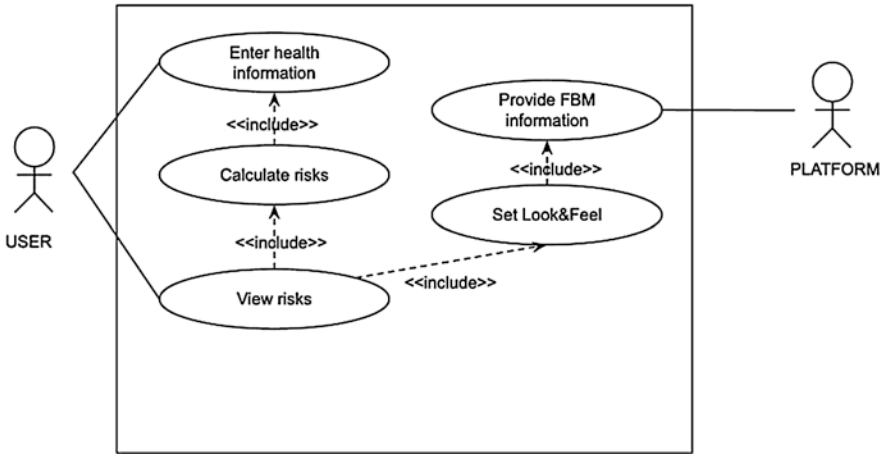

Fig. 11.10 Use case diagram for the service

well as confrontation (Prochaska and Velicer 1997). The main goal of the prototype is to increase the physical activity of the user, so this service should focus on educating and confronting about problems that can actually be tackled by that.

As a result, the implemented service has two tasks. Firstly, it has to calculate some risk factors, based on the health and physical condition of the user. Secondly, it has to adapt its nonfunctional factors dynamically according to the FBM, which relies on data provided by the platform. The use case diagram below (Fig. 11.10) visualizes these tasks with its corresponding external actors.

On the one hand, there is the user, who can enter information about his physical condition. This data is necessary to calculate the risk to suffer from selected problems and diseases. On the other hand, there is the platform that can provide information about the FBM profile of the user. This separation highlights the concept of service archetypes. The whole service is one archetype with exactly specified functional requirements: calculating risks according to fixed algorithms. These risks, however, are presented differently to the user, depending on the influencing FBM factors, which translate to different nonfunctional requirements.

11.3.2.1 Data Layer

The algorithms and questionnaires to calculate health risks rely on multiple types of data. As already mentioned, some need the user to enter a simple number or a free text, while others are developed in a multiple-choice style. Additionally, some information are not known to the user or are constantly changing which leads to accessing external sources to obtain it. However, not all users are using external services, for example, for tracking their daily steps, so this service has to handle this case as well. In a nutshell, obtaining the data has to be abstract, to support multiple

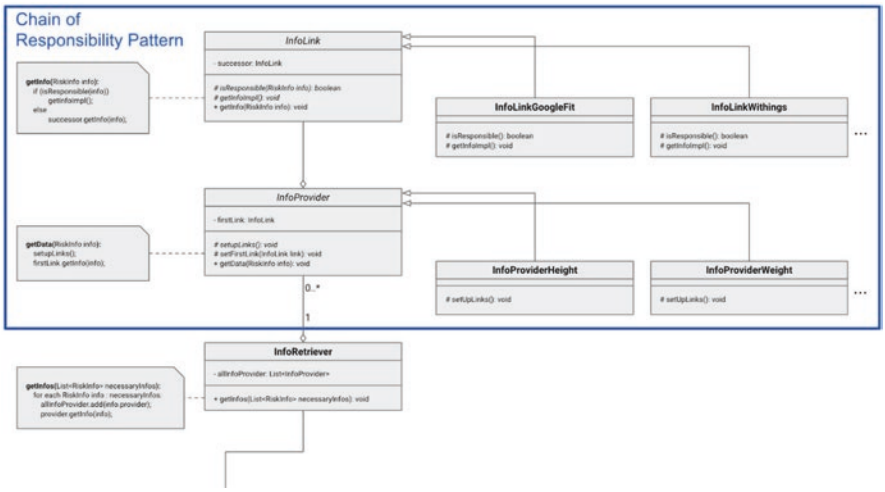

Fig. 11.11 Class diagram of InfoProvider with InfoLinks

data types, reusable for multiple algorithms that need the same information, and flexible in case of unreliable sources.

To achieve these three requirements, the concept of "InfoProviders" has been developed. One InfoProvider is responsible for obtaining exactly one specified information and has a reusable interface. It is decoupled from data sources which makes it abstract and can be used for multiple risk algorithms that need the same information. Each InfoProvider, in turn, consists of at least one "InfoLink." Their main task is to actually retrieve information from a source. In contrary to the InfoProviders, InfoLinks have a fixed data source but can return multiple types of information. To further increase reusability, they can be reused for various InfoProviders. Furthermore, each InfoLink has the possibility to signal that it cannot retrieve the desired information at the moment, which makes it possible to connect them in series and create a fallback system. Due to their abstract interface, InfoLinks can be used in any order and combination for every InfoProvider.

As a result, this concept has a high degree of abstraction, because of the fixed InfoProvider interfaces, and reusability, because of the possibility to reuse on the one hand the same InfoProvider for multiple algorithms and on the other hand the same InfoLink for multiple InfoProviders. Moreover, the arbitrary order and combination of InfoLinks lead to more flexibility.

The concrete implementation is done with the "chain-of-responsibility" design pattern which is highlighted in the class diagram below (Fig. 11.11). This design pattern should be used, whenever there are "more than one object [that] may handle a request" (Vlissides et al. 1995, p. 253), like retrieving a certain information, and results in reduced coupling between a sender and receivers as well as added flexibility.

There are two abstract classes, *InfoProvider* and *InfoLink*, which have concrete inheritances for certain information and data sources. The *InfoRetriever* acts as a unified interface for the logic layer to conveniently access the data. The most important detail is the "GetData(...)" method. Whenever it is called, the concrete implementation of an InfoProvider first has to set up its InfoLinks (*setupLinks()*). This basically creates all necessary InfoLink objects and orders them correctly. After that, the first InfoLink will be called for the desired information. The concrete InfoLink implementation, in turn, implements two methods that are structured and called by their abstract superclass, as demonstrated in "GetInfo(...)." The first method checks whether it can provide this exact data from the data source, and the second one actually returns it. Additionally, it has a reference to the next InfoLink, so it can redirect the request if it cannot provide the data.

11.3.2.2 Logic Layer

The main responsibility of the logic layer is to calculate health risks. However, it is quite difficult, especially for a non-expert user, to correctly understand and interpret the results. Without any background knowledge or comparisons, it is hard to know whether a value, for example, the risk to suffer from a stroke, is good or bad. As a consequence, there won't be just the calculation of the result but also a percentage-based valuation. This classifies the absolute values into a scale from 0 to 100%, whereby "100%" means that it is the best and correspondingly "0%" the worst possible outcome. Another adaption will be the possibility to swap out algorithms. There are numerous ways to calculate these health risks. Some are more suitable for certain demographics (Li et al. 2014) and some proof to be correct only after a long period of time (Assmann et al. 2002). Furthermore, science will progress so there might always be the case to change an algorithm. To tackle this problem, the six health risks will be fixed, but their way of calculation can be changed dynamically. Just like the data layer, the logic layer is also based on the principles of well-known design patterns. The main part is structured according to the template pattern which "lets subclasses redefine certain steps of an algorithm without changing the […] structure" (Vlissides et al. 1995, p. 360). This is useful, since there are always the same three parts in the same order but with different implementations: retrieving necessary data, calculating the health risk, and evaluating the result.

The class diagram above (Fig. 11.12) outlines the structure of the logic layer. To have access to information about the user, each "RiskAlgorithm" has a reference to the InfoRetriever, which was already described in the data layer. The method "calculate(…)" of the abstract class RiskAlgorithm is the core of the template pattern. It structures the flow of the calculation by combining functions of subclasses with the InfoRetriever. However, there is one addition to the classical pattern. There is not only one level of subclasses below the RiskAlgorithm but two. The first level represents the six different health risks and the second one the actual implementations of the different algorithms. In combination, the lowest level, e.g., "RiskAlgorithmCardioPROCAM," incorporates the calculation of health risks. Its

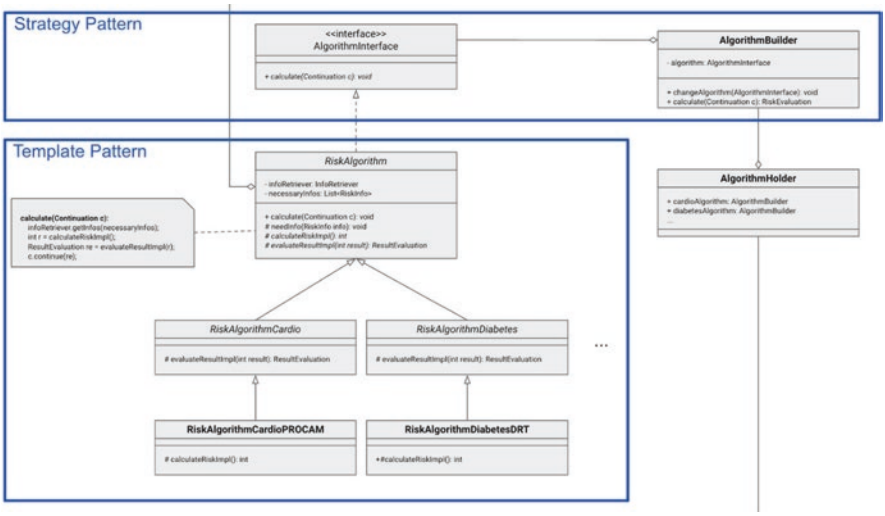

Fig. 11.12 Class diagram of the logic layer

superclass in turn, e.g., "RiskAlgorithmCardio," can evaluate this calculation. The final superclass "RiskAlgorithm" actually calls these functions and redirects their results.

To be able to swap out algorithms, preferably during run time, the strategy pattern is put on top of the template. Its use case is to "define a family of algorithms, encapsulate each one, and make them interchangeable" (Vlissides et al. 1995, p. 349) so that they are independent from the code that uses them. As illustrated above, there is an "AlgorithmInterface" on top of the RiskAlgorithm to have fixed interfaces, which can be used by the compositor. In this scenario, the "AlgorithmBuilder" is the compositor, whose responsibility is to have a reference to one AlgorithmInterface and calls its "calculate" function or swap out the entire reference whenever the algorithm needs to be changed. Whenever an object wants to trigger the calculation of an algorithm, it needs to call the corresponding function of the AlgorithmBuilder. This abstraction decouples the RiskAlgorithm from the triggering object.

11.3.2.3 Presentation Layer

The next step is to display the calculated and evaluated health risks properly, according to the influencing FBM factors of the user. To achieve this, the view has to be able to adapt or change to different requirements that are presented in Table 11.1: look and feel and usability. Since not only the user interface has to change (look and feel) but also its behavior (usability), the concept of fragments fits perfectly. They are Android-specific elements whereby "each fragment defines its own layout and its own behavior" (Android: Fragments 2016). This also highlights why the system

Table 11.1 Features for service archetype manifestations

	Features for	
	Lack of ability	Lack of motivation
Usability	• Simple and efficient usage • Style of usage according to OS guidelines • Easy to understand • No frills	• Informational • Clear and distinct status information
Look and feel	• Simply structured • Identical/similar representations • Succinctly	• Extreme colors • Clear connection between progress and UI • Emotional pictures and text

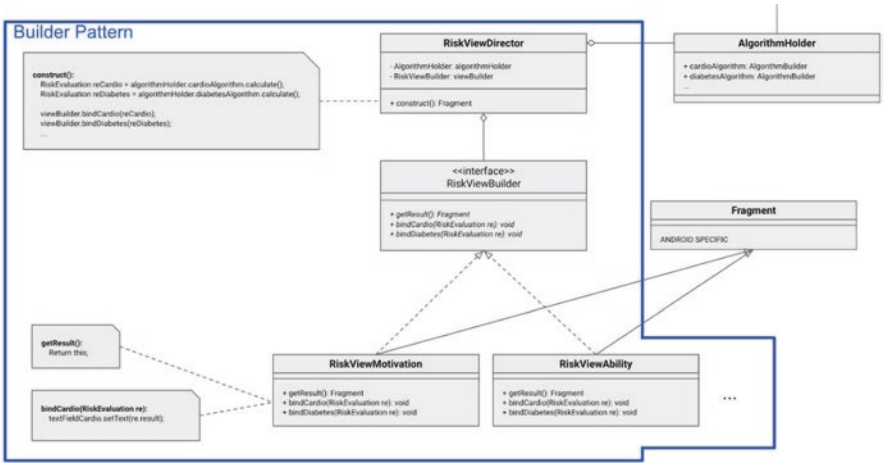

Fig. 11.13 Class diagram of the presentation layer

architecture of the service cannot be compared with the model-view-controller pattern. The presentation layer consists not only of a plain view, which would only affect the look and feel, but also some logic, or controller code, to influence the behavior and thereby the usability differently.

This leads to two fragments with different user interfaces and actions, depending on the FBM factors. One for "lack of motivation" and one for "lack of ability." Nonetheless, they receive exactly the same data from the logic layer. This can be realized with the builder pattern which allows to "separate the construction of a complex object from its representation so that the same construction process can create different representations" (Vlissides et al. 1995, p. 110). In this case, the representation is a fragment with only two different manifestations. The second section indicated that the split of the FBM into two groups might be broken down even further in the future. This architecture combined with the builder pattern supports this change since it would be easy to simply add more representations (Fig. 11.13).

The central point of the presentation layer is the “RiskViewDirector.” It contains the “AlgorithmHolder” which encapsulates the six AlgorithmBuilder, presented in the previous section. Additionally, it will be instantiated with a “RiskViewBuilder,” depending on the FBM factors that can be retrieved from the shared preferences. The initial instantiation might look like the following.

Summed up, the presentation layer combines fragments with evaluated health risks, depending on FBM factors. The health risks, in turn, are calculated and evaluated at the logic layer and are combined and accessible through one AlgorithmHolder. To be able to calculate these risks, the logic layer needs access to information that can be provided by the data layer through the InfoRetriever. These fixed interfaces enable fast and easy changes for future developments.

11.4 Solutions and Impact for Consumers

The objective of this work was to demonstrate with the help of a prototype how behavioral interventions, in form of fitness apps, can be tailored to specific users to increase their usage as well as their effects. These interventions aim to lead to a long-term behavior change, like being more physically active.

We described the theoretical framework behind tailored interventions. We stated that users can be classified into ten different target groups, based on a questionnaire that analyzes their TTM stage as well as influencing FBM factors. The goal was to move users to the last TTM stage which is synonymous with maintaining the changed behavior.

The target groups, in turn, can be divided into five service archetypes, whereby each of these has the goal to move the user to the next target group regarding its TTM stage. The archetypes differ only in the way they try to achieve this goal, the processes of change. Additionally, each service archetype has two manifestations that indicate different ways to influence the user. These two variants are derived from a set of influencing FBM factors.

The prototype is modeled strongly on this theoretical framework. It first navigates the user through a tutorial with a questionnaire to determine the most suitable target group. Afterward, a list of ServiceCards is displayed. Each one represents exactly one service archetype. Based on the target group of the user, these ServiceCards can be sorted, so that the most relevant service archetype is at the first and the least relevant at the last position.

By just linking from a ServiceCard to an already existing app, which incorporates the matching process of change, they do not have to be developed all over again. However, there is a lack of suitable applications for the first service archetype or the first TTM stage. For this reason, a service, the risk test, has been implemented to fill this gap. To address the two manifestations, there is not just one link to a fixed service; instead, the user will be redirected to another application depending on the target group he is currently in whenever the service is not capable of adapting what was demonstrated at the risk and fitness test.

Fig. 11.14 Matching users with applications

To develop a process to decide to which services can be linked from which archetypes and which manifestations, the two components of the target groups, the TTM and FBM, are translated to functional and nonfunctional requirements. This leads to a set of different tailored requirements for each target group which allows to either match existing applications with them or to exactly tailor new services.

The image above (Fig. 11.14) demonstrates the combination of the theoretical framework and the actual implementation. There is still the classification into ten target groups, whereby the shape indicates the service archetype and the color the manifestation. By means of a questionnaire, users can be put into one target group. In this example, user A is in the white rounded rectangle, user B in the white triangle, and user C in the black diamond.

As a result, each one of these users will get recommendations for different services. User A starts at the beginning with the motivational version of the self-implemented service. As he progresses through the TTM stages, he will finally get the same recommendation as user B currently has. User C has a different set of influencing FBM factors, so he will be redirected to different applications. The sequences below (Fig. 11.15) highlight that users will have different, individually tailored recommendations to services that incorporate the exact process of change with the matching influencing FBM factors to move them to the next TTM stage, which ultimately leads to a behavior change. The task of the last service is to keep the user at this stage and maintain the changed behavior.

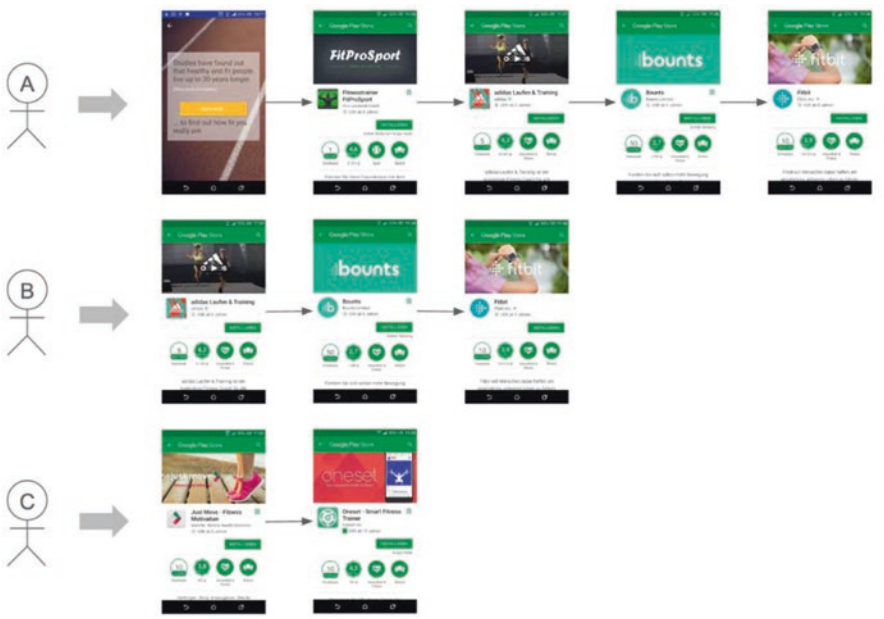

Fig. 11.15 Tailored recommendations of different services

Two problems of current health and fitness applications are addressed. They very often have only a limited effect, and many users stop using them after a while. Both issues prevent the change and maintaining of a healthier lifestyle. The prototype, resulting from this work, tackled these problems by incorporating a theoretical framework to increase usage as well as effects of fitness apps. However, the purpose of the implementation was to move from a theoretical to a practical perspective and demonstrate this framework in form of an Android application. It is not yet intended for normal users because there are some features that serve well for demonstrating the theory but not for a practical usage.

As already described, the feed, the list of five ServiceCards, should be simplified. Early test users, who were not familiar with the TTM concept, were confused by the amount of content. Only the first ServiceCard is important, so the last four can be hidden. Since this basically makes the list layout ineffective, a complete change of the main screen can be useful. A similar problem occurred at selecting a concrete service. At the moment, the user can choose between several applications, which appeared to be too confusing.

According to Krebs and Duncan (2015), a big proportion of users discontinue using an app because it takes too much time to enter data. This is likely to be the case for the questionnaire at the beginning. Solutions might be to reduce the number of questions or delay them until they are absolutely necessary. The FBM classification, for example, is only needed when the actual service has to be selected and not for rendering the list of ServiceCard at the beginning.

Eventually, it might be interesting to implement these recommended changes and compare the long- term success of this application with several others that are currently available. By increasing the number of target groups, which allows more precise recommendations, or adjusting the match with services, there are multiple changes that can be made to perform extensive A/B tests.

In conclusion, users can be analyzed and receive individually tailored recommendations to different services. These services have various functionalities and are each one step to changing and maintaining the desired behavior to be more physically active.

References

Android: Fragments. (2016). Retrieved December 19, 2016, from https://developer.android.com/guide/components/fragments.html

Apple: Health & Fitness Apps. (2016). Retrieved December 19, 2016, from https://itunes.apple.com/us/genre/ios-health-fitness/id6013

Assmann, G., Cullen, P., & Schulte, H. (2002). Simple scoring scheme for calculating the risk of acute coronary events based on the 10-year follow-up of the prospective cardiovascular Münster (PROCAM) study. *Circulation, 105*(3), 310–315.

Bock, B. C., Marcus, B. H., Pinto, B. M., & Forsyth, L. (2001). Maintenance of physical activity following an individualized motivationally tailored intervention. *Annals of Behavioral Medicine, 23*(2), 79–87.

Boehm, B. W. (1984). Verifying and validating software requirements and design specifications. *IEEE Software, 1*(1), 75–88.

Chung, L., & do Prado Leite, J. C. (2009). On non-functional requirements in software engineering. In *Conceptual modeling: Foundations and applications* (pp. 363–379). Berlin Heidelberg: Springer.

Cortez, R., & Vazhenin, A. (2013). Developing re-usable components based on the Virtual-MVC design pattern. In *International Workshop on Databases in Networked Information Systems* (pp. 132–149). Springer Berlin Heidelberg.

Dennison, L., Morrison, L., Conway, G., & Yardley, L. (2013). Opportunities and challenges for smartphone applications in supporting health behavior change: Qualitative study. *Journal of Medical Internet Research, 15*(4), e86.

Fogg, B. J. (2009). A behavior model for persuasive design. In *Proceedings of the 4th International Conference on Persuasive Technology* (p. 40). ACM.

Glanz, K., Rimer, B. K., & Viswanath, K. (Eds.). (2008). *Health behavior and health education: Theory, research, and practice*. San Francisco: John Wiley & Sons.

Glinz, M. (2007). On non-functional requirements. In *15th IEEE International Requirements Engineering Conference (RE 2007)* (pp. 21–26). IEEE.

Hofer, S. (2016). *Consumer Healthcare Wearables – A teardown of health and fitness solutions to understand their impact on behavior change*. Master's Thesis, Friedrich Alexander Universität Erlangen-Nürnberg.

Institute of Electrical and Electronics Engineers. (1984). *IEEE guide to software requirements specifications*. IEEE.

Krebs, P., & Duncan, D. T. (2015). Health App use among US mobile phone owners: A national survey. *JMIR mHealth uHealth, 3*(4), e101.

Li, K., Hüsing, A., & Kaaks, R. (2014). Lifestyle risk factors and residual life expectancy at age 40: A German cohort study. *BMC Medicine, 12*(1), 59.

Lightsey, B. (2001). *Systems engineering fundamentals*. Ft. Belvoir: Defense Acquisition University.

openEHR Foundation. (2007). In T. Beale & S. Heard (Eds.), *Archetype definitions and principles. 1.0*.

Prochaska, J. O., DiClemente, C. C., & Norcross, J. C. (1992). In search of how people change: applications to addictive behaviors. *American Psychologist, 47*(9), 1102.

Prochaska, J. O., & Velicer, W. F. (1997). The transtheoretical model of health behavior change. *American Journal of Health Promotion, 12*(1), 38–48.

Psychologisches Institut Freiburg. (2001). Erfassung der "Stages of Change" im Transtheoretischen Modell Prochaska's – eine Bestandsaufnahme. *Forschungsberichte des Psychologischen Instituts der Albert-Ludwigs-Universität Freiburg i. Br*.

Rütten, A., Abu-Omar, K., Adlwarth, W., & Meierjürgen, R. (2007). Sedentary lifestyles. Classification of different target groups for the promotion of health-enhancing physical activities. *Gesundheitswesen (Bundesverband der Arzte des Offentlichen Gesundheitsdienstes (Germany)), 69*(7), 393–400.

Spittaels, H., Bourdeaudhuij, I. D., Brug, J., & Vandelanotte, C. (2007). Effectiveness of an online computer-tailored physical activity intervention in a real-life setting. *Health Education Research, 22*(3), 385–396.

Statista: App Store Categories. (2016). Retrieved December 19, 2016, from https://www.statista.com/statistics/270291/popular-categories-in-the-app-store/

Statista: Smartphone Market. (2016). Retrieved December 19, 2016, from https://www.statista.com/topics/2711/us-smartphone-market/

Vlissides, J., Helm, R., Johnson, R., & Gamma, E. (1995). Design patterns: Elements of reusable object-oriented software. *Reading: Addison-Wesley, 49*(120), 11.

Wiegers, K., & Beatty, J. (2013). *Software requirements*. London: Pearson Education.

Andreas Hamper studied Information Systems at the University of Erlangen-Nuremberg. Since 2011, he is a member of the research staff and PhD student at the Institute of Information Systems. In his research he is focusing on mobile technologies and health care.

Isabella Eigner studied International Information Systems at the University of Erlangen-Nuremberg. Since 2014, she is a member of the research staff and PhD student at the Institute of Information Systems. In her research she is focusing on data mining in health care.

Alexander Popp studied Information Systems at the University of Erlangen-Nuremberg. In 2016, he did his master's work on mobile technology for preventive health care at the Institute of Information Systems.