Chapter 13
Social Sustainability in the e-Health Domain via Personalized and Self-Adaptive Mobile Apps

Eoin Martino Grua, Martina De Sanctis, Ivano Malavolta, Mark Hoogendoorn, and Patricia Lago

Abstract Within software engineering, social sustainability is the dimension of sustainability that focuses on the "support of current and future generations to have the same or greater access to social resources by pursuing social equity." An important domain that strives to achieve social sustainability is e-Health, and more recently e-Health mobile apps.A wealth of e-Health mobile apps is available for many purposes, such as lifestyle improvement and mental coaching. The interventions, prompts, and encouragements of e-Health apps sometimes take context into account (e.g., previous interactions or geographical location of the user), but they still tend to be rigid, e.g., apps use fixed sets of rules or they are not sufficiently tailored toward individuals' needs. Personalization to the different users' characteristics and run-time adaptation to their changing needs and context provide a great opportunity for getting users continuously engaged and active, eventually leading to better physical and mental conditions. This chapter presents a reference architecture for enabling AI-based personalization and self-adaptation of mobile apps for e-Health. The reference architecture makes use of a dedicated goal model and multiple MAPE loops operating at different levels of granularity and for different purposes. The proposed reference architecture is instantiated in the context of a fitness-based mobile application and exemplified through a series of typical usage scenarios extracted from our industrial collaborations.

E. M. Grua (✉) · I. Malavolta · M. Hoogendoorn · P. Lago
Vrije Universiteit Amsterdam (VU), Amsterdam, The Netherlands
e-mail: e.m.grua@vu.nl; i.malavolta@vu.nl; m.hoogendoorn@vu.nl; p.lago@vu.nl

M. De Sanctis
Gran Sasso Science Institute (GSSI), L'Aquila, Italy
e-mail: martina.desanctis@gssi.it

C. Calero et al. (eds.), *Software Sustainability*,
https://doi.org/10.1007/978-3-030-69970-3_13

13.1 Introduction

e-Health mobile apps are designed for assisting end users in tracking and improving their own health-related activities [1]. With a projected market growth of US$102.3 billion by 2023, e-Health apps represent a significant market [2] providing a wide spectrum of services, i.e., life style improvement, mental coaching, sport tracking, and recording of medical data [3]. The unique characteristics of e-Health apps wrt other health-related software systems are that e-Health apps (1) can take advantage of smartphone sensors, (2) can reach an extremely wide audience with low infrastructural investments, and (3) can leverage the intrinsic characteristics of the mobile medium (i.e., being always on, personal, and always carried by the user) for providing timely and in-context services [4].

However, even if the interventions, prompts, and encouragements of current e-Health apps take context into account (e.g., previous interactions or geographical location of the user), they still tend to be *rigid* and not fully tailored to individual users, e.g., by using fixed rule sets or by not considering the unique traits and behavioral characteristics of the user. In this context, we see *personalization* [5] and *self-adaptation* [6–8] as effective instruments for getting users continuously engaged and active, eventually leading to better physical and mental conditions. The addition of intervention tailoring (via personalization and self-adaptation) is a crucial step in addressing the main sustainability concern that e-Health mobile apps want to achieve: *social sustainability*. By providing better interventions, we are not only more likely to have the user interested in maintaining engagement with the app but also help the user achieve better physical and mental conditions by allowing the app to better address the personal needs and by extension the social ones too.

In this work, we combine personalization and software self-adaptation to provide users of mobile e-Health apps with a better, more engaging and effective experience. To this aim, we propose a *reference architecture (RA) that combines data-driven personalization with self-adaptation*. The main design drivers that make the proposed reference architecture unique are:

- The combination of multiple *Monitor–Analyze–Plan–Execute* (MAPE) loops [9] operating at different levels of granularity and for different purposes, e.g., to suggest users the most suitable and timely activities according to their (evolving) health-related characteristics (e.g., active vs. less active), but also to cope with technical aspects (e.g., connectivity hiccups, availability of IoT devices and third-party apps on the user's device) and the characteristics of the physical environment (e.g., indoor vs. outdoor, weather).
- A *dedicated goal model* for representing health-related goals via a descriptive concise language accessible by healthcare professionals (e.g., fitness coaches, psychologists).
- The exploitation of our *online clustering algorithm* for efficiently managing the evolution of the behavior of users as multiple time series evolving over time. This online clustering algorithm has been already extensively tested in a previously

published article [10], showing promising results by doing better than the current state of the art.

The main characteristics of the proposed reference architecture are the following: (1) it caters the personalization of services to specific user preferences (e.g., preferred sport activities); (2) it guarantees the correct functioning of the features via the use of connected IoT devices (e.g., a smart-bracelet) and runtime adaptation strategies; (3) it adapts the services depending on contextual factors such as environmental conditions and weather; (4) it supports a smooth participation of domain experts (e.g., psychologists) in the personalization and self-adaptation processes; and (5) it can be applied in the context of a single e-Health app and by integrating the services of third-party e-Health apps (e.g., already installed sport trackers). All of these characteristics are shown in this work by evaluating the reference architecture and the goal model with fitness coaching scenarios. We want to emphasize how most characteristics have been engineered with the main goal of achieving social sustainability. Possible exceptions are characteristics (2) and (5), which more specifically addresses technical sustainability of the reference architecture. Our emphasis on social sustainability will be further explained and explored throughout the chapter.

Lastly, in a previous study [11] we reported a preliminary version of our Reference Architecture (RA). Here we extend the work by: (1) framing the work in the overall context of social sustainability, (2) document the methodology used to design our RA, (3) report a scenario-based evaluation of our RA, (4) provide a goal model to be used with the RA, (5) a viewpoint definition used to create the view of our RA.

13.2 Background

The notion of *reference architecture* (RA) is borrowed from Volpato et al. [12], who define it as "a special type of software architectures that provide a characterization of software systems functionalities in specific application domains," e.g., SOA for service orientation and AUTOSAR for automotive. In the context of this study, a *self-adaptive software system* is defined as a system that can autonomously handle changes and uncertainties in its environment, the system itself, and its goals [7].

For the definition of *personalization*, we build on that by Fan and Poole [5] and define it as "a process that changes a system to increase its personal relevance to an individual or a category of individuals." Furthermore, to enhance personalization, we use CluStream-GT (CluStream for Growing Time-series) [10]. CluStream-GT was chosen for this RA as it is the state-of-the-art clustering algorithm for time-series data (especially within the health domain). CluStream-GT works in two phases: offline and online. First, the *offline phase* initializes the algorithm with a small initial dataset; this is done either at design time or at the start of runtime. Afterward, during the *online phase* the algorithm clusters the data that is being collected at runtime. Clustering allows the RA to group similar users together, where similarity is determined by the data gathered from the apps. This gives the RA a more sustainable

and scalable method of personalization, without requiring to create individual personalization strategies but maintaining a suitable degree of personalization [10, 13]. An example case where clustering can be used to aid personalization is with the use of cluster-based Reinforcement Learning [14].

The methodology used for the design of our RA is the one presented by Angelov et al. [15] (see Fig. 13.1), where the authors present their RA Framework to facilitate software architects in the design of *congruent RAs*, i.e., RAs where the design, context, and goals are explicit and coherent (adapted from [15]).

The RA Framework (or framework for short) consists of two elements: a multidimensional classification space, and a set of predefined RA types (and variants of these types). The former, through the use of strict questions and answers, supports software architects in classifying RAs according to their *context* (**Where?**, **Who?**, and **When?** questions in Fig. 13.1), *goals* (**Why?** in Fig. 13.1), and *design* (**How?** and **What?** in Fig. 13.1) dimensions. The latter consists of specific combinations of values from the multidimensional space. These types, and variants, are used to evaluate the congruence of the RA being designed. If a RA is congruent (i.e. matches a type or variant) it has a greater chance of becoming a success, where, by success the authors mean ". . . the acceptance of the architecture by its stakeholders and its usage in multiple projects" [15]. For each dimension, the authors have defined subdimensions with respective questions and answers. During the design of our RA we have worked with each dimension and, with the use of the framework, classified our RA according to the possible values available for each subdimension. As knowledge of our RA and its components is necessary to understand the design process, we further explain the use of the framework in Sect. 13.7.

In recent years a larger body of work on software engineering and software architecture address sustainability. Sustainability can be divided into four dimensions: technical, economical, environmental, and social [16]. Within this work we present an RA for the e-Health domain with the main goal of better addressing the social dimension of sustainability, whilst the technical contributions of this work include the combination of AI and self-adaptation. In this work we build on the following definition of social sustainability: "focusing on supporting current and future generations to have the same or greater access to social resources by pursuing generational equity. For software-intensive systems, this dimension encompasses the direct support of social communities, as well as the support of activities or processes that indirectly create benefits for such communities" [16].

13.3 Related Work

Several RAs for IoT can be found in the literature [17–20]. In particular, Bauer et al. [19] present several abstract *architectural views* and *perspectives*, which can be differently instantiated. The adaptation of the system's configuration is also envisioned, at an abstract level. IoT-A [18] aims to be easily customized to different needs, and it makes use of *axioms* and *relationships* to define connections among

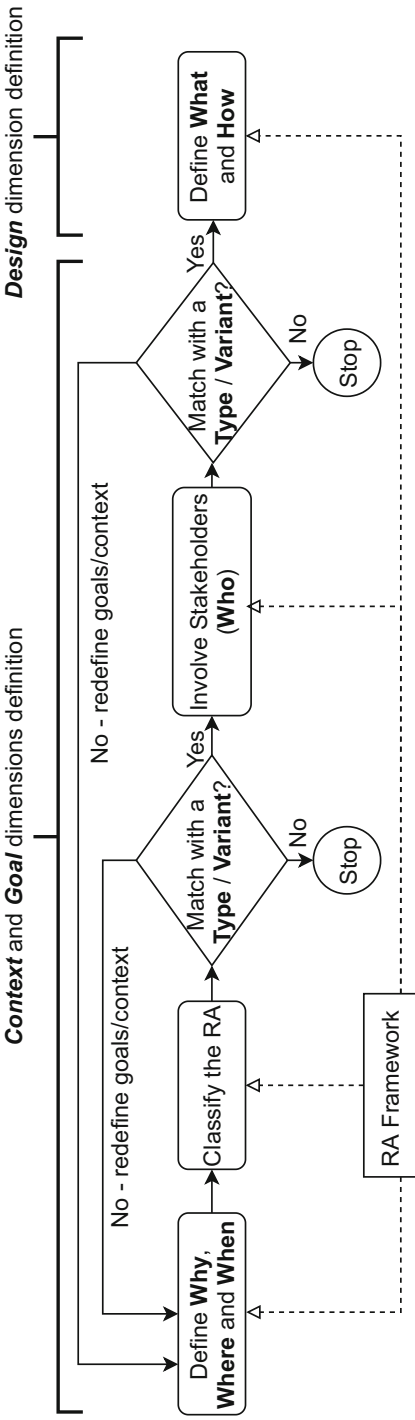


Fig. 13.1 Methodology for the design of our RA [15]

IoT entities. Industrial Internet Reference Architecture (IIRA) [17] is particularly tailored for industrial IoT systems. Web Services Oxigenated (WSO2) [20] presents a layered structure and targets scalability and security aspects too. All of the above RAs are abstract and domain independent. As such, they do not address required features specific to the IoT-based e-Health domain. Moreover, they lack the needed integration with AI for personalization used to tailor interventions to the user's health-related characteristics, an important technique used by the RA to address social sustainability.

Other works providing service-oriented architectures (SOAs) focused on adaptation but neglected user-based personalization. For example, Feljan et al. [21] defined an SOA for planning and execution (SOA-PE) in Cyber Physical Systems (CPS) and Mohalik et al. [22] proposed a MAPE-K autonomic computing framework to manage adaptivity in service-based CPS. Morais et al. [23] present RAH, a RA for IoT-based e-Health apps. RAH has a layered structure, and it provides components for the prevention, monitoring and detection of faults. Different from RAH, our RA explicitly manages the self-adaptation of e-Health mobile apps, both at the user and architectural levels. Mizouni et al. [24] propose a framework for designing and developing context-aware adaptive mobile apps. Their framework lacks other types of adaptation, i.e., adaptation for user personalization and adaptation with other IoT devices—which is possible with our RA.

Lopez and Condori-Fernandez [25] propose an architectural design for an adaptive persuasive mobile app with the goal of improving medication adherence. Accordingly, the adaptation is here focused only on the messages given to the user and lacks the other levels of adaptation (environment adaptation, etc.) that our RA covers. Kim [26] proposes a general RA that can be used when developing adaptive apps and implements a e-Health app as an example. However, being general, the RA lacks the level of detail present in our work, the integration of AI for personalization, and a way for involving domain experts in app design and operation, which is essential in adaptive e-Health.

In summary, to the best of our knowledge, ours is the first RA for e-Health mobile apps that simultaneously supports (1) *personalization* for the different users, **Users** by exploiting users' smart objects and preferences to dynamically get data about, e.g., their mood and daily activities, and (2) *runtime adaptation* to user needs and context in order to keep them engaged and active, so that we can better address social sustainability.

13.4 Reference Architecture

Figure 13.2 shows our RA with the following stakeholders and components. Section 13.8 defines the corresponding viewpoint.

Users provide and generate the data gathered by the e-Health app. At first installation, the users are asked to input information to better understand their

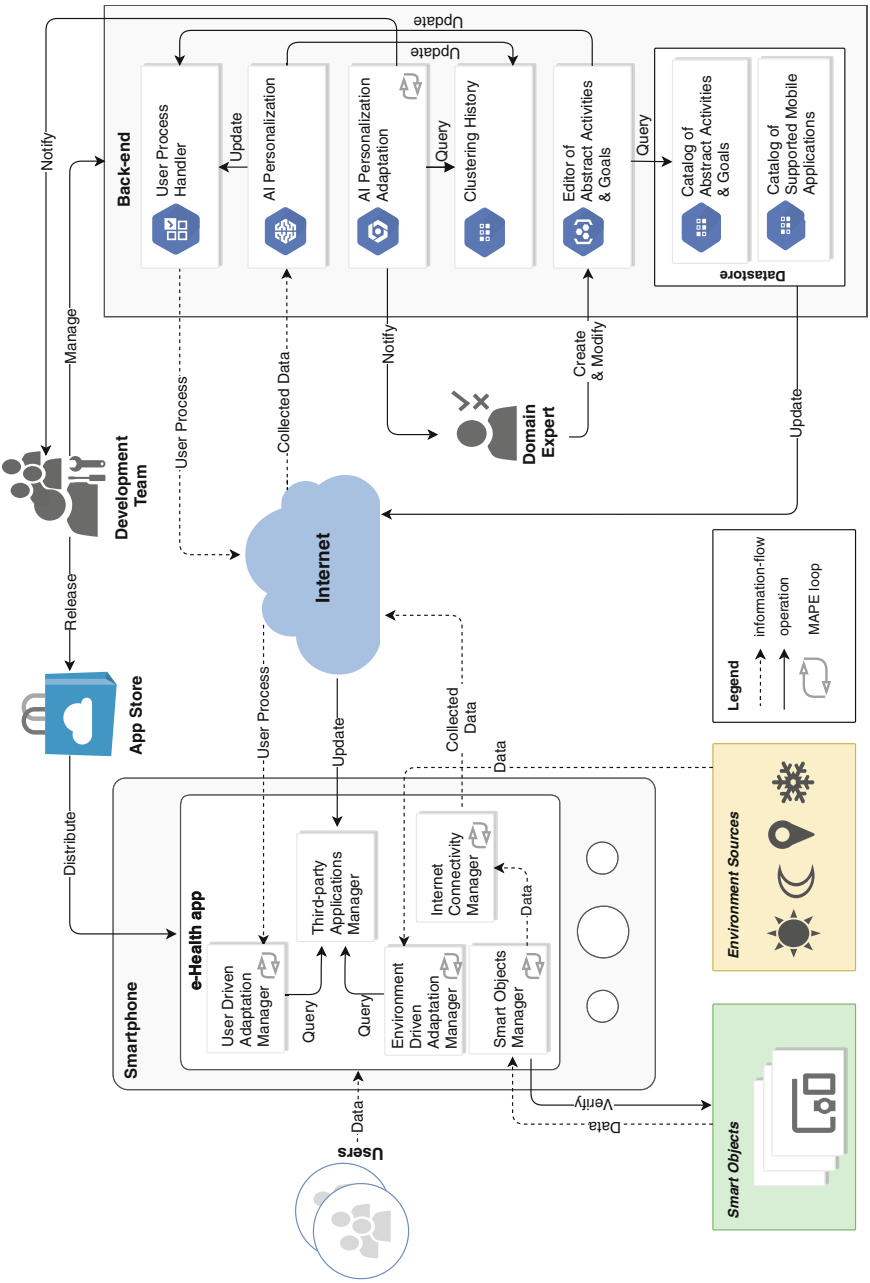


Fig. 13.2 Reference architecture for personalized and self-adaptive e-Health apps

aptitudes. After an initial usage phase and data collection, the system has enough information to assign them to a cluster.

The **smartphone** is the host where the self-adaptive e-Health app is installed. In the mobile app, four components, namely, *User Driven Adaptation Manager*, *Environment Driven Adaptation Manager*, *Smart Objects Manager*, and *Internet Connectivity Manager*, implement a MAPE loop to dynamically perform adaptation. The *Third-party Applications Manager*, in turn, is responsible for communication with third-party apps supported by the RA that can be exploited by the e-Health app both during its nominal execution and when adaptation is performed. It is also responsible for storing user preferences. Further details on these components are given in Sect. 13.5.

Smart Objects are devices, other than the smartphone, that the app can communicate with. They are used to gather additional data about the users as well as augmenting the data collected by the smartphone sensors. For instance, a smartwatch would be used by the app to track the user's heartrate, therefore adding extra information on the real-time performance of the user.

Environment is the physical location of the user, and its measurable properties. It is used by the e-Health app to make runtime adaptations according to its current operational context and to the user's scheduled activities, as described in Sect. 13.5.5.

The back-end of our RA (right-hand side in Fig. 13.2) is Managed by a *Development team*. It additionally exposes an interface to the *Domain Expert* that is also involved in the e-Health app design and operation. The back-end contains the components needed to store the collected user data and to manage the user clusters. It also hosts components supporting the general functioning of the app.

User Process Handler is in charge of sending User Processes to the users; it takes care of sending the same User Process to all users of the same cluster. A *User Process* is composed of one or more *Abstract Activities*. These activities are inspired by the ones introduced in [27], although they differ both in the structure and in the way they are refined, as explained later. An *Abstract Activity* is defined by a vector of one or more *Activity categories* and an associated goal, with each vector entry representing a day of the week. Examples of Abstract Activities are discussed later in Sect. 13.9.

Each *Abstract Activity* is defined by the Domain Expert via the *Editor of Abstract Activities & Goals* and later stored in the *Catalog of Abstract Activities & Goals*. Each Activity category identifies the kind of activity the user should perform. As an example, the user can receive either a *Cardio* or *Strength* Activity category and so should perform an activity of that kind. More precisely, for each user, the Activity categories are converted to *Concrete Activities* at runtime via the use of the *User Driven Adaptation Manager* and based on the user's preferences. For instance, a cardio Activity category can be instantiated into different *Concrete Activities* such as running, swimming, and walking. Moreover, if an *Abstract Activity* is composed of multiple Activity categories, all or some of type Cardio, they can be converted into different *Concrete Activities*. This implies that users who receive the same User

Process will still be likely to have different *Concrete Activities*, therefore personalizing the experience to the individual user (this is further discussed in Sect. 13.5.2).

The goals associated with an *Abstract Activity* are also important for distinguishing between Abstract Activities, besides converting them into *Concrete Activities*. Two Abstract Activities containing the same vector of Activity categories can be different solely based on their associated goal. More details on the goal model are given in Sect. 13.6.

The *User Process Handler* receives Updates from (1) the *AI Personalization* and the *Editor of Abstract Activities & Goals* in order to send User Processes to their associated users. The AI Personalization Updates the *User Process Handler* every time a user moves from one cluster to another, while the *Editor of Abstract Activities & Goals* Updates it every time new clusters are analyzed by the Domain Expert (along with the new associated User Process). These updates guarantee that the *User Process Handler* remains up to date about the User Processes and their associated users.

AI Personalization sends an Update to the *Clustering History* component whenever a change occurs in the clusters. The *AI Personalization* component uses the CluStream-GT algorithm to cluster users into clusters in a real-time and online fashion [10]. It receives the input data from the e-Health app (see Collected Data in Fig. 13.2). More than one instance of CluStream-GT can be running at the same time. In fact, there is one instance per category of data. For example, if the e-Health app is recording both ecological momentary assessment [28] and biometric data, one for the purpose of monitoring **mood** and the other for **fitness**, there will be two running instances of the algorithm.

AI Personalization Adaptation is in charge of monitoring the evolution of clusters and detecting if any change occurs. Examples include the merging of two clusters or the generation of a new one. To do so, it periodically Queries the *Clustering History* database. If one or more new clusters are detected, this component will Notify both the Development Team and the Domain Expert. The Domain Expert will examine the new information and add the appropriate User Process to the *Catalog of Abstract Activities & Goals* via the dedicated editor. In turn, the Development Team is notified just as a precaution so that it can verify if the new cluster is not an anomaly. The specifics of the corresponding MAPE loop are described in Sect. 13.5.1.

The role played by AI via the CluStream-GT algorithm is relevant in our RA as it strongly supports both personalization and self-adaptation, thus guaranteeing a continuous user engagement that is crucial in e-Health apps. Specifically, personalization is achieved by clustering the users based on their preferences and their physical and mental condition. This supports the RA in assigning appropriate *User Processes* to each user, and further adapt them to continuously cope with the current status of the user and by doing so better addressing social sustainability concerns.

Clustering History is a database of all the clusters created by the *AI Personalization* component. For each cluster it keeps all of the composing micro-clusters with all of their contained information.

Editor of Abstract Activities & Goals allows the Domain Expert to create and modify *Abstract Activities* (and their associated goals) and to combine them as User Processes. This is achieved via a web-based interactive UI and the editor's ability to Query the *Catalog of Abstract Activities & Goals*. It is also the editor's responsibility to update the *User Process Handler* if any new User Process has been created and is currently in use.

Catalog of Abstract Activities & Goals is a database of all User Processes that the Domain Expert has created for each unique current and past cluster. When a new cluster is defined, the Domain Expert can assign to it an existing User Process from this catalog, or create a new one and store it.

Catalog of Supported Mobile Applications is a database containing the metadata needed for interacting with supported third-party mobile apps installed on users' devices. This database stores information such as the specific types of Android intents (and their related extra data) needed for launching each third-party app, the data it produces after a tracking session, etc. Indeed, our e-Health app does not provide any specific functionality for executing the activities suggested to the user (e.g., running, swimming); rather, it brings up third-party apps (e.g., Strava[1] for running and cycling, Swim.com[2] for swimming) and collects the data produced by the apps after the user performs the physical activities. The main reasons for this design decision are: (1) we do not want to disrupt the users' habits and preferences in terms of apps used for tracking their activities and, (2) we want to *build on* existing large user bases; we do not want to reinvent the wheel by reimplementing functionalities already supported by development teams with years-long experience.

Whenever the e-Health app evolves by supporting new applications (or no longer supporting certain applications), the *Catalog of Supported Mobile Applications* Updates, through the *Datastore*, the *Third-party Applications Manager*. The *Third-party Applications Manager*'s responsibility is to keep the list of supported mobile apps up to date and provide the corresponding metadata to the *User Driven Adaptation Manager* and the *Environment Driven Adaptation Manager*, when needed.

The e-Health app and back-end communicate via the Internet. Specifically, the communication from the e-Health app to the back-end is REST based and it is performed by the *Internet Connectivity Manager*, which is responsible for sending the Collected Data to the *AI Personalization* component in the back-end. Communication from the back-end to the e-Health app is performed by the *User Process Handler*, which is in charge of sending the User Process to the e-Health app via push notifications.

[1] https://www.strava.com/

[2] https://www.swim.com/

13.5 Components Supporting Self-Adaptation

The RA has five components used for self-adaptation. To accomplish its responsibilities, each of these components implements a MAPE loop.

13.5.1 AI Personalization Adaptation

The main goal of the AI Personalization Adaptation is to keep track of the clusters evolution and to enable the creation of new User Processes. It does it through its MAPE loop depicted in Fig. 13.3.

During its Monitor phase, the AI Personalization Adaptation monitors the macro-clusters. In its Analyze phase it determines if there are changes in the monitored macro-clusters. To do so, the AI Personalization Adaptation periodically queries the Clustering History database. It compares the current clusters with the previously saved ones. If any of the current ones are significantly different, then the AI Personalization Adaptation enters its Plan phase. The Plan phase gathers the IDs of the users and macro-clusters involved in these significant changes. Since this change involves the need for the creation of new User Processes for all of the users belonging to the new clusters, the Domain Expert must be involved in this adaptation. To achieve this we have exploited the type of adaptation described in [29], which considers the involvement of humans in MAPE loops. In particular, in [29] the authors describe various cases in which a human can be part of a MAPE loop. AI Personalization Adaptation falls under what the authors refer to as: 'System Feedback (Proactive/foreground)'. This type of adaptation is initiated by the system which may send information to the human. The human (i.e., Domain Expert) uses this information to execute the adaptation (by creating the new User Processes necessary). To send the needed information to the Domain Expert, AI Personalization Adaptation takes the gathered knowledge from the Plan phase and gives it to Execute. Execute notifies (Fig. 13.2) both the Development Team and the Domain Expert about the detected cluster change(s) and relays the gathered information.

To determine if a cluster is significantly different from another, we use a parameter *delta*. This parameter is set by the Development Team at design time and determines how different the stored information of one cluster has to be from another

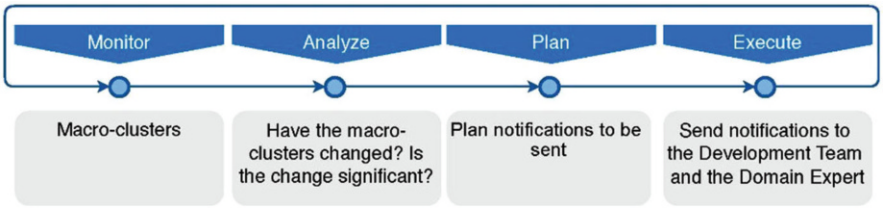


Fig. 13.3 AI Personalization Adaptation MAPE loop

one to identify them as unique. The Development Team is notified as a precaution, to double check the change and verify that no errors occurred.

13.5.2 User Driven Adaptation Manager

The main responsibility of the User Driven Adaptation Manager is to receive the User Process from the back-end and convert the contained Abstract Activities into Concrete Activities. A Concrete Activity represents a specific activity that the user can perform, also with the support of smart objects and/or corresponding mobile apps. As an example, *running* is a concrete activity during which the user can exploit a smart bracelet to monitor their cardio rate as well as a dedicated mobile app to measure the run distance and the estimated burned calories. A Concrete Activity is designed as a class containing multiple attributes that is stored on the smartphone. The attributes are:

- **Selectable**: This is True if the User Driven Adaptation Manager or the Environment Driven Adaptation Manager can choose this Concrete Activity, when dynamically refining Abstract Activities; False otherwise. It is set by the user via the user preferences.
- **Location**: This specifies if the activity is performed indoors or outdoors. This attribute is used by the Environment Driven Adaptation Manager to choose the appropriate Concrete Activity according to weather conditions (see Sect. 13.5.5).
- **Activity category**: This defines what type of category the Concrete Activity falls under; e.g., for a fitness activity, it specifies a cardio or strength training.
- **Recurrence**: This tracks how many times the user has performed the Concrete Activity in the past. It allows the User Driven Adaptation Manager to have a preference ranking system within all the selectable Concrete Activities.

For each user, the Concrete Activities are derived from their preferences stored in the Third-party Applications Manager. During its nominal execution, the User Driven Adaptation Manager is in charge of refining the Abstract Activities in the User Process into Concrete ones. To do this, it queries the Third-party Applications Manager and exploits its knowledge of the Concrete Activities and their attributes. After completing the task, the User Driven Adaptation Manager presents the personalized User Process to the user as a schedule, where each slot in the vector of Activity categories corresponds to a day. Therefore creating a personalized user schedule of Concrete Activities.

Refining a User Process is required every time that the user is assigned with a new process, to keep up with its improvements and/or cluster change. To this aim, a dynamic User Process adaptation is needed to adapt at runtime the personalized user schedule in a transparent way and without a direct user involvement. Figure 13.4 depicts the MAPE loop of the User Driven Adaptation Manager.

Once it accomplishes its main task of refining the User Process, the User Driven Adaptation Manager enters the Monitor phase of its MAPE loop by monitoring the

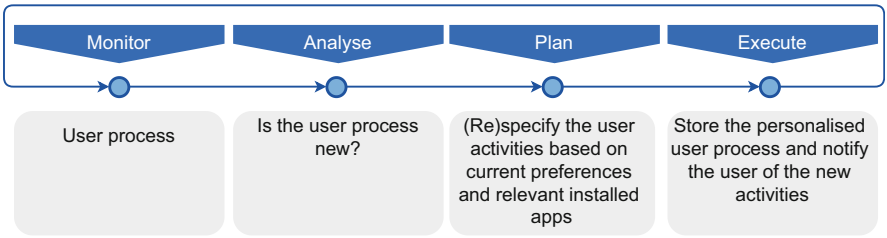


Fig. 13.4 User Driven Adaptation Manager MAPE loop

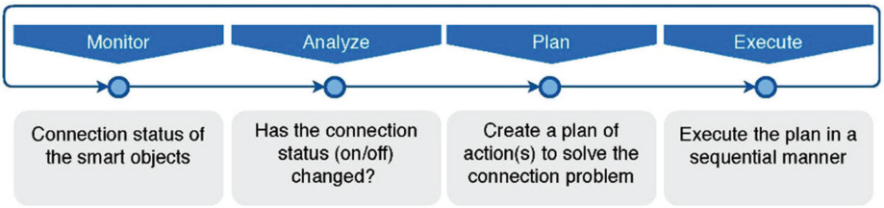


Fig. 13.5 Smart Objects Manager MAPE loop

User Process. The Analyze phase receives the monitored User Process from Monitor. Analyze is now responsible to determine if the user has been assigned a new User Process. If so, the User Driven Adaptation Manager converts the Abstract Activities in this new User Process into Concrete ones, taking into account the user preferences. It makes this conversion by finding suitable Concrete activities during the Plan phase. As all of the Abstract Activities have been matched with a corresponding Concrete activity, the Execute phase makes the conversion, storing this newly created personalized User Process and notifying the user about the new activity schedule.

13.5.3 Smart Objects Manager

This component aims to maintain the connection with the user's smart objects and, if not possible, find alternative sensors to make the e-Health app able to continuously collect user's data and, thus, to perform optimally. To this aim, it implements a MAPE loop, shown in Fig. 13.5, supporting the dynamic adaptation at the architectural level of the smart objects.

The Monitor phase is devoted to the run-time monitoring of the connection status with the smart objects. Connection problems can be due to either the smart objects themselves, which can be out of battery, or to missing Internet, Bluetooth or Bluetooth low energy connectivity. The Analyze phase is in charge of verifying the current connection status (received by Monitor) and see if the connection status

with any of the smart objects has changed. During the Plan phase, the MAPE will create a sequential plan of actions that the Execute will have to perform. All of the actions are aimed at reestablishing the lost connection or at finding a new source of data (e.g. reconnect, notify the user, find a new source of data). For instance, if the smart-watch connected to the smartphone runs out of battery and attempts to reconnect to it fail, the Smart Objects Manager will switch to sensors inbuilt in the smartphone (such as the accelerometer).

13.5.4 Internet Connectivity Manager

The main purposes of the Internet Connectivity Manager are to (1) to send the Collected Data to the back-end and store them locally when the connection is missing, and (2) to provide resilience to the e-Health app's Internet connectivity.

As shown in the MAPE loop in Fig. 13.6, during the Monitor phase the Internet Connectivity Manager runtime monitors the quality of the smartphone's Internet connection.

Analyze is then in charge of detecting whether a significant connection quality alteration is taking place. If so, the Internet Connectivity Manager enters the Plan phase and it plans for an alternative. The alternative can include switching the connection type or storing the currently collected data locally on the smartphone. As a new connection can be established, the component sends the data to the back-end to be used by the AI Personalization.

13.5.5 Environment Driven Adaptation Manager

One of the objectives of the e-Health app is keeping users constantly engaged, to ensure that they execute their planned schedule of activities. To this aim, the Environment Driven Adaptation Manager plays an important role, which is essentially supported by its MAPE loop, depicted in Fig. 13.7.

The purpose of this component is to constantly check whether the currently scheduled Concrete Activity best matches the runtime environment (i.e., weather

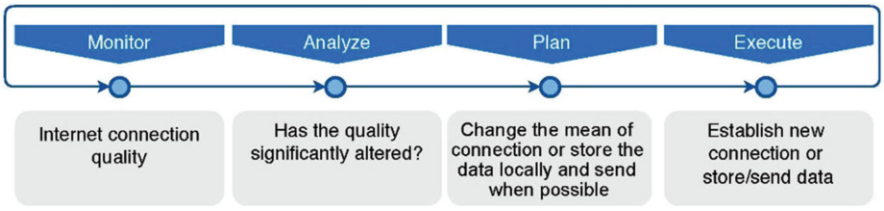


Fig. 13.6 Internet Connectivity Manager MAPE loop

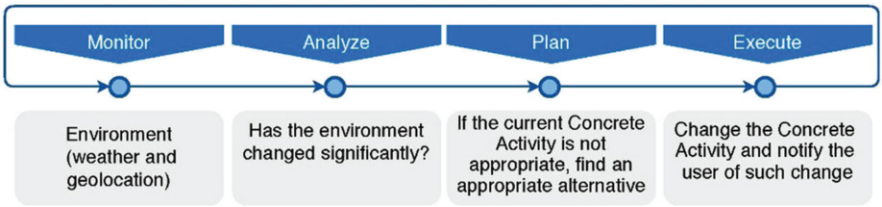


Fig. 13.7 Environment Driven Adaptation Manager MAPE loop

conditions) the user is located in. To do so, the Environment Driven Adaptation Manager monitors in runtime the user's environment. The Monitor phase periodically updates the Analyze phase by sending the environment data. This phase establishes if the environment significantly changed. If so, it triggers the Plan phase that verifies whether the currently planned Concrete Activity is appropriate for the user's environment. If it is not, it finds an appropriate alternative and sends the information to Execute. Execute swaps the planned Concrete Activity with the newly found one and notifies the user of this change.

13.6 Goal Model

Goals have been used in many areas of computer science for a long time. For instance, in AI planning they are used to describe desirable states of the world (e.g., [30]) whereas in goal-oriented requirements engineering (GORE [31]) they are used to model nonfunctional requirements (e.g., [32]). Goals have also been used in self-adaptive systems to express the desired runtime behavior of systems execution [27, 33]. More recently, goals are used to model personal objectives at the user level [34], as done in our work.

As stated before, a User Process is composed of one or more Abstract Activities, each defined as a vector of Activity categories with an associated *goal*. For each cluster, the Domain Expert defines its User Process and corresponding goals, through the *Editor of Abstract Activities & Goals*.

The syntax of our *goal model* is presented in Table 13.1. A goal of an Abstract Activity, namely, G_a, refers to the type of feature that the Abstract Activity represents (e.g., mood, fitness). At the current stage of our work, we have *mood-based goals*, m_g, and *fitness-based goals*, f_g.

A mood-based goal defines as objective a desirable mood that the user should reach, considering their specific pathology. A mood-based goal can be specified in two different ways: as a numerical value belonging to a given discrete range, such as $1, \ldots, n$, or as a string value belonging to a specific string set, such as *[very sad, sad, neutral, happy, very happy]*. This goal type establishes the target mood that users are expected to reach when performing mood-related activities. Specifically, we use the

Table 13.1 Goal model syntax

G_a:	::=	$m_g \mid f_g$
m_g	::=	**one_of** $STRING\_SET$?($FREQ$) |<$or \leq$ $or >$$or \geq$ $or =$ $value$ **in** [1, . . . ; n] ?($FREQ$)
f_g	::=	$INTENSITY_{time}$?($FREQ$) |$INTENSITY_{value}$?($FREQ$) |<$or \leq$ $or >$$or \geq$ $or = f_g$ | f_g **and** $f_g \mid f_g$ **or** f_g |**one_of** $seq\ f_g$ |T |$\perp$
$INTENSITY_{time}$	::=	*Seconds* |*minutes* | *hours*
$INTENSITY_{value}$	::=	*kcal* | *km* | *step_count*
$FREQ$	::=	$TIMES$ **per day** | $TIMES$ **per week** | $TIMES$ **per month**
$TIMES$	::=	[1, . . . ; n] $\forall n \in N$

one_of $STRING\_SET$ construct to allow the Domain Expert to define as goal one mood among the ones listed in the set $STRING\_SET$, for instance in (Eq. 13.1):

$$G_a := m_g \textbf{ one\_of } [neutral, happy, very\ happy] \tag{13.1}$$

When a numerical range is used to describe the user mood, we use relational operators to specify a goal as a value in a subset of the given discrete range. Moreover, for both mood-based goals, the expert can optionally specify the frequency with which the user is asked to register their mood, through the ? $FREQ$ construct. The frequency can be expressed in terms of $TIMES$ *per day*, *per week*, or *per month*, where $TIMES$ belongs to a discrete range of values, as given in (Eq. 13.2):

$$G_a := m_g \geq 7 \textbf{ in } [1, \ \ldots, 10]\ 3 \textbf{ per day} \tag{13.2}$$

A mood-based goal m_g *succeeds* if it satisfies the relation expressed by the goal. In the presence of a frequency, instead, the user enters more than one mood. In this case, the mood-based goal succeeds if the average computed among the registered mood satisfies the relation expressed by the goal m_g; it *fails* otherwise.

A fitness-based goal specifies the required *intensity* and *frequency* with which users should perform fitness-related activities. In particular, the goal model provides two constructs to indicate the intensity, namely, $INTENSITY_{time}$ and $INTENSITY_{value}$. The former is used to express the intensity in terms of duration of the activity (e.g., *seconds*, *minutes*, and *hours*). The latter is used to express the intensity in non-time-based terms. Our goal model foresees the use of values such as *kcal*, *km*, and *step_count*. As for mood-based goals, the Domain Expert can optionally specify the frequency with which the user is asked to perform the suggested activities, via the ? $FREQ$ construct. Relational operators can be used to specify threshold values over intensity-based goals. Moreover, *control-flow constructs*, namely, **and**, **or**, and **one_of**, can also be specified to combine fitness-based goals. These constructs allow us to recursively combine elementary goals, of $INTENSITY_{time}$, $INTENSITY_{value}$, and threshold

types, thus to create goals of different complexity. An example is given in (Eq. 13.3):

$$G_a := f_g \geq 1000 \text{ kcal } 1 \textbf{ per day or } f_g > 5 \text{ km} \tag{13.3}$$

A fitness-based goal f_g of type intensity or threshold *succeeds* if the user performs the suggested activities with the required time-based or value-based intensity; it *fails* otherwise. Goals of type **and** and **or** represent combination of goals and they *succeed*, respectively *fail*, as per the rule defined by the involved logical operators. A goal **one_of** *seq* f_g specifies the need for achieving one of the goals in the given sequence. The choice of the goal to target among the available ones can depend on a utility function or a user's choice.

The presented goal model is open and easy to extend. If a new feature different from *mood* and *fitness* is envisaged, it is sufficient to extend the rule related to G_a with a further nonterminal term on the right-hand side of the rule, referring to the new feature, along with one or more associated rules. The ease of use of the goal model, as well as the *Editor of Abstract Activities & Goals* are designed as tools that allow *Domain Experts* to make changes in the tailoring of the app to better meet the interests and needs of the users. This is an important feature of the RA that allows it to better address social sustainability.

13.7 Methodology

As introduced in Sect. 13.2, to design our RA we used the framework and the methodology of Angelov et al. [15]. In Table 13.2 we list all questions for each dimension (i.e., context, goals, and design), with the answers we gave whilst designing our RA and the rationale for each answer.

In the **goal dimension**, the aim of our RA is providing guidelines for the design of personalized and self-adaptive e-Health apps. To the best of our knowledge no RA of this type exists in this domain (**G1**).

In the **context dimension**, our RA is devoted to any organization in the e-Health domain who can benefit from it (**C1**). Particularly, during the design of our RA we have used our collected experience from multiple collaborations with psychologists and e-Health app providers to formulate the requirements needed to be addressed. We were the sole designers of the RA (**C2**). The main objective was to design RA in a way that it can utilize, in the same architecture, relevant techniques needed to achieve both personalization and self-adaptation within this domain (**C3**).

In the **domain dimension**, the main ingredients of our RA are: software components and their connectors, the CluStream-GT algorithm, the MAPE-loops, and the goal model (**D1**). Specifically, the software components and goal model are semi-detailed as they demonstrate implementation feasibility and a clear objective but are not yet implemented. CluStream-GT is detailed as it is previously published and

Table 13.2 RA according to the three dimensions: context, goals, design

Dimension	Values	Rationale
G1: Why is it defined?	Facilitation	Our aim with this RA is to provide guidelines for the design of personalized and self-adaptive e-Health apps.
↓	↓	↓
C1: Where will it be used?	Multiple organizations	Multiple organizations within the e-Health domain.
C2: Who defines it?	Research centers (D),	The RA was designed by the authors who are all researchers.
	User organizations (R), software organizations (R)	Requirements for this RA were derived by collaborations with domain experts and e-Health app providers.
C3: When is it defined?	Preliminary	The algorithms, goal model, and MAPE-loops do not exist in practice yet.
↓	↓	↓
D1: What is described?	Components, algorithms, protocols, etc.	Components, CluStream-GT, MAPE-loops, domain model.
D2: How detailed is it described?	Semi-detailed architecture, detailed algorithms, and aggregated protocols	The goal model and the software components are semi-detailed, CluStream-GT is detailed, and the MAPE-loops are aggregated.
D3: How concrete is it described?	Abstract elements	At the time of design, our RA mainly abstracts from concrete technologies.
D4: How is it represented?	Semi-formal architecture representation and a formal algorithm	The RA is described according to 42010, CluStream-GT is implemented.

Table 13.3 Final match of our RA to one of the five types identified in [15]

	T/V	G1	C1	C2	C3	D1	D2	D3	D4
RA	5.1	X	X	X	X	X	X	X	X

tested work. The MAPE-loops only demonstrate the general communication and are specified at an aggregated level (**D2**). As our RA is described, we mainly abstract from concrete technologies (**D3**); in fact, the majority of the RA is currently presented in a semi-formal manner with the exception of CluStream-GT (**D4**).

In Table 13.3 we present our final match of the RA with respect to the types/variants (T/V) presented by Angelov et al. [15]. In particular, X denotes a match of the architecture values with those in the T/V. As shown, our RA fits one of the architecture variants identified and described by Angelov et al. (specifically variant 5.1); this demonstrates its congruence wrt its context, goals, and design. As stated in [15], if a RA can be classified into one of their identified types it has a better chance of being successful (i.e., "accepted by its stakeholders and used in multiple projects" [15]).

13.8 Viewpoint Definition

This section describes the essential elements of the viewpoint defined to represent Mobile-enabled Self-adaptive Personalized Systems (or MSaPS Viewpoint for short).

We have used it to create the view of our RA for personalized and self-adaptive e-Health Apps as described in Fig. 13.2. It must be noted, however, that the MSaPS Viewpoint is not limited to reference architecture use: one could use it to design specific e-Health mobile-enabled systems, as well as to describe mobile-enabled systems not targeted at e-Health but involving personalization and self-adaptation.

The MSaPS Viewpoint relies on the guidelines provided in the ISO/IEC/IEEE 42010 Standard [35]. Accordingly, after a short description it frames (cf. Table 13.4) the typical stakeholders, their concerns, the meta-model, and the related conforming visual notation. The indication of which stakeholders may have which concerns is further shown in Table 13.5.

13.9 Scenario-Based Evaluation

To evaluate how our RA would cover typical usage scenarios, we used the domain expertise learnt from our industrial collaborations and have defined the example case and associated scenarios described in this section (see Figs. 13.8 and 13.9). For each scenario, we challenged how the RA can be used. Throughout the example we use a hypothetical user named Connor and focus on fitness-based goals.

Scenario 1 (Fig. 13.8a). Connor downloads a fitness app that uses our proposed RA. As a first step, he has to input some preferences about the kind of activities he likes the most, complete a questionnaire used to understand his fitness level and give consent for his data to be tracked and used by the app. The fitness app decides on his first weekly schedule of activities.

This is a default schedule created by the Domain Expert, in accordance with the information provided by Connor. The default schedule, represented as an Abstract Activity, is adapted by the User Driven Adaptation Manager in accordance with Connor's preferences and supported third-party applications. *This scenario highlights how our RA supports both user level adaptation (where the Abstract Activities assigned to Connor are adapted by the User Driven Adaptation Manager) and architecture level adaptation (where the Third-party Applications Manager realizes the Concrete Activities by dynamically integrating the specific apps Connor uses on his mobile device).*

Scenario 2 (Fig. 13.8b). During the first week Connor performs the Concrete Activities assigned to him. This first week is needed by the app to gather enough data from Connor so that the AI Personalization can determine to which macro-cluster Connor belongs. After successfully clustering Connor, the AI Personalization sends an update to the User Process Handler, which is now able to send the appropriate

Table 13.4 Elements of the MSaPS viewpoint

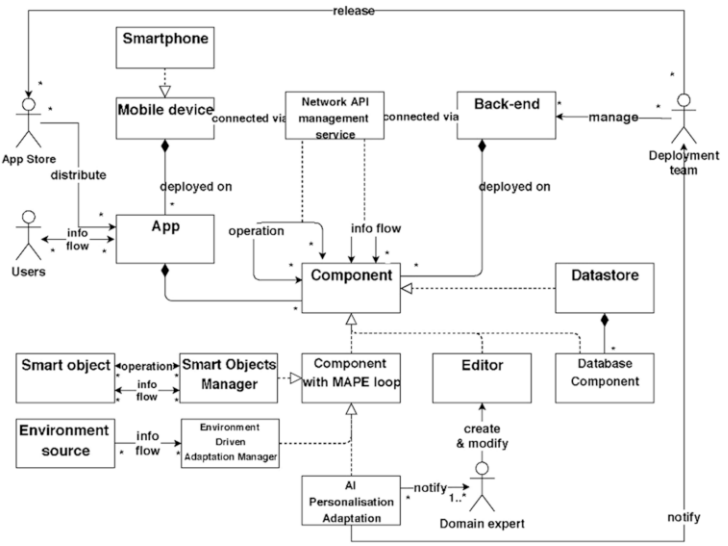
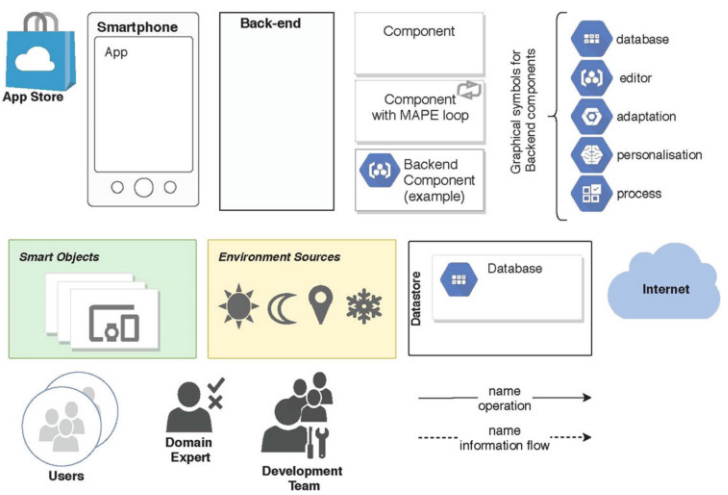
Element	Description
Viewpoint description	This viewpoint captures the essential architectural and contextual elements supporting the design of mobile-enabled self-adaptive and personalized systems
Typical stakeholders	Domain experts, software architects, members development teams, user
Concerns	C1: How to extend a mobile app with personalization and self-adaptation? C2: How to integrate external smart objects and environmental information flows? C3: How to integrate Domain Expert knowledge into the mobile app's personalization? C4: How to integrate third-party apps as part of the mobile app's personalization? C5: What are the components of MAPE loops and how do they interact? C6: Where is the user data stored?
Meta-model	
Conforming notation	

Table 13.5 Stakeholders and related concerns

Concerns/stakeholders	User	Domain expert	Developer	Software architect
C1: Extend App w/Pers/Adapt			✓	✓
C2: Integrate External Elements			✓	✓
C3: Integrate		✓	✓	✓
C4: Integrate Apps			✓	✓
C5: MAPE Interactions			✓	
C6: User Data	✓			

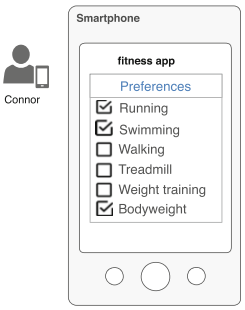
User Process to Connor. By querying the Third-party Applications Manager, the Abstract Activity is adapted by the User Driven Adaptation Manager into appropriate Concrete Activities. Like with the default schedule, the two *Cardio* entries are converted into running, whilst the newly given *Strength* one is converted into weight lifting. Furthermore, the new goal he receives is more challenging. *This scenario illustrates the same levels of adaptation as scenario 1, completed by the same components. Additionally, the user level adaptation is further personalized by clustering Connor and the User Process Handler sending him his cluster-related User Process.*

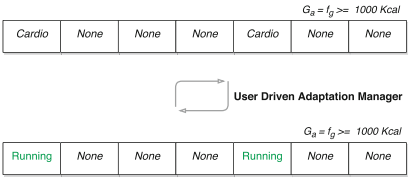
Scenario 3 (Fig. 13.8c). On Monday Connor goes running as suggested by the app. Whilst he is running outdoors, both the Wi-Fi and 4G have no connection. The Internet Connectivity Manager detects this and so decides to store the data locally. When Connor gets back home after completing his run, the Wi-Fi connection is reestablished. Aware of this, the Internet Connectivity Manager sends the locally stored Collected Data to the back-end. *This scenario illustrates an architectural level adaptation—performed by the Internet Connectivity Manager by storing the data locally and sending it to the back-end when the Internet connection is reestablished.*

Scenario 4 (Fig. 13.9a). On Wednesday as Connor is in the gym doing the assigned weight training, the connection with the smartwatch is interrupted. The disconnection is detected by the Smart Objects Manager that at runtime reconnects to the smartwatch allowing the app to resume collecting the data about Connor via the smart object. *This scenario describes an example of architectural level adaptation—performed by the Smart Objects Manager when Connor's smartwatch is no longer detected by the app.*

Scenario 5 (Fig. 13.9b). On Friday, the Environment Driven Adaptation Manager detects that the weather forecast predicts rain for the day. As Connor's scheduled Concrete Activity is running, an outdoor activity, the Environment Driven Adaptation Manager needs to make a runtime adaptation. It queries the Third-party Applications Manager for Cardio activities suitable for indoors. As swimming is the best alternative, it switches running with swimming and notifies Connor of the change, saying that the activity will be carried out via the *swim.com* app. *This scenario focuses on both user level adaptation (when the Concrete Activity is adapted by the Environment Driven Adaptation Manager), and architectural level adaptation (when the Third-party Applications Manager accesses the third-party app).*

(a) Scenario 1: Connor installs the application

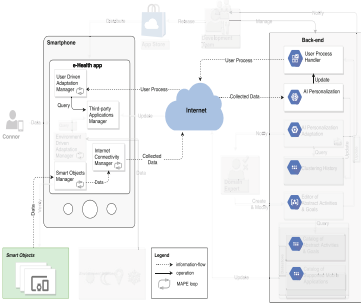


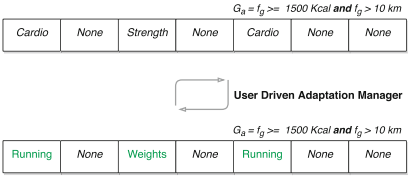


Scenario 1 steps:

1. Connor downloads and installs the e-Health app on his smartphone. He then sets his preferred types of exercise, which the phone stores as selectable Concrete Activities.
2. From the inputted preferences the app selects the appropriate Abstract Activity.
3. The User Driven Adaptation Manager converts the Abstract Activity into Concrete Activities and notifies Connor.

(b) Scenario 2: Connor gets assigned to a macro-cluster and sent the respective User Process.

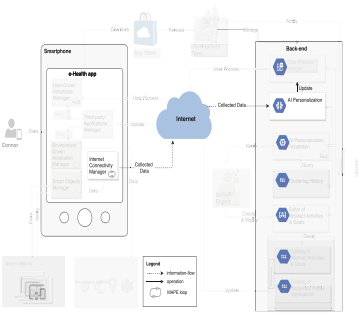


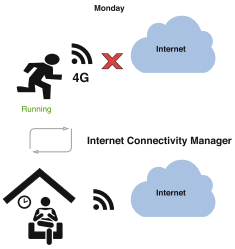


Scenario 2 steps:

1. Connor completed the default Abstract Activity. With the resulting collected data, he gets assigned to a macro-cluster.
2. The User Process Handler sends to Connor his assigned macro-cluster User Process.
3. The User Process is received and processed by the User Driven Adaptation Manager, which then notifies Connor of the new Concrete Activities.

(c) Scenario 3: Connor has no WiFi or 4G connection so the Internet Connectivity Manager saves the data to send at a later point.



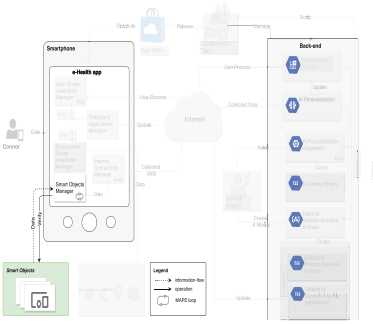


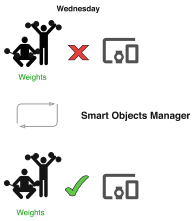
Scenario 3 steps:

1. Connor is starting to run so to complete his first given Concrete Activity.
2. The Internet Connectivity Manager detects that the WiFi and 4G is not connecting to the Internet.
3. The Internet Connectivity Manager saves the `Collected Data` locally during the run. It then sends the `Collected Data` as Connor gets back home and has WiFi connection.

Fig. 13.8 Scenarios 1–3

(a) Scenario 4: The e-Health app loses connection to Connor's smart-watch. The Smart Objects Manager re-establishes the connection.

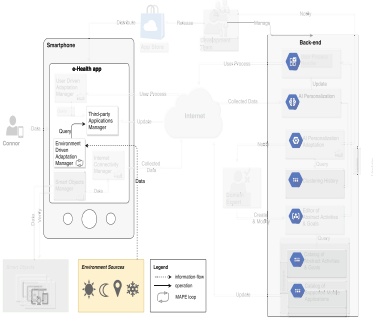


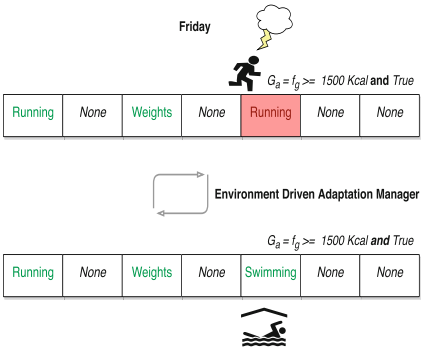


Scenario 4 steps:

1. The e-Health app looses connection to Connor's smart-watch.
2. The Smart Objects Manager detects the changed state of the connection to the smart-watch.
3. The Smart Objects Manager autonomously re-establishes connection to Connor's smart-watch, allowing the e-Health app to resume collecting the data via that smart object.

(b) Scenario 5: The Environment Driven Adaptation Manager detects a change of weather forecast and so adapts the Concrete Activity





Scenario 5 steps:

1. The Environment Driven Adaptation Manager detects that the weather forecast predicts rain.
2. During the time in which it is forecasted to rain Connor has scheduled Running, an outdoors Concrete Activity.
3. Given this clash the Environment Driven Adaptation Manager adapts Running to Swimming, since it is an indoor activity and doesn't clash with the rain forecast.

Fig. 13.9 Scenarios 4–6

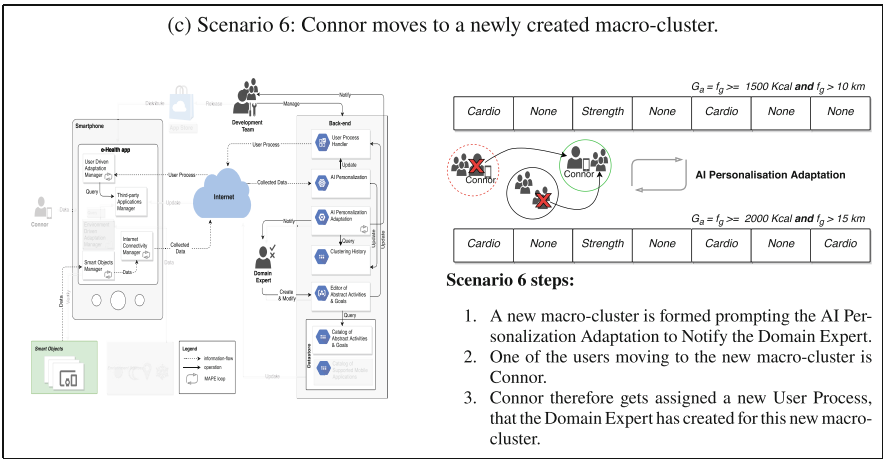


Fig. 13.9 (continued)

Scenario 6 (Fig. 13.9c). Connor has now finished his second week and has successfully reached his assigned goal. In order to maintain the goal engaging and challenging, Connor's success, along with the success of other users, causes the AI Personalization to create a new macro-cluster for them. As the new macro-cluster is one that has never occurred in the system's history, the AI Personalization Adaptation deems this change significant and so `notifies` the Domain Expert to analyze the new macro-cluster and associate with it a new User Process. The notified Domain Expert makes the new User Process via the web-based Editor of Abstract Activities & Goals. Given the users of the new macro-cluster's success (including Connor), the Domain Expert makes the User Process goal more challenging, increasing the amount of k*cal* to 2000 and the k*m* to 15 (as shown in the figure). This new User Process is sent to the members of the new macro-cluster via the User Process Handler. *This scenario illustrates all three levels of adaptation: the cluster level adaptation to the new macro-cluster performed by the AI Personalization Adaptation, the user level adaptation performed by the User Driven Adaptation Manager when adapting a new User Process, and the architectural level adaptation by associating third-party apps to Concrete Activities performed by the Third-party Applications Manager.*

13.10 Discussion

It is important to note that our RA is *extensible enough to support other domains* beyond fitness and mood. Specifically, the goal model has been designed such that supporting an additional domain can be achieved by adding (1) a new nonterminal

term in the root rule G_a and (2) one or more rules describing the goal within the new domain. Also, many of the existing rules (e.g., *F R E Q*) are generic enough to be reused by newly added rules. On the client side no changes are required, whereas the only components which may need to be customized to a new application domain are: (1) the Editor of Abstract Activities & Goals, so that it is tailored to the different domain experts and the extended goal model, and (2) the Catalog of Supported Mobile Applications, so that it now describes the interaction points with different third-party apps.

Abstract Activities allow Domain Experts to define *incremental goals* spanning over the duration of the whole User Process. In addition, User Processes are defined at the cluster level (potentially including thousands of users) and can cover large time spans (e.g., weeks or months). Those features make the operation of the RA sustainable from the perspective of Domain Experts, who are not required to frequently intervene for defining new goals or User Processes. Furthermore, these features make the apps adopting our RA socially sustainable on multiple levels. The cluster-level-defined User Processes allow for tailoring to a "community" of similar users, empowering them to achieve a better life. On an individual level, the app fine-tunes the User Processes to better suit the user's needs and interests; this allows the individual user to better achieve their goals both in the immediate and in the systemic (as defined in [36]). Lastly, these features allow for the larger group of users utilizing this RA to reach the same level of health benefits, as the interventions have been specifically tailored for them for this goal.

Through the conversion from Activity Categories to Concrete Activities, which takes place during the dynamic Abstract Activities refinement, we accommodate both *Type-to-Type* adaptation (e.g., from the *Cardio* Activity Category to the *Running* Concrete Activity) and the most common *Type-to-Instance* adaptation (e.g., by using the Strava mobile app as an instance of the *Running* Concrete Activity). Similarly, a *Type-to-Type* adaptation is reported by Calinescu et al. [37] presenting an approach where elements are replaced with other elements providing the *same* functionality but showing a superior quality to deal with changing conditions (e.g., dynamic replacement of service instances in service-based systems). In our approach, however, we go beyond, by replacing activities with others providing *different* functionality to deal with changing conditions. To the best of our knowledge, this adaptation type is uncommon in self-adaptive architectures, despite quite helpful.

The components of the RA running on the smartphone can be deployed in two different ways, each leading to a different business case. Firstly, those components can be integrated into an existing e-Health app (e.g., Endomondo[3] for sports tracking) so as to provide personalization and self-adaptation capabilities to its services. In this case the development team of the app just needs to deploy the client-side components of the RA as a third-party library, suitably integrate the original app with the added library, and launch the server-side components. The

[3] https://www.endomondo.com/

second business case regards the creation of a new meta-app integrating the services of third-party apps, similarly to what apps like IFTTT[4] do. In this case, the meta-app makes an extensive usage of the Third-party Applications Manager component and orchestrates the execution of the other apps already installed on the user device.

Finally, we are aware that our RA is responsible for managing highly sensitive user data, which may raise severe *privacy* concerns. In order to mitigate potential privacy threats, the communication between the mobile app and the back-end is TLS-encrypted and the payload of push notifications is encrypted as well, e.g., by using the Capillary Project [38] for Android apps, which supports state-of-the-art encryption algorithms, such as RSA and Web Push encryption. Eventually, we highlight that, according to the privacy level required by the Development Team, the components running in the back-end can be deployed either on premises or on the Cloud, e.g., by building on public Cloud services like Amazon AWS and executing them behind additional authentication and authorization layers.

13.11 Conclusions and Future Work

In this paper we presented a RA for e-Health mobile apps. Its goal is to combine AI-based personalization and self-adaptation. The RA achieves self-adaptation on three levels: (1) adaptation to the users and their environment, (2) adaptation to smart objects and third-party applications, and (3) adaptation according to the data of the AI-based personalization, ensuring that users receive personalized activities that evolve with the users' runtime changes in behavior. This work emphasizes how personalization and self-adaptation within the e-Health domain can be beneficial in addressing social sustainability. By tailoring user interventions we empower mobile app developers to better help their users in achieving better physical and mental health; this leads to increased support for the community of people who suffer from mental and physical illness and are working on increasing their health. The RA therefore achieves what is defined as the core principal of social sustainability in the realm of software-intensive systems. As future work we are realizing a prototype implementing the RA and designing a controlled experiment to evaluate its effects on user behavior and performance at runtime.

References

1. Williams PAH, McCauley V (2013) A rapidly moving target: conformance with e-health standards for mobile computing. In: 2nd Australian eHealth Informatics and Security Conference

[4] https://ifttt.com/

2. Global Industry Analysts, I (2019) mhealth (mobile health) services – market analysis, trends, and forecasts. https://tinyurl.com/rbvdtc3
3. Paschou M, Sakkopoulos E, Sourla E, Tsakalidis A (2013) Health internet of things: metrics and methods for efficient data transfer. Simul Model Pract Theory 34:186–199
4. Fling B (2009) Mobile design and development: Practical concepts and techniques for creating mobile sites and Web apps. O'Reilly Media, Inc.
5. Fan H, Poole MS (2006) What is personalization? Perspectives on the design and implementation of personalization in information systems. J Organ Comput Electron Comm 16 (3–4):179–202
6. Grua EM, Malavolta I, Lago P (2019) Self-adaptation in mobile apps: a systematic literature study. In: IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS). pp 51–62
7. Weyns D (2017) Software engineering of self-adaptive systems: an organised tour and future challenges. In: Handbook of Software Engineering
8. Yang Z, Li Z, Jin Z, Chen Y (2014) A systematic literature review of requirements modeling and analysis for self-adaptive systems. In: International Working Conference on Requirements Engineering: Foundation for Software Quality. Springer, pp 55–71
9. IBM (2006) An architectural blueprint for autonomic computing. Technical report. IBM
10. Grua EM, Hoogendoorn M, Malavolta I, Lago P, Eiben A (2019) Clustream-GT: Online clustering for personalization in the health domain. In: IEEE/WIC/ACM International Conference on Web Intelligence. ACM, pp 270–275
11. Grua EM, De Sanctis M, Lago P (2020) A reference architecture for personalized and self-adaptive e-health apps. In: Software Architecture: 14th European Conference, ECSA 2020 Tracks and Workshops, L'Aquila, Italy, 14–18 September 2020, Proceedings. Springer, pp 195–209
12. Volpato T, Oliveira BRN, Garcés L, Capilla R, Nakagawa EY (2017) Two perspectives on reference architecture sustainability. In: Proceedings of the 11th European Conference on Software Architecture: Companion. ACM, pp 188–194
13. Kim KJ, Ahn H (2004) Using a clustering genetic algorithm to support customer segmentation for personalized recommender systems. In: International Conference on AI, Simulation, and Planning in High Autonomy Systems. Springer, pp 409–415
14. Grua EM, Hoogendoorn M (2018) Exploring clustering techniques for effective reinforcement learning based personalization for health and wellbeing. In: 2018 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE, pp 813–820
15. Angelov S, Grefen P, Greefhorst D (2012) A framework for analysis and design of software reference architectures. Inf Softw Technol 54(4)
16. Lago P, Verdecchia R, Fernandez NC, Rahmadian E, Sturm J, van Nijnanten T, Bosma R, Debuysscher C, Ricardo P (2020) Designing for sustainability: lessons learned from four industrial projects. In: Environmental Informatics – Sustainability aware digital twins for urban smart environments (EnviroInfo). Springer
17. (2019) The industrial internet of things volume G1: reference architecture. Industrial Internet Consortium. https://bit.ly/2talimM
18. Bassi A, Bauer M, Fiedler M, Kramp T, van Kranenburg R, Lange S, Meissner S (2016) Enabling things to talk: designing IoT solutions with the IoT architectural reference model, 1st edn. Springer
19. Bauer M et al (2013) IoT reference architecture. In: enabling things to talk: designing IoT solutions with the IoT architectural reference model
20. Fremantle P (2015) A reference architecture for the internet of things. WSO2 White paper. https://bit.ly/2RMzCft
21. Feljan AV, Mohalik SK, Jayaraman MB, Badrinath R (2015) SOA-PE: a service-oriented architecture for planning and execution in cyber-physical systems. In: 2015 International Conference on Smart Sensors and Systems (IC-SSS). pp 1–6

22. Mohalik SK, Narendra NC, Badrinath R, Le D (2017) Adaptive service-oriented architectures for cyber physical systems. In: IEEE Symposium on Service-Oriented System Engineering, SOSE. pp 57–62
23. de Morais Barroca Filho I, Junior GSA, Batista TV (2019) Extending and instantiating a software reference architecture for iot-based healthcare applications. In: Int. Conf. on Computational Science and Its Applications. pp 203–218
24. Mizouni R, Matar MA, Al Mahmoud Z, Alzahmi S, Salah A (2014) A framework for context-aware self-adaptive mobile applications SPL. Expert Syst Applic 41(16):7549–7564
25. Lopez FS, Condori-Fernández N (2016) Design of an adaptive persuasive mobile application for stimulating the medication adherence. In: International Conference on Intelligent Technologies for Interactive Entertainment. Springer, pp 99–105
26. Kim HK (2013) Architecture for adaptive mobile applications. Int J Bio-Sci Bio-Technol 5 (5):197–210
27. Bucchiarone A, Lluch-Lafuente A, Marconi A, Pistore M (2009) A formalisation of adaptable pervasive flows. In: WS-FM. pp 61–75
28. Shiffman S, Stone AA, Hufford MR (2008) Ecological momentary assessment. Annu Rev Clin Psychol 4:1–32
29. Gil M, Pelechano V, Fons J, Albert M (2016) Designing the human in the loop of self-adaptive systems. In: International Conference on Ubiquitous Computing and Ambient Intelligence. Springer, pp 437–449
30. Dal Lago U, Pistore M, Traverso P (2002) Planning with a language for extended goals. In: Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence. pp 447–454
31. Mylopoulos J, Chung L, Nixon BA (1992) Representing and using nonfunctional requirements: a process-oriented approach. IEEE Trans Softw Eng 18(6):483–497
32. Santos M, Gralha C, Goulão M, Araújo J (2018) Increasing the semantic transparency of the KAOS goal model concrete syntax. In: Conceptual Modeling – 37th International Conference, ER. pp 424–439
33. Morandini M, Penserini L, Perini A (2008) Towards goal-oriented development of self-adaptive systems. In: 2008 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems, SEAMS. pp 9–16
34. Qian W, Peng X, Wang H, Mylopoulos J, Zheng J, Zhao W (2018) MobiGoal: flexible achievement of personal goals for mobile users. IEEE Trans Serv Comput 11(2):384–398
35. International Organization for Standardization (2011) ISO/IEC/IEEE 42010:2011 – Systems and Software Engineering – Architecture Description. Technical report. International Organization for Standardization (ISO)
36. Lago P (2019) Architecture design decision maps for software sustainability. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS). IEEE, pp 61–64
37. Calinescu R, Weyns D, Gerasimou S, Iftikhar MU, Habli I, Kelly T (2018) Engineering trustworthy self-adaptive software with dynamic assurance cases. IEEE Trans Softw Eng 44 (11):1039–1069
38. Hogben G, Perera M (2018) Project capillary: end-to-end encryption for push messaging, simplified. https://android-developers.googleblog.com/2018/06/project-capillary-end-to-end-encryption.html?m=1