A Reference Architecture for Personalized and Self-adaptive e-Health Apps

Eoin Martino Grua[1 [0000−0002−5471−4338]], Martina De Sanctis[2 [0000−0002−9417−660X]], and Patricia Lago[1 [0000−0002−2234−0845]]

[1] Vrije Universiteit Amsterdam, The Netherlands, {e.m.grua | p.lago}@vu.nl
[2] Gran Sasso Science Institute, L'Aquila, Italy, martina.desanctis@gssi.it

Abstract. A wealth of e-Health mobile apps are available for many purposes, such as life style improvement, mental coaching, etc. 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., by using fixed rule sets or being not sufficiently tailored towards individuals. 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 paper presents a reference architecture for enabling AI-based personalization and self-adaptation of mobile apps for e-Health. The reference architecture makes use of multiple MAPE loops operating at different levels of granularity and for different purposes.

Keywords: Self-adaptive systems · Personalization · Reference Architecture · Mobile apps · e-Health.

1 Introduction

E-Health mobile apps are designed for assisting end users in tracking and improving their own health-related activities [28]. With a projected market growth to US$102.3 Billion by 2023, e-Health apps represent a significant market [12] providing a wide spectrum of services, i.e., life style improvement, mental coaching, sport tracking, recording of medical data [24]. The unique characteristics of e-Health apps w.r.t. other health-related software systems are that e-Health apps (i) can take advantage of smartphone sensors, (ii) can reach an extremely wide audience with low infrastructural investments, and (iii) 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 [9].

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* [7]

and *self-adaptation* [15,27] as effective instruments for getting users continuously engaged and active, eventually leading to better physical and mental conditions.

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 that combines data-driven personalization with self-adaptation*. The main design drivers that make the proposed reference architecture unique are: (i) the **combination of multiple *Monitor - Analyze - Plan - Execute* (MAPE) loops** [17] 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); (ii) 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 [14], showing promising results by doing better than the current state-of-the-art.

The main characteristics of the proposed reference architecture are the following: (i) it caters the personalization of provided services to the specific user preferences (e.g., preferred sport activities); (ii) it guarantees the correct functioning of the provided features via the use of connected IoT devices (e.g., a smart-bracelet) and runtime adaptation strategies; (iii) it adapts the provided services depending on contextual factors such as environmental conditions and weather; (iv) it supports a smooth participation of domain experts (e.g., psychologists) in the personalization and self-adaptation processes; and (v) 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).

2 Background

The notion of *reference architecture* (RA) is borrowed from Volpato *et al.* [26], 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 [27].

For the definition of *personalization* we build on that by Fan and Poole [7] 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 (standing for: CluStream for Growing Time-series) [14]. 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 *of-*

fline phase initializes the algorithm with a small initial dataset; this is done either at design time or at the start of runtime. After, 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 [14, 19]. An example case where clustering can be used to aid personalization is with the use of cluster based Reinforcement Learning [13].

The *methodology* used for the design of our RA is the one presented by Angelov *et al.* [2].

3 Related Work

Several RAs for IoT can be found in the literature [1,3,4,10]. In particular, Bauer *et al.* [4] 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 [3] aims to be easily customized to different needs, and it makes use of *axioms* and *relationships* to define connections among IoT entities. IIRA [1] is particularly tailored for industrial IoT systems. WSO2 [10] 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.

Other works providing service oriented architectures (SOAs) focused on adaptation but neglected user-based personalization. E.g., Feljan *et al.* [8] defined a 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. Differently from RAH, our RA explicitly manages the self-adaptation of the e-Health mobile app, both at users- and architectural levels. Mizouni *et al.* [21] 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 [20] 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 [18] proposes a general RA that can be used when developing adaptive apps and implements a e-Health app as an example. However, being it 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 the 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 (i) *personalization* for the different users, by exploiting the users' smart objects and preferences to dynamically get data about e.g., their mood and daily activities, and (ii) *runtime adaptation* to the user-needs and context in order to keep them engaged and active.

4 Reference Architecture

Fig. 1 shows our RA[3] with the following stakeholders and components.

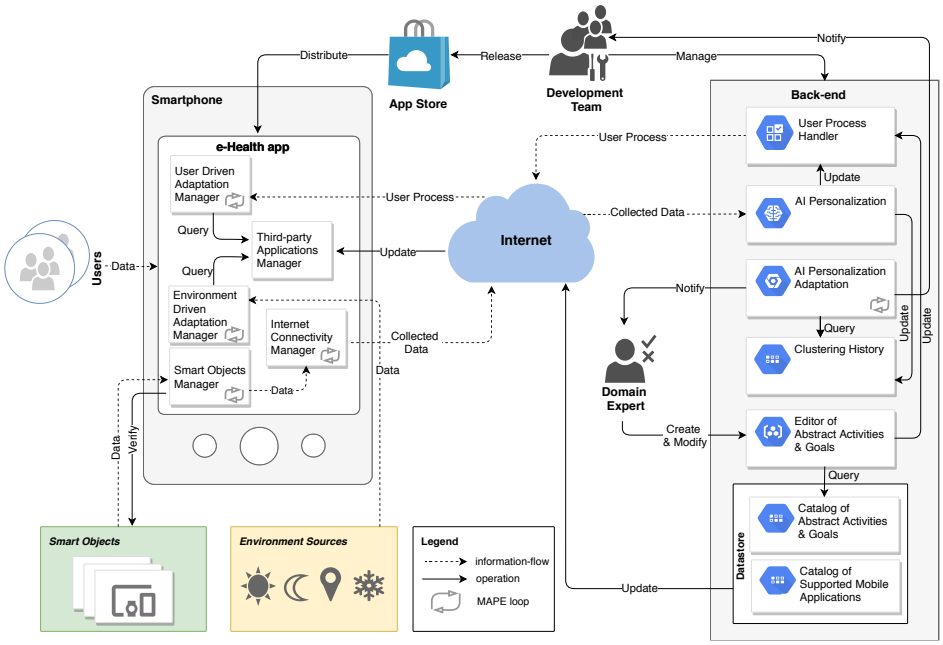

Fig. 1: Reference architecture for Personalized and Self-adaptive e-Health Apps

Users provide and generate the `Data` gathered by the e-Health app. At the first installation, the users are asked to input information to better understand their aptitudes. After an initial usage phase and data collection, the system has enough information to assign them to a cluster.

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* (*UD Adaptation Manager* and *ED*

[3] For the interested reader, we have defined the corresponding viewpoint here: http://s2group.cs.vu.nl/casa-2020-technical-report/

Adaptation Manager from here on, respectively), *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 the 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 the user's preferences. Further details on these components are given in Sec. 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 smart-watch would be used by the app to track the user's heart-rate, 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 w.r.t. its current operational context and to the user's scheduled activities (see Sec. 5.5).

The back-end of our RA (right-hand side of Fig. 1) 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, by taking 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 [5], although they differ both in the structure and in the way they are refined, as later explained. 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[4]. For the sake of space we leave the description of the formalization of the goal model to future work.

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 run-time via the use of the *UD 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 Sec. 5.2).

[4] Examples of Abstract Activities are shown here: http://s2group.cs.vu.nl/casa-2020-technical-report/

The *User Process Handler* receives `Updates` from (i) the *AI Personalization* and (ii) 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 [14]. It receives the input data from the e-Health app (see `Collected Data` in Fig. 1). More than one instance of CluStream-GT can be running at the same time. In fact, there is one instance per category of data. E.g., if the e-Health app is recording both ecological momentary assessment [25] 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 Sec. 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.

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[5] for running and cycling, Swim.com[6] 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: (i) we do not want to disrupt the users' habits and preferences in terms of apps used for tracking their activities, (ii) we want to *build on* existing large user bases, (iii) we do not want to reinvent the wheel by re-implementing 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* responsibility is to keep the list of supported mobile apps up to date and provide the corresponding metadata to the *UD Adaptation Manager* and the *ED 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.

5 Components supporting Self-adaptation

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

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. 2. 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 Personalisation Adaptation periodically queries the Clustering History database. It compares the current clusters with the previously saved ones. If any

[5] http://strava.com [6] http://swim.com

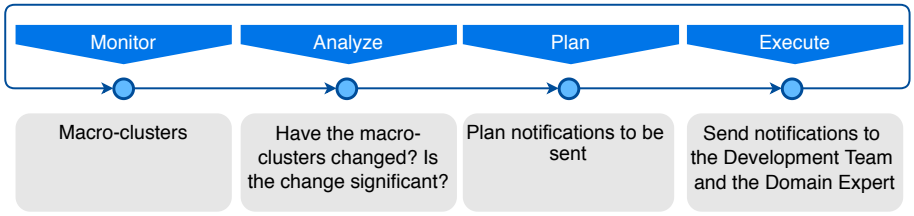

Fig. 2: AI Personalization Adaptation MAPE loop.

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 of 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 [11], which considers the involvement of humans in MAPE loops. In particular, in [11] 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. 1) 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 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.

5.2 User Driven Adaptation Manager

The main responsibility of the UD 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:

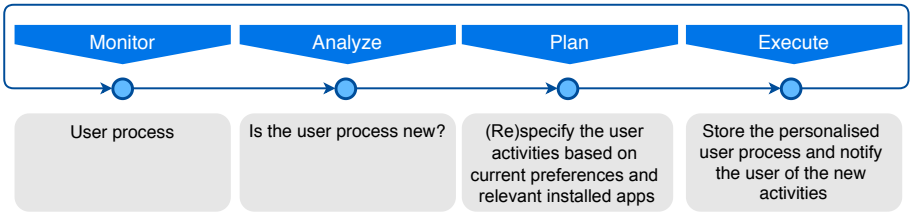

Fig. 3: UD Adaptation Manager MAPE loop.

• **Selectable**: is `True` if the UD Adaptation Manager or the ED 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**: it specifies if the activity is performed indoors or outdoors. This attribute is used by the ED Adaptation Manager to choose the appropriate Concrete Activity according to weather conditions (see Sect. 5.5).

• **Activity category**: it defines what type of category does the Concrete Activity fall under. E.g., for a fitness activity, it specifies a cardio or strength training.

• **Recurrence**: it tracks how many times the user has performed the Concrete Activity in the past. It allows the UD 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 UD 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 UD 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 the 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 run-time the personalized user schedule, in a transparent way and without a direct user involvement. Fig. 3 depicts the MAPE loop of the UD Adaptation Manager. Once it accomplishes its main task of refining the User Process, the UD Adaptation Manager enters the `Monitor` phase of its MAPE loop, by monitoring the 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 UD 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, stor-

ing this newly created personalized User Process and notifying the user about the new activity schedule.

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, thus to perform optimally. To this aim, it implements a MAPE loop, shown in Fig. 4, supporting the dynamic adaptation at the architectural level of the smart objects. The `Monitor` phase is devoted to the

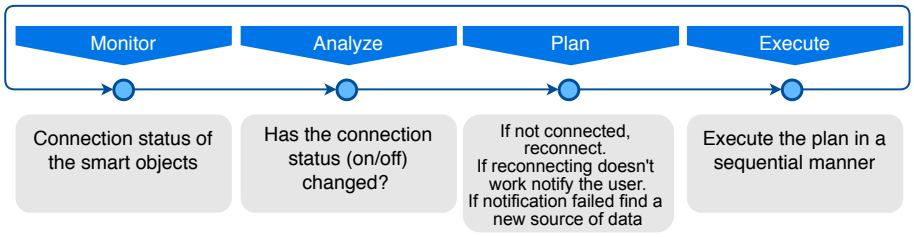

Fig. 4: Smart Objects Manager MAPE loop.

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 re-establishing the lost connection or at finding a new source of data. For instance, if the smart-watch connected to the smartphone runs out of battery and the attempts to reconnect to it fail, the Smart Objects Manager will switch to sensors inbuilt in the smartphone (such as the accelerometer).

5.4 Internet Connectivity Manager

The main purposes of the Internet Connectivity Manager are to (1) send the `Collected Data` to the back-end and store them locally when the connection is missing, and (2) provide resilience to the e-Health app's internet connectivity. As shown in the MAPE loop in Fig. 5, 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

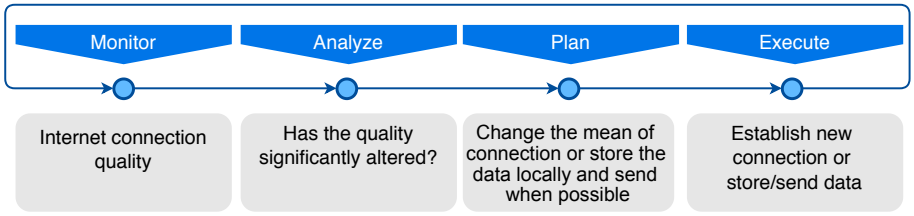

Fig. 5: Internet Connectivity Manager MAPE loop.

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.

5.5 Environment Driven Adaptation Manager

One of the objectives of the e-Health app is keeping the users constantly engaged, to ensure that they execute their planned schedule of activities. To this aim, the ED Adaptation Manager plays an important role, which is essentially supported by its MAPE loop, depicted in Fig. 6. The purpose of this component is to con-

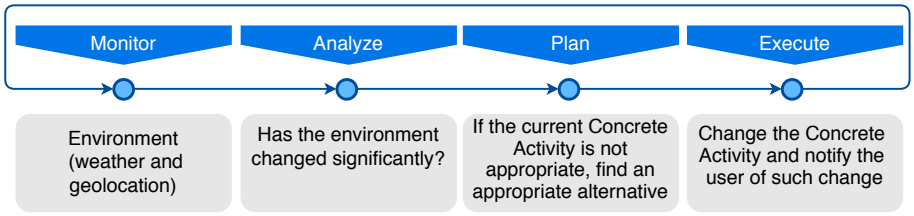

Fig. 6: ED Adaptation Manager MAPE loop.

stantly check whether the currently scheduled Concrete Activity best matches the runtime environment (i.e., weather conditions) the user is located in. To do so, the ED Adaptation Manager monitors in run-time 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.

6 Discussion

It is important to note that our RA is *extensible so to support other domains* beyond fitness and mood. On the client side no changes are required, whereas the only components which may need to be customized to a new application domain are: (i) the Editor of Abstract Activities & Goals, so that it is tailored to the different domain experts; and (ii) 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, 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.* [6] 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[7] for sports tracking) so 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 second business case regards the creation of a new meta-app integrating the services of third-party apps, similarly to what apps like IFTTT[8] 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 [16] for Android apps, which supports state-of-the-art

[7] http://endomondo.com [8] http://ifttt.com

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 in the Cloud, e.g., by building on public Cloud services like Amazon AWS and execute them in a protected environment, e.g., behind additional authentication and authorization layers.

7 Conclusions and Future Work

In this paper we presented a RA for e-Health apps. Its goal is to combine AI-based personalization and self-adaptation. The RA achieves self-adaptation on three levels: (i) adaptation to the users and their environment, (ii) adaptation to smart objects and third-party applications, and (iii) adaptation according to the data of the AI-based personalization, ensuring that users receive personalized activities that evolve with the users' run-time changes in behavior.

As future work we are realizing a prototype implementing the RA and designing a controlled experiment to evaluate its effects on users' behavior and performance at run-time.

References

1. The industrial internet of things volume G1: reference architecture. Industrial Internet Consortium (2019), `https://bit.ly/2talimM`
2. Angelov, S., Grefen, P., Greefhorst, D.: A framework for analysis and design of software reference architectures. Information and Software Technology **54**(4) (2012)
3. Bassi, A., Bauer, M., Fiedler, M., Kramp, T., van Kranenburg, R., Lange, S., Meissner, S.: Enabling Things to Talk: Designing IoT Solutions with the IoT Architectural Reference Model. Springer Publishing Company, 1st edn. (2016)
4. Bauer, M., et. al: IoT Reference Architecture. In: Enabling Things to Talk: Designing IoT solutions with the IoT Architectural Reference Model (2013)
5. Bucchiarone, A., Lluch-Lafuente, A., Marconi, A., Pistore, M.: A formalisation of adaptable pervasive flows. In: WS-FM. pp. 61–75 (2009)
6. Calinescu, R., Weyns, D., Gerasimou, S., Iftikhar, M.U., Habli, I., Kelly, T.: Engineering trustworthy self-adaptive software with dynamic assurance cases. IEEE Trans. Software Eng. **44**(11), 1039–1069 (2018)
7. Fan, H., Poole, M.S.: What is personalization? perspectives on the design and implementation of personalization in information systems. Journal of Organizational Computing and Electronic Commerce **16**(3-4), 179–202 (2006)
8. Feljan, A.V., Mohalik, S.K., Jayaraman, M.B., Badrinath, R.: 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 (2015)
9. Fling, B.: Mobile design and development: Practical concepts and techniques for creating mobile sites and Web apps. O'Reilly Media, Inc. (2009)
10. Fremantle, P.: A Reference Architecture for the Internet of Things. WSO2 White paper (2015), `https://bit.ly/2RMzCft`
11. Gil, M., Pelechano, V., Fons, J., Albert, M.: Designing the human in the loop of self-adaptive systems. In: International Conference on Ubiquitous Computing and Ambient Intelligence. pp. 437–449. Springer (2016)

12. Global Industry Analysts, I.: mhealth (mobile health) services - market analysis, trends, and forecasts (2019), https://tinyurl.com/rbvdtc3
13. Grua, E.M., Hoogendoorn, M.: Exploring clustering techniques for effective reinforcement learning based personalization for health and wellbeing. In: 2018 IEEE Symposium Series on Computational Intelligence (SSCI). pp. 813–820. IEEE (2018)
14. Grua, E.M., Hoogendoorn, M., Malavolta, I., Lago, P., Eiben, A.: Clustream-GT: Online clustering for personalization in the health domain. In: IEEE/WIC/ACM International Conference on Web Intelligence. pp. 270–275. ACM (2019)
15. Grua, E.M., Malavolta, I., Lago, P.: 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 (2019)
16. Hogben, G., Perera, M.: Project capillary: End-to-end encryption for push messaging, simplified. (2018), https://tinyurl.com/y8n8btoc
17. IBM: An architectural blueprint for autonomic computing. Tech. rep., IBM (2006)
18. Kim, H.K.: Architecture for adaptive mobile applications. Int. J. Bio-Sci. Bio-Technol **5**(5), 197–210 (2013)
19. Kim, K.j., Ahn, H.: 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. pp. 409–415. Springer (2004)
20. Lopez, F.S., Condori-Fernández, N.: Design of an adaptive persuasive mobile application for stimulating the medication adherence. In: International Conference on Intelligent Technologies for Interactive Entertainment. pp. 99–105. Springer (2016)
21. Mizouni, R., Matar, M.A., Al Mahmoud, Z., Alzahmi, S., Salah, A.: A framework for context-aware self-adaptive mobile applications SPL. Expert Systems with applications **41**(16), 7549–7564 (2014)
22. Mohalik, S.K., Narendra, N.C., Badrinath, R., Le, D.: Adaptive service-oriented architectures for cyber physical systems. In: IEEE Symposium on Service-Oriented System Engineering, SOSE. pp. 57–62 (2017)
23. de Morais Barroca Filho, I., Junior, G.S.A., Batista, T.V.: Extending and instantiating a software reference architecture for iot-based healthcare applications. In: Int. Conf. on Computational Science and Its Applications. pp. 203–218 (2019)
24. Paschou, M., Sakkopoulos, E., Sourla, E., Tsakalidis, A.: Health internet of things: Metrics and methods for efficient data transfer. Simulation Modelling Practice and Theory **34**, 186 – 199 (2013)
25. Shiffman, S., Stone, A.A., Hufford, M.R.: Ecological momentary assessment. Annu. Rev. Clin. Psychol. **4**, 1–32 (2008)
26. Volpato, T., Oliveira, B.R.N., Garcés, L., Capilla, R., Nakagawa, E.Y.: Two perspectives on reference architecture sustainability. In: Proceedings of the 11th European Conference on Software Architecture: Companion. pp. 188–194. ACM (2017)
27. Weyns, D.: Software engineering of self-adaptive systems: an organised tour and future challenges. Chapter in Handbook of Software Engineering (2017)
28. Williams, P.A.H., McCauley, V.: A rapidly moving target: Conformance with e-health standards for mobile computing. In: 2nd Australian eHealth Informatics and Security Conference (2013)