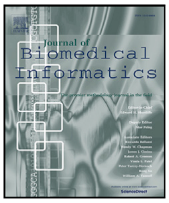
Contents lists available at ScienceDirect

Journal of Biomedical Informatics

journal homepage: www.elsevier.com/locate/yjbin



Original Research

AwarNS: A framework for developing context-aware reactive mobile applications for health and mental health

Alberto González-Pérez [a,*], Miguel Matey-Sanz [a], Carlos Granell [a], Laura Díaz-Sanahuja [b], Juana Bretón-López [b,c], Sven Casteleyn [a]

[a] GEOTEC Research Group, Institute of New Imaging Technologies, Universitat Jaume I, Castellon, 12071, Spain
[b] Department of Basic Psychology, Clinical and Psychobiology, Universitat Jaume I, Castellon, 12071, Spain
[c] CIBER de Fisiopatología de la Obesidad y Nutrición (CIBEROBN), Instituto de Salud Carlos III, Madrid, 28029, Spain

ARTICLE INFO

MSC:
6802
6804

Keywords:
mHealth
Smartphone
Data collection
Intervention
Context-awareness
Digital phenotyping

ABSTRACT

In recent years, interest and investment in health and mental health smartphone apps have grown significantly. However, this growth has not been followed by an increase in quality and the incorporation of more advanced features in such applications. This can be explained by an expanding fragmentation of existing mobile platforms along with more restrictive privacy and battery consumption policies, with a consequent higher complexity of developing such smartphone applications. To help overcome these barriers, there is a need for robust, well-designed software development frameworks which are designed to be reliable, power-efficient and ethical with respect to data collection practices, and which support the sense-analyse-act paradigm typically employed in reactive mHealth applications. In this article, we present the AwarNS Framework, a context-aware modular software development framework for Android smartphones, which facilitates transparent, reliable, passive and active data sampling running in the background (sense), on-device and server-side data analysis (analyse), and context-aware just-in-time offline and online intervention capabilities (act). It is based on the principles of versatility, reliability, privacy, reusability, and testability. It offers built-in modules for capturing smartphone and associated wearable sensor data (e.g. IMU sensors, geolocation, Wi-Fi and Bluetooth scans, physical activity, battery level, heart rate), analysis modules for data transformation, selection and filtering, performing geofencing analysis and machine learning regression and classification, and act modules for persistence and various notification deliveries. We describe the framework's design principles and architecture design, explain its capabilities and implementation, and demonstrate its use at the hand of real-life case studies implementing various mobile interventions for different mental disorders used in clinical practice.

1. Introduction

As of 2021, roughly 80% of the world's population has access to a smartphone [1,2]. This widespread proliferation, along with the possibilities these devices bring to health [3] and particularly mental health [4], provide a promising opportunity to make health services accessible to a broader population. They are capable of reducing barriers to access to (mental) health care [5], such as cost, availability of treatment or social stigma of disorders [6]. Mobile Health (mHealth) apps have the potential to deliver psychological treatment through patients' self-applied interventions or by following a blended therapy format [4,7]. While healthcare service users are increasingly willing to use such medical apps, even more in the post-COVID era [8], also practitioners stand to benefit from higher availability, potential cost reductions [9] and faster access to relevant information [10]. In 2020, investment in mHealth was the third biggest investment in digital health worldwide [11].

While a large body of studies praises their potential, others highlight the difficulty to get mobile applications to support mental health done right [12,13]. Technical complexity, development costs, regulatory requirements and user acceptance form important challenges towards effective, full-fledged mobile health solutions [14].

One decade after the initial optimism and expectations raised at the start of the mHealth revolution [15], the reality is that we have barely scratched the surface of what smartphones can technically offer [16]. According to a recent systematic review [17], the majority of mobile apps developed for mental health disorders do not yet consider the variety and potential of the device's embedded sensors

* Corresponding author.
E-mail addresses: alberto.gonzalez@uji.es (A. González-Pérez), matey@uji.es (M. Matey-Sanz), carlos.granell@uji.es (C. Granell), lsanahuj@uji.es (L. Díaz-Sanahuja), breton@uji.es (J. Bretón-López), sven.casteleyn@uji.es (S. Casteleyn).

https://doi.org/10.1016/j.jbi.2023.104359
Received 21 October 2022; Received in revised form 10 March 2023; Accepted 5 April 2023
Available online 10 April 2023

or advanced (client-side) data analytics, even though these are key ingredients to monitor and understand the patient's behaviour in real-time, and offer *true* ecological momentary assessments (EMA) and interventions (EMI) [18]. Indeed, a recent study confirms that only a small percentage of health apps in commercial marketplaces harness the potential of built-in smartphone sensors for diagnosis and treatment [19]. The technical and managerial complexity inherent to developing multidisciplinary mHealth apps [14] is hampering more advanced mobile solutions and is further exacerbated by constant technological change. The diversity of smartphone manufacturers, each with their software layer(s) on top of the mobile operating system, which imposes operational restrictions due to privacy and battery concerns, further complicates development [20]. Indeed, several mHealth applications report unreliability of their data collection processes resulting in missing data [21]. Data missingness has consistently been a hurdle for smartphone-based data collection, and – with few exceptions – remains largely unreported. In addition, there are other technical challenges and trade-offs (battery usage versus data collection strategy; real-time versus batch data processing) for which there is no consensus on their solution, as the use cases of each application have specific technical needs and requirements [22].

Closely related to technical challenges is the cost incurred to develop mobile mHealth apps. It is estimated that even a low-feature mobile health app costs between $70.000 and $150.000 when developed from scratch, an amount higher than typically foreseen by research grants [14,23]. Even adding a feature to an existing app proves burdensome, and financial restrictions may stall development [24]. When more advanced features – such as passive sensing or advanced data analysis [20,22] – need to be implemented, development costs increase significantly. Feature fragmentation, where different OS versions require different implementations for the same feature, further stresses development efforts [25]. The current shortage of software developers [26], and the need for multi-disciplinary teams (healthcare specialists — software developers) [14], puts further pressure on the development costs. Given these budgetary constraints, researchers mostly opt to use "generic", closed-box solutions developed by industry, which generally do not fully cover their needs [27]. For example, re-usable development libraries by industry giants, such as Apple's ResearchKit, or the more recently proposed "no-coding platforms", do not cover more complex necessities of state-of-the-art mHealth solutions [28]. Even though the costs of developing and maintaining mHealth apps are known to be high, and dependent on each specific use case covered, they remain largely unrecognised and undocumented in the literature [29], and mHealth app development costs are seen as an impediment to advance in the domain [30], as is development time [27].

Finally, regarding regulatory requirements, several studies highlight an almost general lack of privacy policies and Terms of Agreement in mental health apps and, when present, they often show misleading language [13,31]. This is particularly concerning in apps that constantly monitor the private and personal data of patients, as users fear their data may be disclosed [32]. In the long term, the lack of attention to privacy might generate an increasing mistrust among current and potential users [12], and negatively affect uptake and retention rates of mHealth applications. Regulatory initiatives, such as Europe's GDPR, have been raising awareness of the issue, and within the research community, privacy design and development guidelines [33] and trustworthiness checklists [34] have been proposed. "Privacy by design", whereby mHealth apps and frameworks incorporate privacy in their development process, is hereby considered a promising research direction.

In summary, developing mHealth apps is a challenging, costly and time-consuming endeavour, even for relatively simple apps, and all the more so for apps which aim to systematically (passively) collect sensor and/or patients data, analyse and act in (quasi) real-time upon it, as is required for state-of-art EMAs and EMIs. Furthermore, the recent research in digital biomarkers and digital phenotyping [35], which aims to quantify human behaviours with respect to mental health, requires reliable, high-frequency monitoring information over a longer period. In this article, we aim to mitigate these technical, budgetary and regulatory challenges, by offering AwarNS, an Android-based open-source, modular, re-usable and extensible context-aware development framework. It follows the sense-analyse-act paradigm, widely used in Internet of Things-driven scenarios [36], and encapsulates shared, common functionalities of this paradigm. At the core of the framework lies our NativeScript Task Dispatcher (NTD), a resource-efficient and reliable task scheduling and execution model, which was empirically proven to provide systematic reliable background execution (e.g., sensing) over extended periods in Android smartphones [20]. Its reactive, event-driven task model furthermore allows running arbitrary tasks triggered by temporal or data-driven events, which enables app developers to implement just-in-time, context-aware EMAs and EMIs. AwarNS provides the necessary data collection mechanisms and data representation models (sense), real-time analysis facilities (analyse), and generic, extensible tasks (act), while being extensible for developers to plug in custom analysis and concrete domain- or application-specific behaviour as required by the application scenario at hand. Furthermore, it operates solely on the phone, with no external server-side dependencies – even though optional remote server synchronisation is supported – , making it suitable for offline and privacy-strict scenarios.

Table 1
Statement of significance.

Problem	There exists no generic solution to reliably sample patients' context through smartphones, analyse the data and react to its changes on the device, which is critical for the success of digital phenotyping and just-in-time assessments and interventions.
What is Already Known	Existing tools can sample the context, but they are either intrusive or struggle with phone OS' background execution restrictions. Only a few can deliver assessments and interventions but rely on external servers to analyse the data.
What this Paper Adds	A versatile, reusable framework to develop reactive mHealth applications, based on reliable systematic context sampling while preserving privacy, to enable timely (offline) assessments and interventions.

AwarNS thus allows mHealth application developers to monitor and detect changes in a wide variety of contextual features offered by smartphones, while solving and hiding the technical complexity of reliable (background) data collection and (real-time) task scheduling on smartphones, even across subsequent OS versions. The modular nature of the framework and the availability of flexible, re-usable sense-analyse-act building blocks, along with its open-source nature (see [37]) and extensive technical documentation, helps to minimise development time and cost. AwarNS was furthermore designed with privacy and regulatory concerns in mind (e.g., the possibility to apply data filtering or transformations before storage), and has proven its usefulness in the development of various mHealth applications (see Section 5). Table 1 summarises the significance of this work.

In literature, researchers have indeed recognised the need to reduce development burden and costs, first by reusing applications and tools [38–40], and beyond this, through generic, adaptable solutions – just like AwarNS – that aim to simplify the development, maintenance and cost of context-aware (mHealth) apps. A representative selection of the most relevant solutions is presented in Section 6, and compared to our solution based on an extended version of Kumar et al.'s comparison dimensions [41]. We distinguish between application development frameworks [42–45], such as AwarNS, which allow developers to implement their mHealth application using common and reusable functionality, and software platforms [46–53], which are aimed at lab technicians and offer out-of-the-box tools configurable to some degree. In both groups, open-source solutions allow some degree of modification to suit specific application use cases [38,39], while commercial platforms are closed-sourced and heavily restricted in customisation [27]. As identified in a recent systematic review by Kumar et al. [41], these solutions have shown the potential to simplify the implementation of behavioural (e.g., risk behaviours, obesity monitoring, assessing the relationship between behavioural trends and academic performance, etc.), mental health (e.g., tracking symptoms of depression, anxiety, bipolar disorder, etc.) and physiological health (e.g., heart rate monitoring, detect and intervene on heart failure, elevated blood pressure, etc.) mobile applications. However, as the authors point out: "[a]dding support for high-level feature extraction on smartphones has the potential of enabling the design of advanced just-in-time interventions in health sensing applications" [41, p. 8:20]. The ability to analyse collected data on the device and deliver just-in-time interventions based on the results of those analyses is, precisely, one of the key differentiating factors of AwarNS. All of the analysed solutions have been used in a variety of use cases, demonstrating their versatility and re-usability. AwarNS positions itself alongside the open-source application development frameworks yet distinguishes itself by:

- its modular and pluggable software architecture, which allows including or excluding functionality as required by each use case.
- its proven reliable background data collection engine, which ensures an uninterrupted data flow.
- its ability to schedule and execute arbitrary tasks (i.e. custom code) based on data (e.g., the user's location), temporal (e.g., daily) or external triggers (e.g., UI interactions, server-sent events), or other tasks' results, which enables the delivery of just-in-time assessments (EMAs) and interventions (EMIs).
- the seamless integration and scheduling of sense-analyse-act tasks, where tasks can be chained and one task may act upon or trigger other tasks, which means that the result of a data collection task may trigger an (on-device or server-side) data analysis task, which in turn may trigger additional data collection and/or an intervention action (EMI).
- the inclusion of primitives to support the sense-analyse-act paradigm (e.g. passive and active data collection, mobile phone and wearable sensors support, built-in location-based and machine learning-based analyses).
- its built-in support for Test-driven Development (TDD), which promotes the technical soundness of framework extensions and usage.

The rest of the paper is structured as follows. Sections 2–4 present the design principles, the architecture design and implementation highlights of the framework. Section 5 presents some of the use cases and applications where the framework has been used. Section 6 includes qualitative analysis and comparison with existing solutions. Last, Section 7 includes a conclusion of the work presented here and future work.

2. AwarNS framework: Design principles

The AwarNS Framework is a mobile software framework intended to simplify the development of smartphone applications that base their functionality on systematic data acquisition and/or event-based task executions, and are capable of reacting to changes detected in the context of the user, all being done right on the device. AwarNS-based smartphone applications silently run in the background, monitoring the patients and their context through passive or active data captures, until a relevant event occurs. Captured events trigger custom, user-developed actions to, for instance, analyse collected data, gather additional information, request further input or actions, or communicate to the user. Even though it is more generally applicable, AwarNS was conceived in the context of health and mental health applications (see Section 5).

The design of the framework was informed by an extensive systematic literature review [54] studying and comparing 158 mental health mobile applications to understand the status quo and identify common components, a comprehensive comparison of existing generic solutions to develop mHealth applications (see Section 6), and our own experience developing a variety of mHealth applications used in research and clinical practice (see Section 5). Based on this, a set of design principles was discerned – versatility (i.e., enabling distinct use cases), reliable execution, privacy preservation, code reuse, and software verification and validation through well-established testing practices – which we list and explain below.

2.1. Versatility

Versatile software solutions help reduce technical complexity and development costs by hiding intricate technical details behind simpler – composable and extendable – abstractions whilst at the same time reducing developer overhead by allowing the selection of subsets of segregated features (modularity) [55]. mHealth applications with different requirements imply multiple and diverse features, some of them are common across many apps, while others are domain- or even app-specific. AwarNS aims to incorporate a broad set of common features, while still allowing app developers the flexibility to select which features to use and/or customise, to compose a new feature from the pool of existing features, or to add new custom features. While this promotes feature and code reusability, it also requires a certain degree of versatility to allow developers to adapt, extend, or modify AwarNS to concrete needs and contexts. AwarNS achieves this through modularity, composability and extensibility: it offers a modular and versatile design composed of a minimum set of (required) features through a core package, while the rest of the features are contained in isolated packages (modules) that can be optionally independently included in apps. Furthermore, AwarNS promotes the development of small, specific features as extensions and compositions of existing features (see next Section 3), and a plugin architecture which allows developers to develop and integrate completely new packages.

2.2. Reliability

Reliable software solutions also aim to reduce development costs and increase user acceptance, as they lead to fewer reported errors that need to be addressed and overall higher user satisfaction because the software works as intended [56]. A natural consequence of custom and from-scratch app development is that every time, similar and recurring problems are faced, and critical issues may be over-sighted. Reliable background task scheduling is one such recurring problem, which is technically challenging and further complicated by subtle barriers that some mobile OS (custom layers) impose. This results in missing data samples when an event occurs, causing the data and the associated event to be lost. Indeed, several mHealth researchers report a significant amount of missing data measurements (e.g. [21]), which consequently negatively impacts the subsequent data analyses and (assessment and intervention) decisions made later on. The AwarNS framework employs at its core the NativeScript Task Dispatcher, which is experimentally shown to be highly reliable when it comes to executing scheduled background tasks (e.g. data measurements — passive sensing) [20]. It takes into account feature fragmentation, hides the technical complexity for app developers, and provides abstractions to simplify the definition and planning of these background activities.

2.3. Privacy

Privacy-preserving software solutions help to meet regulatory requirements and increase user acceptance by working with and storing – anonymised when possible – minimally necessary sensitive data while respecting user's privacy [57]. Detecting changes in multiple health conditions means that a large variety and amount of data must be collected and analysed, but for ethical and privacy reasons, only the information relevant to the clinical practice should be stored. Collecting and storing unnecessary data is considered invasive to users' privacy and against ethical standards in clinical practices. Therefore, it is vital that mHealth apps only store the necessary patient data and avoid data exchange and/or external storage when possible. AwarNS's design takes these considerations in mind in several ways. Firstly, it works completely on the user's phone and offers complete freedom regarding which data to collect and at which sampling frequency, to adjust it to the needs of each use case. The data collection and storage processes are also completely decoupled, allowing the collected data to be processed in memory and only kept as long as necessary before it is discarded (e.g., location data used to determine presence at a certain place). By default, data storage occurs locally first (on the user's device), optionally offering the ability to implement custom adapters to synchronise that data remotely, where additional filters (before synchronisation) can be applied. Lastly, related to transparency, every acquisition of sensitive data in the background is done through a foreground service, which comes with a notification informing the user that the AwarNS Framework is being used by the app, only while data is being collected in the background.

2.4. Re-usability

Re-usable software solutions aim to reduce technical complexity and development costs by fully or partially covering the features required by one or more application use cases [58]. Common practice reported in mHealth literature is to create ad-hoc apps for a particular study case and/or cohort of patients, which are often unavailable beyond the original research setting and are closed-source. A logical consequence is that shared functionalities, such as systematic sensor data collection or sending notifications, are re-developed over and over again, and valuable implementation experience and best practices are lost. In this research, we couple an extensive literature review [54] to understand the necessities and current state-of-the-art in the field, a thorough understanding of the technical challenges [20,22] and our multiple years of experience developing mHealth solutions for different usage scenarios (see Section 5), to bundle base/core processes and features to develop context-aware (reactive) mHealth applications into the AwarNS Framework. Therefore, AwarNS offers common re-usable functionality such as reliable task scheduling and execution, acquiring built-in and wearable sensor data, active sensing, data persistence, (optional) remote collected/generated data synchronisation, notifications delivery and specific data analysis (i.e. geofencing, machine learning algorithms), all while hiding the underlying technicalities from the mobile app developers. The framework is designed to be modular and re-usable, yet also extensible through a plugin-based architecture. This allows developers to compose new features from a pool of existing and thoroughly tested ones while keeping the freedom to customise them or extend the framework with use case-specific functionality. Indeed, the evolution towards re-usable software development is common in maturing fields, where basic necessities and common solution patterns have crystalised.

2.5. Testability

Testable software solutions help to reduce development costs and meet regulatory requirements by enabling test automation, which reduces time spent on software validation and increases the chances of complying with Software as a Medical Device (SaMD) regulations [59]. Medical devices pass rigorous testing procedures to ensure they work as expected in typical usage scenarios. They must conform to a specification, be validated by experts and put in practice through several trials during their development before being granted a trust certification. We have a strong belief that health and mental health apps should be no exception, and it is unacceptable to deliver a medical application whose features have not been verified and validated by technicians and experts. AwarNS addresses technical verification, by applying well-established software testing practices and paradigms which combine acceptance, integration and unit testing. We followed the well-known Test-driven Development (TDD) paradigm, also known as test-first development, ensuring that the implementation of the Framework's logic and features is technically sound. The used simulation and testing tools allow us to simulate any possible usage scenario without the need to manually perform it, phone in hand, and developed tests are incremental, to test any task in isolation or combination. Furthermore, all testing artefacts that were internally used are also exposed for external validation and use by mobile app developers when using or extending AwarNS.

3. AwarNS framework: Design and building blocks

AwarNS has been designed as a modular framework, where a Core package defines a minimal but essential set of features to help mHealth application developers to create mobile apps exhibiting advanced context-aware features. In addition, the Core package establishes a common language for context awareness – in terms of protocols and data models – so that each optional module communicates consistently with the Core package and with all other modules. The Core package and some other modules offering commonly shared functionality, which are grouped under the Common category in Fig. 1, form the basis upon which application developers can develop their mobile apps according to the Sense-Analyse-Act paradigm. Modules belonging to the sense category monitor end users and their environment. They observe some phenomena and dump collected data into the phone's volatile memory. Analysis modules require data from the sensing modules or from other analyses, to compute and/or transform it into relevant outputs. Act modules perform concrete actions towards end users, and typically react to the results of certain analyses (analysis module). Certain modules belong to more than one category. For example, a module in charge of requesting input from the user acts and senses at the same time.

The typical workflow is as follows: tasks contained in sensing modules trigger tasks in analysis modules, which in turn trigger actions declared in acting modules. Nevertheless, each phase is optional (e.g. an action can be triggered immediately after data collection, without any analysis, for example, to persist data), and within each phase, the developer can choose which modules to include and which not according to the application scenario at hand. As such, the framework complies with the re-usability and versatility principles, and the modular nature of the Framework, along with the tools provided by the Core package, makes it natural to re-use functionality common to many mHealth applications, as well as implement custom modules with bespoke, single-purpose feature sets. Furthermore, the separation of concerns greatly facilitates the implementation of best practices to preserve user privacy, ensure code reuse, and ease testability.

AwarNS relies on an event-driven architecture and uses the Reactive Programming Paradigm: events drive the workflow execution, and trigger reactive tasks. In AwarNS terminology, workflows are thus known as task graphs, in which nodes are tasks and vertices denote task execution orchestration triggered by events. Tasks encode the functionality of the app, and are either provided by the framework (e.g., sending notifications to the user, acquiring the user's location, persisting some information), based on templates provided by the AwarNS framework (e.g., generic data acquisition primitives), or custom-made. Tasks are

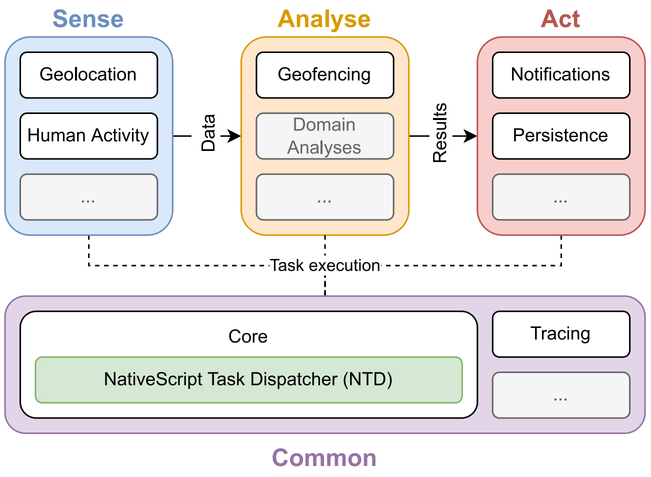


Fig. 1. AwarNS Framework conceptual architecture overview, including the module categorisation based on the Sense-Analyse-Act paradigm.

reactive to events, which are generated as a result of task execution or by non-AwarNS entities, such as a button tap or server-sent event. Finally, data – resulting from tasks (e.g., a data analysis) or specialised data providers used by tasks (e.g., a geolocation data provider) – flows through the workflow in the form of records, which are extended from a common data model.

Fig. 2 illustrates an example of a task graph for a simplified use case of an intervention for patients with gambling disorder. It shows the graphical representation of the corresponding application workflow instrumented with the AwarNS Framework, containing sense, analyse and act modules contextualised to the domains of human activity, geolocation, geofencing, notifications and persistence.

The application workflow in Fig. 2 is organised according to the sense-analyse-act paradigm (respectively blue, yellow and red in Fig. 2), and can be logically divided into four operational phases: workflow setup, movement detection, location sampling and intervention. The first phase, workflow setup, consists of the configuration and wrap-up of the workflow, which are respectively triggered by the `startEvent` and `stopEvent`, which the application can use to enable and disable the workflow at any time. The start and stop events enable and disable the detection of changes in human activity, which guides the rest of the workflow. The second phase, movement detection, consists of activating or deactivating recurrent (every minute) geolocation capturing when the user starts (`userFinishedBeingStill` event) or stops (`userStartedBeingStill` event) moving, saving phone resources when possible. In the third phase, location sampling, each time a location is acquired, it is analysed to detect entering or exiting an area of interest (AOI), which are gambling places in this use case. In the fourth phase, intervention, whenever entering (`movedInsideAreaOfInterest` event) or exiting (`movedOutsideAreaOfInterest` event) an area of interest, a notification is sent to the patient, to dissuade the user from staying within the area (alert) or to encourage the user to keep avoiding such areas after leaving them (reinforcement), just as it would do when supporting a stimulus control treatment component. Throughout the process, all relevant collected data is also stored locally (using the `writeRecords` task), i.e. changes in human activity, captured locations and variations in the distance (proximity) to specified gambling areas. To preserve users' privacy, locations are filtered (using the `filterGeolocationByAoIProximity` task) to keep only those within the range of a gambling establishment, omitting thus those locations that are not of interest from a clinical practice perspective.

Tasks are implemented in complete isolation, making them independent and testable, using the test utilities provided by the Framework. Tasks can be either custom code implemented by a developer from scratch or based on templates provided by the Framework. In the latter case, developers are supported with built-in utilities. For example, for both sense tasks in Fig. 2, i.e. acquiring human activity and geolocation data, a template is provided by the Framework's Core package. Developers only need to specify the output data type of the task and how it will be obtained, i.e. by specifying the data provider. For example, the "Human Activity Detection" box wraps two tasks to start and stop the detection of changes in the activity of the user based on the data from the corresponding data provider (`PushHumanActivityprovider`). A data provider may deploy a push or pull-based strategy for data retrieval. For example, `PushHumanActivityprovider` uses a push-based strategy, as it is unpredictable when changes in human activity will occur, while `PullGeolocationProvider` deploys a pull-based strategy, as geolocation data is likely to be available at a relatively high frequency.

The Framework provides a common data model (`Record`) to facilitate standardised communication with the Core package and other modules. This common data model is extended by the `HumanActivityChange`, `Geolocation` and `AoIProximityChange` entities (ovals in Fig. 2). As a result, the task in charge of locally storing the collected data (`writeRecords`) understands its content and facilitates its later retrieval. This common data model can be extended and used by sensing, analysis and acting tasks to inform about key events that happen over time. AwarNS distinguishes two types of data, namely those corresponding to instantaneous and long-lived events, each with its own representation. While the former is based on single captured data points (i.e., geolocations), the second is based on changes in state over longer time periods (i.e., human activity). Support for both data representations in AwarNS's common data model ensures specific supporting primitives for querying, analysing and storing such data.

4. AwarNS framework: Implementation

The AwarNS Framework has been developed using NativeScript, a cross-platform mobile application development framework, where developers can code the application views in HTML/CSS and the application logic in JavaScript/TypeScript. Unlike other similar web-based solutions (e.g., Ionic), NativeScript produces native-like applications, and works with any modern web development framework or library (e.g., Angular, Vue, React, etc.) or without a framework at all (i.e., vanilla JavaScript). It also exposes all the existing native APIs through JavaScript and is interoperable with other well-known mobile application development frameworks such as Ionic (Portals) and Flutter (UI). Indeed, as of 2021, the chosen technologies employ the most widely used programming languages [60], with solid support and large developer communities and backed by organisations like the OpenJS Foundation. This favours having a development team to work on a web-based technology stack for the development of mobile applications, without hiring another specialised team, for example, in web development (e.g., to create dashboards, etc.) [25]. Despite the use of a cross-platform solution, the Framework is currently only supported on Android devices, due to more restricted access to detection capabilities and background scheduling in iOS; further investigation on how to cope with iOS restrictions is foreseen.

During the development of AwarNS-based real-life applications (see Section 5) and informed by our extensive literature review [54], we identified several generic, shared sense, analyse and act features and implemented them as independent, re-usable modules within the framework. That is, the AwarNS Framework follows a modular architecture, where the Core package is the central actor that provides the essential building blocks for context-aware (mHealth) applications, and various optional modules provide more specific functionality. Fig. 3 shows all currently available modules and the dependencies between them represented as overlapping shapes. The Core package and Tracing module

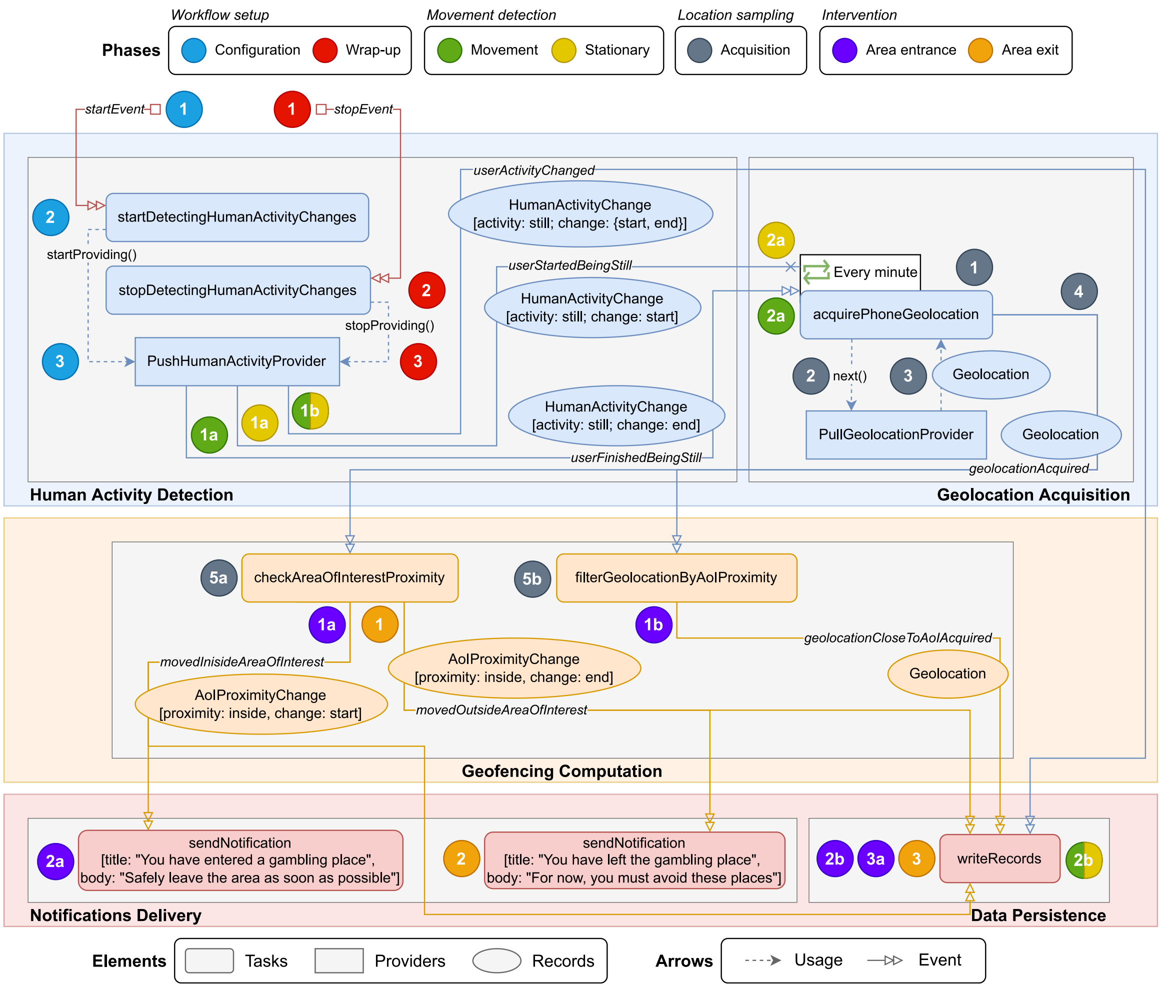

Phases
Workflow setup
Configuration
Wrap-up
Movement detection
Movement
Stationary
Location sampling
Acquisition
Intervention
Area entrance
Area exit
startEvent
stopEvent
startDetectingHumanActivityChanges
stopDetectingHumanActivityChanges
PushHumanActivityProvider
startProviding()
stopProviding()
userActivityChanged
HumanActivityChange [activity: still; change: {start, end}]
userStartedBeingStill
HumanActivityChange [activity: still; change: start]
HumanActivityChange [activity: still; change: end]
userFinishedBeingStill
Every minute
acquirePhoneGeolocation
next()
Geolocation
PullGeolocationProvider
geolocationAcquired
Human Activity Detection
Geolocation Acquisition
Sense
checkAreaOfInterestProximity
filterGeolocationByAoIProximity
movedInisideAreaOfInterest
AoIProximityChange [proximity: inside, change: start]
AoIProximityChange [proximity: inside, change: end]
movedOutsideAreaOfInterest
geolocationCloseToAoIAcquired
Geolocation
Geofencing Computation
Analyse
sendNotification [title: "You have entered a gambling place", body: "Safely leave the area as soon as possible"]
sendNotification [title: "You have left the gambling place", body: "For now, you must avoid these places"]
writeRecords
Notifications Delivery
Data Persistence
Act
Elements
Tasks
Providers
Records
Arrows
Usage
Event

Fig. 2. High-level operation example of the AwarNS Framework applied to intervention for gambling disorder. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

form the Common category on top of which the Framework modules are grouped according to the Sense-Analyse-Act paradigm (see Fig. 1). This section is accordingly divided into the Common modules, and the sensing, analysis and acting modules. Because AwarNS is an extendible software framework, we also describe the mechanisms which allow the development of custom sensing, analysis or acting modules/features.

4.1. Common modules

4.1.1. The core package

The heart of the AwarNS Framework is the Core package, which offers a scheduling mechanism and convenient building blocks to build (mHealth) applications that gather, analyse and react to contextual information. It sits on top of the NativeScript Task Dispatcher (NTD) library, previously developed by the authors [20]. The NTD provides domain-agnostic task definition and background execution mechanisms, where arbitrary tasks can be triggered by temporal-, data-driven or external events. That is why, on top of NTD, the Core package provides further primitives and abstractions to facilitate sense-analyse-act workflows, namely, (1) Provider Task Templates, which supply generic data acquisition support, to greatly simplify the implementation of data acquisition tasks; (2) Data Provider interfaces, which specify how data is collected and whether the pre-conditions for performing the data collection are met, and their concrete Data Provider implementations, which can be plugged into Task Templates and provide concrete data collection; and (3) the Record, a data model abstraction to represent the data collected and generated by the Framework and its extensions,

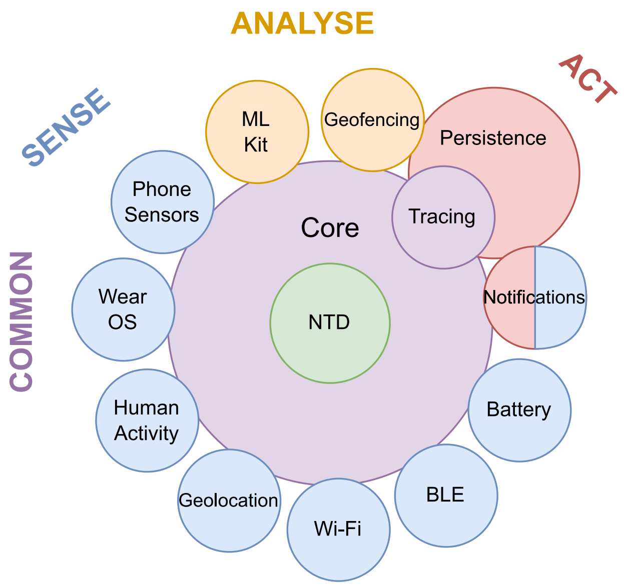


Fig. 3. AwarNS Framework modules. Touching edges represent dependencies between modules. Colours represent categories. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

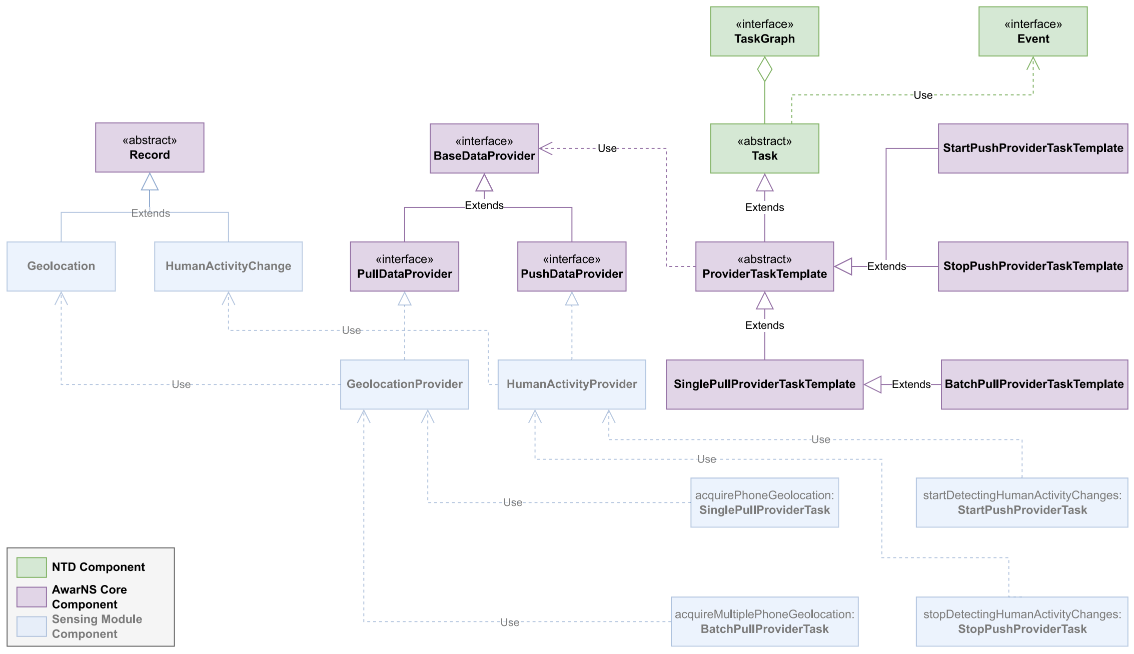

«interface» TaskGraph
«interface» Event
Use
«abstract» Record
«interface» BaseDataProvider
«abstract» Task
StartPushProviderTaskTemplate
Extends
Geolocation
HumanActivityChange
«interface» PullDataProvider
«interface» PushDataProvider
«abstract» ProviderTaskTemplate
StopPushProviderTaskTemplate
GeolocationProvider
HumanActivityProvider
SinglePullProviderTaskTemplate
BatchPullProviderTaskTemplate
acquirePhoneGeolocation: SinglePullProviderTask
startDetectingHumanActivityChanges: StartPushProviderTask
acquireMultiplePhoneGeolocation: BatchPullProviderTask
stopDetectingHumanActivityChanges: StopPushProviderTask
NTD Component
AwarNS Core Component
Sensing Module Component

Fig. 4. Core Package UML diagram. Includes examples of the extension of the provided entities by the Geolocation and Human Activity modules.

allowing standardisation of communication between the Framework entities. Fig. 4 shows the relationship between these entities and also includes examples of concrete implementations provided by AwarNS that correspond to the use case discussed in Section 3.

The core at the core: the NTD library

The NTD library allows the specification and execution of event-driven, reactive workflows. It declares three types of basic entities: Task graphs, Tasks and Events. Tasks are self-contained code units, where developers can implement any kind of logic for the device to execute in the background. Additionally, the developer can state any task execution prerequisites (i.e., permissions, available hardware or disabled features) and the way to fulfil them (e.g., asking the user to perform certain actions). Tasks can run based on temporal events, data-driven events (i.e., produced by other tasks, typically sensing or data analysis tasks) or external events (i.e., based on UI, external hardware, or server-sent events). Tasks can also declare if they have additional specific requirements for the system to execute them, for example, if they must run in a foreground service because they collect sensitive information or they require specific permissions from the user. Upon finishing their execution, tasks always produce an event, which possibly includes relevant data (e.g., geolocation). Events are the task execution drivers and their main data communication mechanism. Events can trigger the execution of one or more (reactive) tasks, and, since tasks always produce events, their execution can be chained. Finally, task graphs are the mechanisms to facilitate the composition of complex reactive task execution workflows. The Core package exposes these basic entities for the implementation of the various AwarNS modules (see below), for future extensions of the framework and for application developers to implement their custom application logic. The NTD follows the design guidelines presented in [20], was extensively tested and shown to ensure highly reliable, long-term (background) data collection, which corresponds with one of the main design principles of the AwarNS Framework.

Simplifying data collection processes: Provider task templates and data providers

Provider Task Templates and Data Providers seamlessly work together to ease data collection. Provider Task templates define the generic logic of the data collection processes and provide a way to integrate them into task graphs, taking advantage of background and event-based task execution behaviour. Data Providers are actual implementations, which are plugged into Provider Task Templates, capable of connecting to a specific source and acquiring the data, and checking and enforcing the requirements imposed by the data source (e.g., permission or feature enable requests, authentication).

As briefly covered in Section 3, the AwarNS Framework supports push and pull data retrieval and offers corresponding Provider Task Templates and Data Providers to achieve this. For pulling data from a source, the Framework comes with single and batch provider task templates. The former defines the logic to acquire single data updates from a data source, while the latter defines the logic to keep polling a data source in a timeframe determined by the frequency at which the task becomes invoked. To connect to a concrete data source and pull the actual data, both single and batch task templates use a concrete implementation of a `PullDataProvider` . The captured data is subsequently encapsulated in an event, which is emitted by the corresponding task template upon completion. For single data, a single data point is captured in the task completion event, while for batch data pulling, all the data captured during the polling timeframe is encapsulated. In Fig. 4, the `acquirePhoneGeolocationTask` instantiates a single pull task template, while the `acquireMultiplePhoneGeolocationTask` instantiates a batch task template; both of them using a `GeolocationProvider`, a concrete implementation of the pull Data Provider which connects to the mobile phone's GPS sensor to capture geolocations.

For requesting data updates via push, the Framework provides two task templates that define the logic for starting and stopping listening for push-based events. As for pull-based task templates, they require a concrete implementation of a Data Provider, namely a `PushDataProvider`, to connect to a concrete push-based data source and acquire the actual data. Push-based Data Providers set up a subscription for the app to listen for certain updates pushed by an external entity (e.g., a system API pushing detected physical activity changes, instructing some external device to start reporting data updates, etc.). Examples of concrete instances of push-based task templates are the `startDetectingHumanActivityChanges` and `stopDetectingHumanActivityChanges` tasks in Fig. 4, supported by

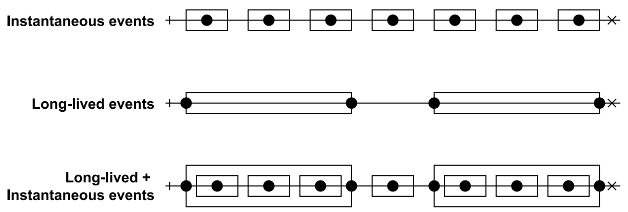


Fig. 5. Representation of the difference between instant and long-lived events, and how they are interrelated over time.

HumanActivityProvider, a concrete implementation of a push Data Provider which connects to the Android Play Services Location library to get human activity updates.

Choosing between push- and pull-based task templates, and their corresponding push- and pull-based data providers, depends on the nature of the data that is collected, how often the app interacts with the data source, and how long it remains active. On one hand, pull-based data providers run on time-triggered Provider Task Templates that regularly poll the provider for new data. This means that it allows specifying when data updates are fetched and the provider is only active for as long as it takes to acquire the data. On the other hand, interactions with a push-based data provider are limited to start and stop sending data updates, in a fire-and-forget manner. This type of data provider fits perfectly in situations in which getting timely updates is crucial, as in the case of change detection, because this type of action is not restricted by the duty cycle of the underlying scheduling mechanism.

Implementation-wise, Provider Task Templates and Data Providers, both described by corresponding interfaces, permit maintaining a separation of concerns regarding the reusability and extensibility of the data collection processes. They enforce consistency with regard to data acquisition, i.e. all data acquisition events generate data consistently and use a well-defined naming scheme. The underlying NativeScript Task Dispatcher furthermore ensures reliable data acquisition execution, without the need for the developer to go into low-level Android task execution mechanisms. These features greatly reduce the developer's effort when implementing data-driven mHealth apps.

A common specification to represent data: Records

By extending the Record abstraction, a developer can obtain a consistent representation of some data obtained at a specific point in time. This type of data can range from a sensor measurement to the result of a complex calculation. These records are passed through the task workflow encapsulated inside events and form the basis for communication between framework modules. AwarNS distinguishes between data records for instant events (i.e., a single data capture, such as a geolocation capture) and long-lived events (e.g., events that are not instantaneous, such as walking), where the start and end times are relevant. Fig. 5 shows, on the two top timelines, the conceptual difference between records representing instant and long-lived events. The bottom timeline shows an example of how the two types can coexist. For example, considering the Geolocation record in Fig. 2, only the acquired data (i.e. geolocation) and timestamp matter, not when the corresponding event (geolocationAcquired) started or ended. In contrast, for long-lived events, their start and end times are relevant (AoIProximityChange), e.g. when the user enters or leaves an area of interest. At an implementation level, long-lived event data records are identified by a change flag in the base Record abstraction, which allows data records to be framed to a particular event and facilitates querying, for example, considering data only while users were running or while they were within a relevant area.

All AwarNS Data Providers generate output entities compliant with the generic Record data model, as must any custom Data Providers. This allows Provider Task Templates to include these data in Events outputted upon ending their execution, ensuring proper data communication between the Core package and framework modules.

Enforcing task and module isolation: Task sandboxing and plugin loaders

Aside from the Task, Event and Task Graph concepts included in the NTD library and exposed by the Core package, the NTD provides two other mechanisms to enhance the development experience, strongly connected to the principles of Testability and Versatility (Section 2). On the one hand, the Task Sandboxing tools simplify the development and testing of Tasks in complete isolation. These tools enable the simulation of task invocation trigger events and capture them for evaluation within a test environment, which is essential for running both closed- and open-box tests against developed tasks. On the other hand, related to the isolation and the versatility principles, the Plugin Loader interface handles the registration of the modules that will be used during the initialisation of the Framework at each application startup. Developers can choose to import (only) those modules required to implement the use case at hand through their associated loader, the same applies to the individual tasks contained within each module, i.e., unused tasks can be excluded. Both the Task Sandboxing tools and the Plugin Loader interface are exposed by the Core package.

4.1.2. The Tracing module

Besides the Core package, AwarNS also includes the Tracing module in the Common category to facilitate the traceability of the execution of several tasks and events chained together. The Tracing module provides tasks and task decorators to ease the debugging of complex task graphs. It offers the makeTraceable decorator that takes a list of task instances and wraps them with logging mechanisms. The resulting traces contain the task name, the execution time, whether it succeeded or failed, the result, and, if necessary, the reason for the task execution failure. Similarly, the module provides the trackEvent and trackSensitiveEvent tasks to log concrete events. The resulting traces contain the name of the event and its content is included in the trace. Both types of traces include a unique chain identifier to keep track of associated events and/or task executions. In correspondence with privacy concerns, traces can be configured to omit sensitive data, allowing developers to specify which content is being logged. All generated traces are stored in a common local data store, which can optionally be synchronised with a remote database, or exported in CSV and JSON data formats, using the Persistence module (see Section 4.4.1).

4.2. Sensing modules

The Framework comes with sensing modules that, through monitoring of physical, physiological and environmental parameters, can be used to quantify the users relative to their environment. Each sensing module wraps a specific hardware (e.g. accelerometer) or software (e.g. human activity through Android's Android Play Services Location library) sensor and allows data readings from them. AwarNS supports most smartphone sensors, such as GPS, accelerometer, gyroscope and magnetometer, and is able to access other smartphone features, such as detecting nearby Wi-Fi or Bluetooth Low Energy devices, or the smartphone battery level, and connect to WearOS-based smartwatches to access its sensory data, such as heart rate, GPS, accelerometer, gyroscope and magnetometer. Some sensors provide access to directly actionable measurements (e.g., the geolocation module), while others require analysis to yield relevant results (e.g. Wi-Fi and BLE modules can be used to detect certain key devices nearby or build indoor positioning systems [61]). In this sense, AwarNS provides the basis for digital phenotyping, as it allows both low- and high-level data to be collected by the sensing modules, and analysis modules to process them and extract physiological biomarkers. For example, the WearOS module

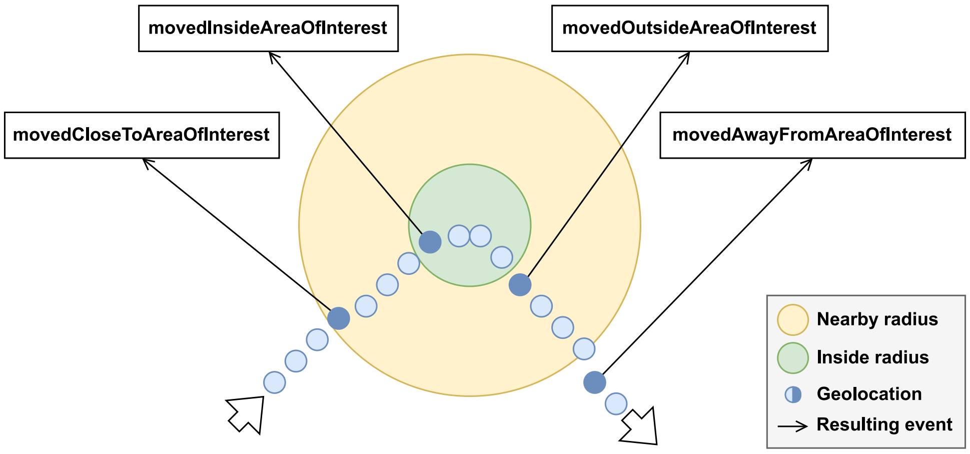

movedInsideAreaOfInterest
movedOutsideAreaOfInterest
movedCloseToAreaOfInterest
movedAwayFromAreaOfInterest
Nearby radius
Inside radius
Geolocation
Resulting event

Fig. 6. Geolocation records (trajectory) analysed by the `checkAreaOfInterestProximity` task and where each relevant event is emitted.

supports heart rate sampling, which can be used to infer nervousness or anxiety [62].

Connecting the AwarNS sensing modules with the core of the framework, we provide in supplemental material a detailed table summarising the tasks related to each module, the events each task produces and the records these events encapsulate when they are emitted. Tasks related to the Geolocation, Wi-Fi, BLE and Battery modules are all pull-based, allowing for the acquisition of new data on demand, both individually and in batches (except for the Battery module). Tasks associated with the Human Activity, Phone Sensors and WearOS modules are all push-based. Push-based tasks do not directly produce events, instead, the ones listed in the table are generated by their associated data provider. All the tasks ensure that execution requirements are met (i.e., required permissions, mandatory features, etc.), and run on a foreground scheduler when needed (i.e., when dealing with sensitive data such as location and IMU sensors).

4.3. Analysis modules

In line with the generality of the Framework, the analysis modules find their use in location analysis, pattern recognition and prediction.

4.3.1. The geofencing module

For location analysis, the Geofencing module can be directly combined with the Geolocation and/or WearOS modules to detect proximity changes of the users to previously defined areas of interest. To do so, the latter modules emit Geolocation records through their corresponding Geolocation data providers, which in turn connect to the phone/smartwatch's sensors. Based on single location or trajectory data input, the predefined task `checkAreaOfInterestProximity` defined in the Geofencing module emits (`movedCloseToAreaOfInterest`), (`movedInsideAreaOfInterest`), (`movedOutsideAreaOfInterest`) or (`movedAwayFromAreaOfInterest`) events, which encapsulate `AoIProximityChange` records to represent long-lived events which determine the start and end time of a user within or nearby an area. Fig. 6 illustrates this process based on a sequence of geolocation records.

Implementation-wise, the Geofencing module includes an API to register, retrieve and remove areas of interest relevant to the application, and it allows observing changes in the registered areas, to act upon them. Additionally, the Geofencing module comes with a predefined task to filter geolocation records right after the acquisition, based on their relative proximity to the defined areas of interest. The `filterGeolocationByAoIProximity` task only outputs geolocation records that are spatially within, or close to, one or more areas. This allows for preserving privacy, as only those geolocation records that are relevant for the purpose of the app (i.e. those acquired while in or near the relevant areas) are persisted. This filtering task is configurable and allows us to (optionally) set the distance, in meters, to interpret when to consider a user close to an area. Due to its generality, the Geofencing module can also be used with other data providers, such as those defined in Wi-Fi and/or BLE modules, to detect proximity.

4.3.2. The ML Kit module

The ML Kit module allows running TensorFlow Lite Machine Learning (ML) models for regression and classification jobs on the phone based on the data generated by other tasks. It does not include ML models but allows the use of models provided by the developer. These models have to be provided as *.tflite* files (i.e., lightweight portable model representation format), including model's metadata (e.g., name, version, description, etc.) with associated labels (i.e., classes) file (only required for classification models). To operate, the module provides the necessary functions to generate regression and classification tasks. These functions require an ML model and an "aim", a string describing the objective of the generated tasks, which is used in their name too. For instance, a regression task named "*aim*Regression" is generated to support a (developer-provided) *aim* (e.g., "stress-level" as aim generates the `stressLevelRegression` task). When such a task is subsequently invoked, the regression or classification is carried out using the specified ML model. Upon completion, the regression or classification task emits the "*aim*Predicted" event containing a `Regression` or a `Classification` record with the results of the model, respectively (e.g., the `stressLevelRegression` task generates the `stressLevelPredicted` event).

4.4. Acting modules

The AwarNS Framework comes with two built-in modules: the Persistence and Notifications modules.

4.4.1. The persistence module

The Persistence module serves to store collected or generated data so that other tasks, application views or remote processes (in case of enabling the optional external synchronisation) can use it. In short, it is in charge of locally handling data storage, data exporting to different file exchange formats and remote synchronisation with external data stores. Focusing on the primary purpose, data storage, the Persistence module includes a software abstraction that takes any custom extension of the record model as input and stores it in a local (CouchBase Lite) database. The interaction with the record storage abstraction can be either manual, using its public API, or automatic, using the `writeRecords` task of

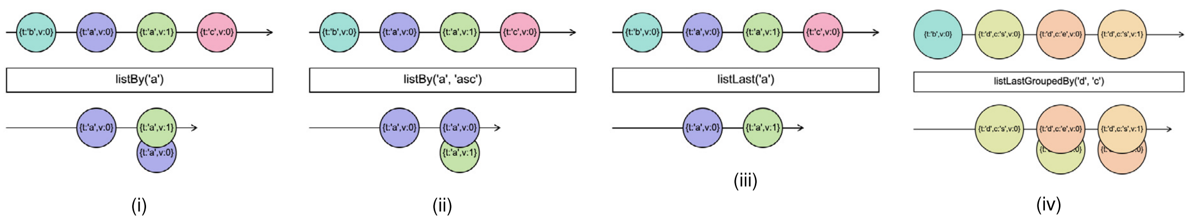


Fig. 7. Marble diagrams schematically showing the Persistence module's advanced query methods: (i) `listBy()` retrieves updates on records of type "a" over time contained within a list (default sorting — descending); (ii) same as (i) but overriding the default sorting behaviour to ascending; (iii) `listLast()` retrieves updates of records of type "a" (only most recent); (iv) `listLastGroupedBy()` to query the most recent updates of records of type "d", grouped by unique values (concretely, "s" i.e. 'start' and "e" i.e. 'end') for property "c" (change). Concretely, this query returns the start en end times for long-lived events, such as entering/exiting an area of interest.

the Persistence module. Either operation automatically triggers a one-way data synchronisation process, which developers can customise via an optionally provided adapter.

The record storage abstraction provides typical persistence operations, such as inserting, retrieving and deleting records. In addition to these persistence operations, the abstraction provides common query methods to access and retrieve records, but also advanced query methods to query changes in them over time. These advanced queries are launched at a certain point in time and return live data streams, which are updated on-the-fly as relevant updates occur over time. Such queries are particularly suitable for change inspection. AwarNS supports four advanced query types, `listBy(recordType)`, `listLast(recordType)` and `listLastGroupedBy(recordType, groupByProperty)`, which are schematically depicted using marble diagrams in Fig. 7. The `listBy()` query method, shown in Figs. 7(a) and 7(b), retrieves records of a certain type, whereby updates of the record type (i.e., overlapping circles in Fig. 7) are ordered in chronologically descending and ascending order, respectively. Aside from filtering records, the advanced methods can also be used to perform complex analyses that combine recently added records – as a result of updates in the data stream – with old ones. An example is the `listLast()` method (Fig. 7(c)), which notifies the existence of updates of a particular record type obtained after the method was called, yet without including the record type history (i.e., only the last received record is added to the result stream; no overlapping circles in Fig. 7(c)). A more sophisticated method is `listLastGroupedBy()`, which retrieves the most recent records of a given type (selection property) grouped by property values. This method is advantageous, for instance, to obtain pairs of records associated with long-lived events, determining the start and end time (group by values) of the event, as shown in Fig. 7(d). Another example is to obtain the most recent visits of a user to all the registered areas of interest. For more fine-grained querying, all the above-advanced methods allow the specification of optional query conditions or filters. These filters can be used, for example, to obtain just the start records of long-lived events, or to obtain the last user's visit to a given area. All in all, the combination of these query methods enables us to fetch real-time data in different ways, as the basis for further analyses.

Next to data storage, the Persistence module also includes mechanisms to export records in CSV and JSON formats. This is particularly useful for applications that need to work completely offline, as users can then download the resulting files and handle them using any (local) tool. By default, the Persistence module completely runs on the mobile (client) device, i.e. offline first. This allows for a better app user experience since the developed applications do not suffer network delays or interruptions when accessing data. Nevertheless, AwarNS also supports one-way synchronisation of the locally stored records into an external system, either locally or remotely. This is done by providing an adapter interface, which developers can implement to connect the Persistence module's storage operations with a remote storage system of choice. Leaving remote storage optional preserves users' privacy and gives freedom to the developers to implement custom logic in the synchronisation adapter, e.g. alter or prevent certain types of data from being remotely stored.

Finally, beyond the storage of records, the Persistence module offers an abstraction mechanism on top of the database to allow an easy definition of app-specific data stores, i.e., not only data records but also other app-specific data (such as notifications, temporary or partial results of analyses, the persistence of the internal state of a module, etc.) stores. For example, in between application wake-ups, the Geofencing module uses an internal data store to keep the proximity status of the nearby areas, and the notifications module stores the list of pending notifications along with their metadata. Via the abstractions offered by the Persistence module, AwarNS provides the following operations on app-specific data stores: single and bulk insert, single and multiple fetches, update, delete, observe changes in the stored data and grant access to the underlying (CouchBase Lite) database engine to perform custom queries.

4.4.2. The notifications module

The second acting module is the Notifications module, which sends notifications to the users via their mobile devices. Next to pure information delivery, such as displaying brief information or providing quick feedback (acting), this module also incorporates sensing features, namely when notifications require feedback from the user (e.g., giving confirmation or answering some questions). Consequently, the Notification module falls both in the sensing and acting categories (see Fig. 3). On the acting side, the Notification module offers a task called `sendNotification`, which allows sending a notification after an event triggers. Apart from the regular properties of a notification, such as a title and body message, it also contains a content type property that indicates the type of content and action associated with the notification (i.e., when tapping the notification). AwarNS supports various actions, such as opening the app (`OPEN_APP`), showing additional textual or multimedia content (`OPEN_CONTENT`), delivering questions (`DELIVER_QUESTIONS`), asking for confirmation (`ASK_CONFIRMATION`) or free-form feedback (`ASK_FEEDBACK`), or developer can implement their custom action. Once received, users can tap on a notification or discard it. Either action triggers a corresponding event, `notificationTapped` and `notificationDiscarded` respectively, which are useful to understand the interaction/involvement of the user with the app and to calculate usage statistics.

On the sensing side, the Notification module offers predefined record types to hold the result of user interaction with the previously mentioned supported content types (except for `OPEN_APP`, which produces no output). The corresponding data record types are `UserReadContent`, used to hold content that was seen including if it was completely seen, `QuestionnaireAnswers` used to hold answers to the delivered questions, `UserFeedback` to store a single answer, and `UserConfirmation` to hold the answer to a yes/no question. As usual, these data records are encapsulated inside events, which the framework emits when a corresponding action was performed.

Finally, the Notification module offers APIs to list unread notifications, mark them as read and set up a notification tap handler, which allows the application developer to select the user interface to show when opening the application through a notification.

4.5. Extending the framework

Aside from the built-in modules presented in this section, application developers can implement custom sensing, analysis or acting features or modules to exhibit domain- or application-specific features. Concretely, the framework can be extended with data providers, new tasks and record types to realise specific features required in custom apps. These features can be either implemented right in the app or wrapped in new external modules, to reuse them across different apps or for the sake of better code organisation.

New sensing features can be implemented using the primitives provided by the Core package: application developers can rely on the framework's base functionality offered by Provider Task Templates, which provide generic data acquisition support, and Data Providers, which provide push- and pull data acquisition, to support custom sensor data acquisition. New analysis features can be implemented as NTD Tasks provided by the Core package, while having the support of AwarNS's basic and advanced querying mechanisms, and machine learning module to integrate regression and classification analysis tasks. Similarly, new acting features can be implemented using Task primitives. Concretely, the Notifications module can be used or extended to deliver more advanced content to the users, such as images, audio or videos, which users can access right after tapping on a notification. Lastly, by implementing the plugin loader interface, any setup and configuration can be ensured.

5. Real-life use cases

The design and development of the AwarNS Framework were driven by continuous analysis of requirements collected while developing AwarNS-based applications in the context of health and mental health domains. In this section, we describe three applications that have been developed and applied to four different use cases, each using a particular configuration of the AwarNS framework and a combination of framework modules. The three applications below have in common the need to regularly and transparently sample the environment and react on time to the detected changes in it. Based on these development experiences and the empirical observation of the needs and requirements of these apps, we seized the opportunity to design and further develop the AwarNS Framework to help all those applications succeed and reduce development time. Through their use in practice, these applications have served as a validation of the Framework features.

The first application is called SyMptOMS, a smartphone application for the delivery of psycho-educational location-based notifications for the treatment of mental disorders associated with specific places. In particular, SyMptOMS has shown positive results when applied to the treatment of panic disorder and agoraphobia [63], and gambling disorder [64]. The second application was the TUG Test, a fully instrumented version of the well-known Timed Up and Go (TUG) test to assess a person's mobility, implemented through a smartphone and a wearable application [65]. The last application is SyMptOMS-ET, a rewrite and extension of the original SyMptOMS application, to automate the process of place-based emotional exposure therapy, using timely assessments and applying a rule-based ecological momentary intervention. Next, we summarise the main features of each application and how they are using the AwarNS Framework.

5.1. SyMptOMS: Geolocated psycho-educational notifications

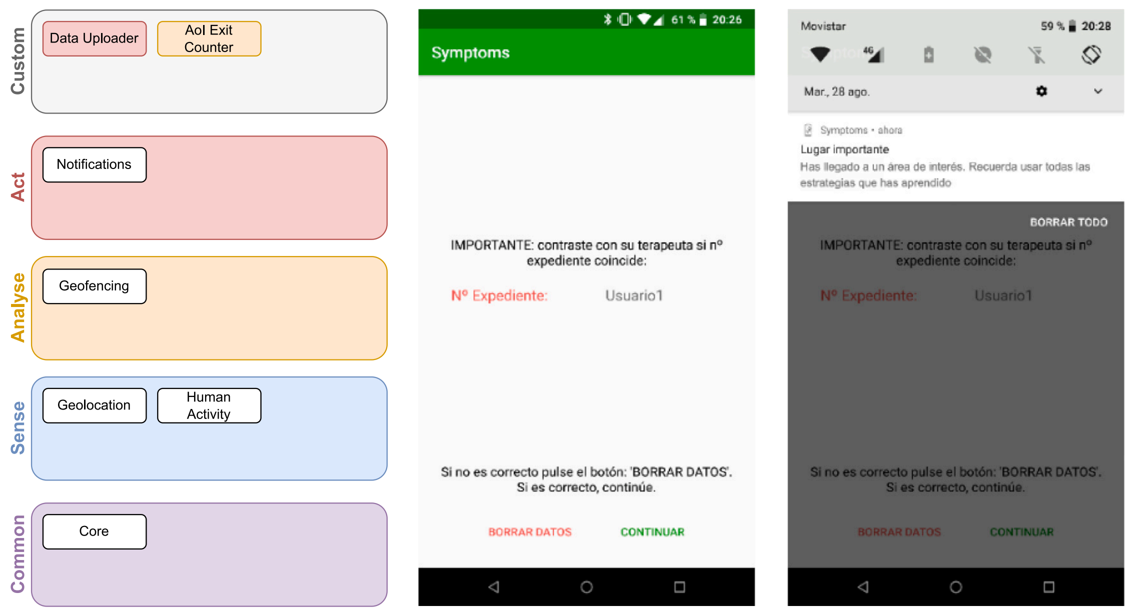
The SyMptOMS application is a companion smartphone app to a web management tool, in which therapists can register patients in treatment, including areas of interest for each patient and personalised messages to be delivered when a patient enters or leaves each specified area of interest. This information is stored, for each patient, in their patient profile. The smartphone app is configured for each patient by downloading their profile and is used as an adjunctive treatment. The app has been successfully used in clinical practice with patients diagnosed with panic disorder and agoraphobia [63], where it delivered the acting instructions of the exposure treatment component, indicating the patient to stay at a given area until reducing the perceived anxiety level. More recently, it has been used as part of a treatment for patients diagnosed with gambling disorder [64], where it was applied in two different treatment phases. First, after the patient has stopped gambling and presents abstinence syndrome, the app supports the stimulus control treatment component, delivering warning messages when detecting the patient's presence nearby a gambling establishment. Second, once the patient has learned more strategies to cope with gambling urges and symptoms of the abstinence syndrome have decreased, the app supports another exposure treatment component, this time with response prevention, instructing the patient to stay outside (but close to) the gambling establishment, without playing, until the urge to gamble lowers. In both use cases, the application was personalised to each patient profile and disorder.

Fig. 8 details the AwarNS modules used by the SyMptOMS app, along with some application screenshots. This app is based on early versions of the Core package and the Geolocation, Human Activity, Geofencing and Notifications modules. Additionally, it declares two custom modules, the Data Uploader and the AoI Exit Counter. The Core package is used for the coordination of all the built-in and custom modules, and the background scheduling of the tasks. The Geolocation module is used to regularly acquire the user's location, every 2 min. The Human Activity package is used for the reporting of physical activity stats. The Geofencing module is used to detect the presence inside or outside one of the registered relevant areas. The app-specific (custom) Data Uploader module enables the direct upload of the collected data to a remote server, with no local persistence. Lastly, the other custom module, the Aoi Exit Counter, is used to count how many times a day the patients left each area. Both the Human Activity and Aoi Exit Counter modules were only used for informational purposes for the therapists; they were not used as part of the psychological intervention. Finally, the Notifications module is used to deliver the enter and/or exit messages, previously defined by the therapist, to the patients when they enter or leave an area. These notifications consisted of dissuading (e.g., "You are in a risk area because nearby there is a place where you gambled. Remember that now it is important to avoid staying here" in case of gambling disorder, for stimulus control) or motivational (e.g., "You have arrived at an area of interest. Remember to use all the strategies you have learned" in case of panic disorder and agoraphobia, and "You are in an area of interest, the exposure begins. If there is an urge to gamble, use the strategies you have learned and leave the area once the urge has decreased" in the case of gambling disorder, at the beginning of the exposure) messages.

5.2. TUG test: Smartphone- and wearable-based instrumented timed up and go test

The TUG Test smartphone application aims to instrumentalise (i.e., automatically obtain results) the execution of the TUG test, a well-known mobility test usually used for fall-risk assessment in elderly people. The application uses the data collected from the Inertial Motion Unit (IMU) sensors of either a smartwatch or smartphone to detect the activities the user executes while performing the TUG test, i.e. stand up, walk, turn around, walk, turn around, sit down. Once done, it computes the amount of time the user spent on each activity. The machine learning models used to detect the activities are previously trained offline with data collected by the application running in data capture mode only.

Fig. 9 details the AwarNS modules used by the TUG Test app, along with some application screenshots. It includes the Core package, and



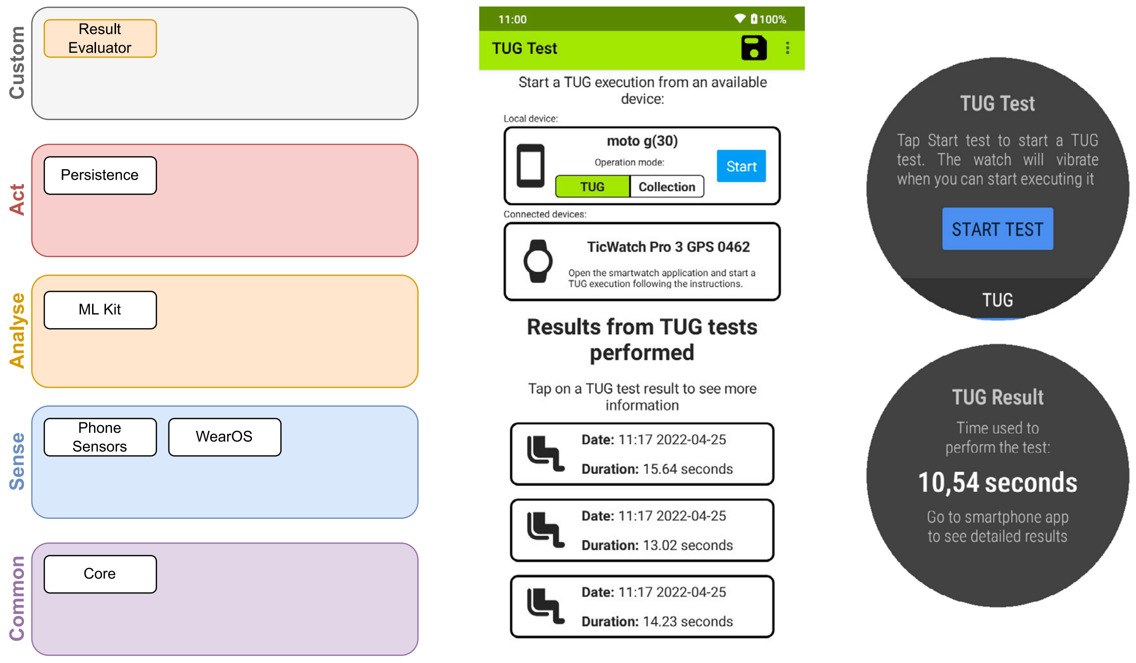
Lugar importante
Has llegado a un área de interés. Recuerda usar todas las estrategias que has aprendido

IMPORTANTE: contraste con su terapeuta si nº expediente coincide:

Nº Expediente: Usuario1

Si no es correcto pulse el botón: 'BORRAR DATOS'. Si es correcto, continúe.

Fig. 8. SyMptOMS application screenshots (centre, right) and framework modules in use (left).



Start a TUG execution from an available device:

Open the smartwatch application and start a TUG execution following the instructions.

Results from TUG tests performed

Tap on a TUG test result to see more information

Date: 11:17 2022-04-25
Duration: 15.64 seconds

Date: 11:17 2022-04-25
Duration: 13.02 seconds

Date: 11:17 2022-04-25
Duration: 14.23 seconds

Tap Start test to start a TUG test. The watch will vibrate when you can start executing it

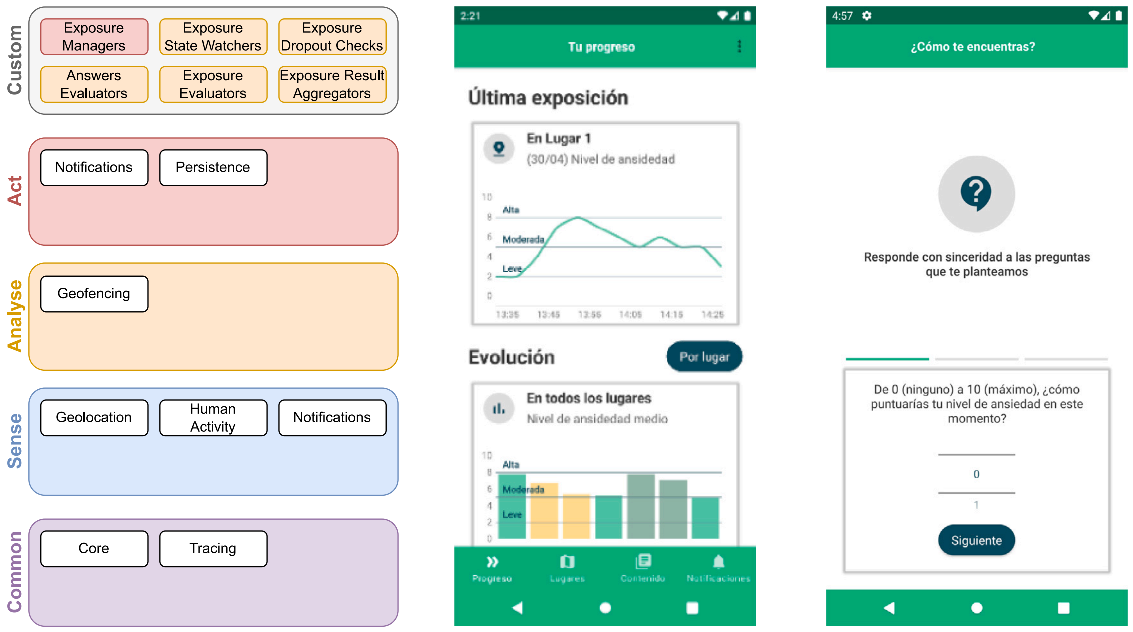
Time used to perform the test:
10,54 seconds
Go to smartphone app to see detailed results

Fig. 9. TUG Test application screenshots (centre, right) and framework modules in use (left).

the Phone Sensors, WearOS, ML Kit and Persistence modules. The Core package is used for the coordination of all built-in and custom modules, and task scheduling. The Phone Sensors module is used to collect IMU sensor samples when the smartphone acts as a sensing device. The WearOS module does the same task when the smartwatch is the sensing device. The ML Kit module uses the sensor samples as input to the previously trained machine-learning model for activity inference. The Persistence module stores sensor samples collected by the Phone Sensors and WearOS modules, and activity recognition results computed by the ML Kit module. Finally, the TUG test app also incorporates an app-specific module called Result Evaluator, which detects when the user has finished the test and computes the time spent in each inferred activity. It is an example of a Framework extension to develop ad-hoc modules. Full details of the application can be found in [65].

5.3. SyMptOMS-ET: in-vivo exposure therapy with timely assessments and rule-based EMI

The SyMptOMS-ET smartphone application represents a rewrite and extension of the original SyMptOMS app. The app was developed as a close collaboration between research teams in computer science and psychology and aims to automate the detailed process of emotional in-vivo exposure therapy in places that cause discomfort to patients undergoing psychological treatment. Similarly to the original SyMptOMS app, this application takes a set of areas of interest provided by



Responde con sinceridad a las preguntas que te planteamos

De 0 (ninguno) a 10 (máximo), ¿cómo puntuarías tu nivel de ansiedad en este momento?

Fig. 10. SyMptOMS-ET application screenshots (centre, right) and framework modules in use (left).

therapists, with which patients associate emotional discomfort or fear. Patients can inspect their exposure areas in the app via an interactive map. When they approach an area of interest (i.e. an exposure area), the application begins by providing questions to acquire baseline assessments regarding emotional discomfort and tolerance values, as well as to assess pre-exposure negative or positive beliefs of the patients. Once patients enter and as long as they stay inside the area, they regularly receive a custom question set to sample those subjective variables on a regular basis. Depending on the reported values, the app delivers specific psycho-educational content to help patients in coping with high values of emotional discomfort and difficult situations or help them maintain low values. Exposures are limited in time, determined by evaluating the trend of emotional discomfort reported regularly or by a maximum time set by psychologists. These rule-based evaluations, established by psychologists, are performed after the patient answered a question set. During exposure, the application not only reacts to changes in reported subjective measurements but also changes in the patient's relative proximity to the exposure area. For example, if a patient leaves an exposure area early, the application delivers specific content trying to convince the patient to return and continue with the exposure. It also asks for feedback in case of total abandonment, so that the therapist is informed of this circumstance. Shortly after finishing an exposure, patients are given a different set of questions to assess their behaviour during the exposure. Patients can also consult aggregated and detailed data on the evolution of their emotional discomfort in the areas where they have had exposures. The SyMptOMS-ET app has been experimentally tested, evaluated and validated for use by a committee of independent psychologists, and a clinical study is currently under recruitment for clinical validation.

Fig. 10 details the AwarNS modules used by the SyMptOMS-ET app, along with some application screenshots. It uses the Core package and Tracing module from the common category, along with the Geolocation, Human Activity, Geofencing, Notifications and Persistence modules. As usual, the Core package is used for module coordination and task scheduling. The Tracing module is used to *decorate* with a logging mechanism for the tasks conforming to the critical background execution path. An adapter, as part of the module implementation, uploads the collected traces to a remote server for its analysis or debugging purposes. Yet, the Tracing module discards logging the result of tasks that output sensitive information. The Geolocation module is used to acquire the patient's location every minute, while in movement and the vicinity of an area of interest, and every 15 min while the device is stationary and not in an area of interest to reduce battery consumption. The Human Activity module is used to detect changes in the physical activity of patients, i.e. being in motion or stationary. When no movement is detected, the Geolocation module is signalled to reduce the location acquisition sampling rate. The Geofencing module is used to detect when a patient is approaching an area of interest, in which case the Geolocation module is signalled to increase the location acquisition sampling rate. Additionally, a geolocation filter is used to restrict geolocation storage only to points acquired in the vicinity of areas of interest. The Notifications module is used to send confirmation questions, question sets, and feedback requests and acquire the answers, and deliver psycho-educational content to the patient. The Persistence module is used to locally store and remotely synchronise collected data, i.e. geolocation points, changes in exposure status, and patient answers and reactions to notifications. All this actively and passively sensed data can be consulted later by the therapists using a companion Web application.

Apart from the Framework modules, the SyMptOMS-ET app also comes with a set of specific modules, namely the Exposure Managers, Exposure State Watchers, Exposure Dropout Checks, Answers Evaluators, Exposure Evaluators and Exposure Result Aggregators modules. The Exposure Managers module contains tasks to act on the internal exposure states, to control their start and end. The tasks included in the Exposure State Watchers module observe the events emitted by the Geofencing module when the exposure is in progress, to report exits and re-entries to the exposure area. The Exposure Dropout Checkers module builds on the previous one to decide when it is necessary to ask for feedback on exposure escapes, i.e. dropouts. The Answers Evaluators module contains tasks to evaluate patients' answers during exposure at specific times using rules established by psychologists. Similarly, the tasks in the Exposure Evaluations module assess whether tailored psycho-educational content is delivered right after the end of an exposure, based on the patient's outcomes during the exposure. Lastly, the Exposure Result Aggregators include tasks for compacting reported post-exposure answers and calculating overall and place-specific emotional discomfort trends across all exposures.

6. Qualitative comparison with existing solutions

In this section, we qualitatively analyse a representative selection of existing generic, adaptable solutions that aim to simplify the development, maintenance and cost of context-aware (mHealth) apps and compare them to the AwarNS Framework. Our selection is based on a 2020 systematic review on mobile and wearable sensing frameworks for mHealth studies and applications [41], extended with an exploratory literature search (to include more recent studies) and further validated using a snowball analysis (2 levels deep) on references of the identified set. We limited the included solutions to those meeting the following criteria: (1) general purpose or health-specific, not targeted at a concrete use case or health condition; (2) running on, at least, Android smartphones; (3) with, at least, active sensing capabilities; (4) whose validity has been proved by additional studies applied to, at least, one specific health condition; and (5) currently in-service or with activity in the last five years (2017–2022). Applying these criteria, we obtained 12 related solutions: Open Data Kit (ODK) [42], AWARE [43], mCerebrum [44], the CARP Mobile Sensing (CAMS) framework [45], the Personal Analysis COmpanion (PACO) [46], Sensus [47], RADAR-base [48], BEIWE [49], the LAMP Platform [50], CommCare [51], movisensXS [52] and mEMA [53]. Below, we detail our findings, grouping the identified solutions into software frameworks, and open-source and commercial platforms. Next, the comparison criteria are introduced. Then, we compare the identified solutions to the AwarNS Framework. We focus on the re-usable functionality offered by these solutions, examine their main features and study how they adhere to the design principles exposed in Section 3. Finally, we discuss limitations.

6.1. Related solutions

In the group of software framework solutions, which offer reusable, common functionality for developers to implement mHealth apps, *ODK [42]* can be considered the precursor of modern smartphone-based data collection frameworks. It has been used in PTSD screening [66], prenatal anxiety and depression screening [67]. It comes with a web dashboard to design forms that allow including contextual data acquisition controls (geolocation, pictures, videos, etc.), which are tied to data forms for manual entry. An ODK-powered app downloads the form definition from the server, which is responsible for facilitating communication between the app and the dashboard. The dashboard is used to display the data submitted through the app's forms. *AWARE [43]* is a mature and widely used framework, which has been used in behaviour phenotyping [68], binge drinking detection [69], and depression and anxiety symptoms prediction [70]. Like ODK, AWARE comes with a dashboard, a server, and a client mobile app. The framework can be used through the ready-to-use AWARE application or integrated into an existing application. It follows a monolithic plugin-based architecture, where most of the data sources are part of the monolith, but others can be incorporated through plugins. Support for data sources is diverse. Data sources are self-contained, including data acquisition, storage, synchronisation and history data access of a single data type. *mCerebrum [44]* has been used in stress management interventions [71]. In terms of design principles, mCerebrum is the closest framework to AwarNS, but conceptually it is very different: each mCerebrum module is implemented as a separate app, whereby a core app, called DataKit, serves as central information storage for all other apps, which use inter-process communication (IPC) for storing and querying data via the DataKit API. A full installation of mCerebrum involves over 20 framework apps, even though partial installations are possible. *CAMS [45]* has been used in behavioural activation [72]. The CAMS framework is the closest in terms of architectural decisions to AwarNS. It follows a modular architecture, where the developer can decide which packages to include in each application. Unlike previous frameworks, it follows a reactive programming paradigm to define simple interactions between the different framework components (i.e., sensor data collection, EMAs, etc.). The framework includes many sensing packages (modules). CAMS features data filtering, transformation and anonymisation of individual records before persistence. It features centralised data storage, but each module implements its own mechanism for data collection.

In the group of open-source platform solutions, which are aimed at lab technicians and offer configurable (ready-to-use) tools, *PACO [46]* was born as an internal Google experiment for experience sampling. It has been used in sadness and boredom screening [73]. It is limited to capturing users' answers, which can be enriched with contextual data (similar to ODK). EMAs are delivered using time-based triggers only. A server and a web interface accompany the PACO app, to remotely visualise the captured data. All data is automatically uploaded, without the possibility of transforming it beforehand. *Sensus [47]* has been used to test clinical models of depression, social anxiety, state affect and social isolation [74]. It is a platform aimed at large crowdsensing studies. It features both time-based and sensor-based EMAs and treats surveys and sensor probing independently. Collected data is encrypted and anonymised prior to server upload, but cannot be further transformed before this happens. The studies are configured from the same mobile app, which supports two functioning modes: researcher and participant. *RADAR-base [48]* has been used to monitor behavioural changes [75] and for depression and epilepsy screening [76]. It is a platform that consists of two mobile apps, for active and passive sensing respectively, a web management tool and a separate data visualisation dashboard. It features wearable support for passive sensing and allows third-party app integrations. It focuses on server-side storage, which has been designed for scalability and to comply with data storage regulations. *BEIWE [49]* has been used in medication adherence and behaviour phenotyping [77]. It is an advanced mobile data collection platform focused on digital phenotyping. Similarly to the previous platforms, BEIWE allows, to a certain extent, to configure when EMAs should be delivered to the user and allows to passively collect data from smartphone data sources. The data is encrypted and uploaded to a remote server, and only then can the data be filtered and transformed. Similarly, reacting to the collected data is not considered, limiting its use for EMI. Finally, *LAMP Platform [50]* has been used for depression and anxiety screening [78]. Initially based on BEIWE, the LAMP Platform extended BEIWE's data collection capabilities with in-app educational content and just-in-time interventions (EMIs) driven by server-side data analysis. Yet, it shares the same data collection and upload limitations as BEIWE: although the collected data is encrypted end-to-end, data cannot be altered or filtered before being stored. Moreover, LAMP-based EMIs require the phone to always be connected to operate, limiting their use in areas where network coverage is unreliable or non-existent.

Lastly, in the group of commercial solutions, *CommCare* [51] has been used in the treatment of PTSD [79] and allows form-based data collection. Like the rest of the solutions in this group, its mobile app is configured through a no-code web interface, which allows specifying basic logic for determining which form controls or screens are displayed on user input or action. *movisensXS* [52] has been used in the treatment of bipolar disorder [80], coping with hearing voices [81] and behaviour change for Anorexia Nervosa [82], while *mEMA* [53] has been used in suicide prevention [83] and binge drinking prevention [84]. Both solutions support passive data collection but are form-centric. They allow the delivery of EMIs based on rules defined through a web-based assistant. However, no custom code is allowed. Collected data is uploaded to a server using secure channels but data analysis is server-side only.

6.2. Comparison criteria

Performance-wise, AwarNS is mainly tied to the performance of the underlying scheduling solution (NTD), hence we refer to [20] for battery consumption and data completeness figures and focus here on qualitative aspects. Our comparison builds on Kumar et al.'s work [41], expanding it with additional criteria to address all of the design principles outlined in Section 3, and validating it through an analysis of criteria used in the set of related solutions. Concretely, Kumar et al. focused their analysis on reusable features, where our more holistic approach also included versatility, reliability and testability features. As a result, and in line with the literature, the comparison dimensions of background behaviour", "Single app", "Modular design" and "Automated testing" were added to our analysis. Other original dimensions in [41] were merged due to space limitations, and some were renamed to indicate a change in scope. The impact of each dimension on its related principle(s) is detailed below.

The *reusability (RU)* principle enforces the presence of reusable features in mHealth applications. Here, we identify "Active data collection" (ADC), "Passive data collection" (PDC), "Assessment delivery" (AD), "Intervention delivery" (ID), "Health-specific features" (HSF), "No-code app" (NCA), "Web dashboard" (WD), "Server sync" (SS) and "On-device analysis support" (ODAS). Solutions presenting ADC and PDC include reusable data collection and representation models, respectively for user- and device-provided information (i.e. built-in sensors, wearables and context sampling). Solutions covering AD and ID provide reusable features for assessment and intervention delivery based on ADC, PDC or the analysis of the two. Solutions with HSF provide some reusable well-known tests – mainly cognitive or psychometric – and/or health domain-specific data analysis models. Solutions featuring an NCA provide a quick reusable way to conduct a research study via an already-made generic configurable app. Similarly, solutions offering a WD provide a reusable way to manage study participants and visualise reports. Solutions offering some degree of SS provide specific reusable data transformation and communication adapters for remote upload and synchronisation. Lastly, solutions offering ODAS provide reusable mechanisms to execute arbitrary code on users' devices, able to process ADC and PDC for quality assurance, data anonymisation, and provide AD and ID right from the phone.

The *versatility (VE)* principle is related to the "Server sync" (SS), "On-device analysis support" (ODAS), "Background behaviour" (BB), "Single app" (SA) and "Modular design" (MD) dimensions. Solutions with server-agnostic SS are more versatile because they allow more varied integrations. Solutions supporting ODAS allow more diverse use cases by enabling offline data analysis workflows. Solutions featuring BB show more versatility by enabling automated task scheduling and execution – for sensing, analysis or acting tasks – whether the main app screen is open or not. Solutions composed of multiple apps (non-SA) are more versatile since the patient only has to install the necessary features. Similarly, solutions that follow MD allow developers to exclude unnecessary features from the application package, thus limiting what the patient must install too.

The *reliability (RE)* principle is represented by "On-device analysis support" (ODAS), "Background behaviour" (BB), "Single app" (SA), "Automated testing" (AT), "Maturity" (MT) and "Proven efficacy" (PE) dimensions. ODAS influences reliability by offering consistent behaviours both in online and low-connectivity environments. The way in which the BB is implemented also affects reliability as demonstrated in [20]. Solutions composed of more than one SA have more moving pieces and hence are more prone to suffer from reliability issues. Solutions tested using AT are less prone to issues during actual usage and hence potentially more reliable. The same applies to more mature (MT) solutions and those solutions whose efficacy has been proven (PE) by conducting additional applicability and feasibility studies to concrete health conditions.

The *privacy (PR)* principle is related to the "Server sync" and "On-device analysis support" dimensions of each solution. Making the first one a requirement hurts privacy, by not allowing captured data to just stay on the patient's device. The same applies to the latter when the captured data cannot be altered prior to its storage and/or remote upload.

Lastly, the *testability (TE)* principle is present in the "Automated testing" and "Proven efficacy" dimensions. Solutions whose quality is assured by a battery of automated tests are testable by definition. The same applies to those solutions whose feasibility and applicability to specific use cases were proven in additional studies.

6.3. Comparing AwarNS to existing solutions

Table 2 lists the selected solutions along with usage/health scenarios reported in the literature. For the analysis, we first extracted the strengths and weaknesses of each solution based on published articles and available technical documentation. Next, for those solutions that work in the background, we analysed their code base to assess if they assure reliable background task execution, according to our previously published guidelines [20]. In the case of commercial platforms, only publicly available information on the website could be analysed, due to restricted access to the code base.

The main conclusions to be drawn from the comparison exercise, concisely summarised in Table 2, are as follows:

- All solutions are compatible with Android and half are also with iOS. AwarNS currently does not support iOS due to iOS' more restricted access to sensing capabilities and background scheduling.
- All solutions (including AwarNS) allow data collection directly from the user, i.e. active data collection. ODK is the only tool which does not allow scheduling such data collection requests.
- All solutions (including AwarNS), except for ODK, PACO and CommCare, collect data from sensors in an automated way, i.e. passive data collection.
- All solutions can be used for assessment to some degree, although some show some limitations. ODK does not allow performing assessment requests based on arbitrary triggers (i.e. time, location or any other detected change), and PACO only supports time-based triggers.
- Only mCerebrum, LAMP, movisensXS, mEMA and AwarNS have support for the delivery of timely interventions based on time-based and data-based triggers. Here, open-source framework solutions (mCerebrum and AwarNS) have the advantage of allowing the implementation of server-agnostic triggers, although mCerebrum requires implementing an app from scratch for that purpose.
- Only mCerebrum, LAMP, movisensXS and mEMA provide (some) predefined health-specific functionalities, such as default psychometric and/or cognitive tests, etc. AwarNS does not offer such pre-cooked health or disorder-specific functionalities but instead offers the basic building blocks that allow developers to focus on the details of each concrete health use case, such as the logic of health decisions based on collected or previously analysed data.
- All solutions but AwarNS and CAMS, offer an out-of-the-box tool for researchers to conduct studies. These two exceptions ("No-code app" column in Table 2) come with application demonstrators to test the features offered, yet no ready-to-use app is available. Instead, both offer a flexible software framework, with support for common functionality and features, which developers can re-use and build upon to obtain custom solutions, suitable for a wide variety of use cases.
- All platform solutions, along with ODK, must be paired with a server to work. AWARE makes this optional, but, when needed, server sync is not server-agnostic, i.e., it must work with the AWARE server. AwarNS, mCerebrum and CAMS are the only server-agnostic solutions; AwarNS is the only solution that offers fine-grained control over what data is stored and synced.

Table 2
Comparing the AwarNS Framework with existing solutions.

Design principles				*RU*	*RU*	*RU*	*RU*	*RU*	*RU*	*RU*	*RU, VE, PR*	*RU, VE, RL, PR*	*VE, RL*	*VE, RL*	*VE*	*RL, TE*	*RL*	*RL, TE*
Category	Nature	Name	Supported platforms[a]	Active data collection	Passive data collection	Assessment delivery	Intervention delivery	Health-specific features	No-code app	Web dashboard	Server sync	On-device analysis support	Background behaviour	Single app	Modular design	Automated testing	Maturity	Proven efficacy
Framework	Open-source	AwarNS	A	✓	✓	✓	✓	✗[b]	✗[c]	✗	Optional, server-agnostic, fine-grained	Complete	Opportunistic	✓	✓	✓	Recent	[63–65]
		ODK(-X) [42]	A	(✓)	✗	(✓)	✗	✗	✓	✓	Required	None	None	✓	✗	✓	Established	[66,67]
		AWARE [43]	A, I[d]	✓	✓	✓	✗	✗	✓	✓	Optional, AWARE-server only	Limited (cannot transform data before storage)	Continuous	✓	(✓)	✗	Established	[68–70]
		mCerebrum [44]	A	✓	✓	✓	✓	✓	✓	Under development	Optional, server-agnostic	Complete	Continuous	✗	✓	✗	Intermediate	[71]
		CAMS [45]	A, I	✓	✓	✓	✗	✗	✗[c]	✗	Optional, server-agnostic	Limited (transform data only)	None	✓	✓	✓	Recent	[72]
Platform	Open-source	PACO [46]	A, I	✓	✗	(✓)	✗	✗	✓	✓	Required	None	None	✓	✗	✓	Intermediate	[73]
		Sensus [47]	A, I	✓	✓	✓	✗	✗	✓	✗	Required	None	None	✓	✗	✓	Established	[74]
		RADAR-base [48]	A	✓	✓	✓	✗	✗	✓	✓	Required	None	OS-driven[e]	✗	✗	✓	Established	[75,76]
		BEIWE [49]	A, I	✓	✓	✓	✗	✗	✓	✓	Required	None	Continuous	✓	✗	✗	Intermediate	[77]
		LAMP [50]	A, I	✓	✓	✓	✓	✓	✓	✓	Required	None	Continuous	✓	✗	✗	Intermediate	[78]
	Commercial	CommCare [51]	A, WB	✓	✗	✓	✗	✗	✓	✓	Required	None	N/D[f]	✓	✗	N/D[f]	Established	[79]
		movisensXS [52]	A	✓	✓	✓	✓	✓	✓	✓	Required	None	N/D[f]	✓	✗	N/D[f]	Established	[80–82]
		mEMA [53]	A, I	✓	✓	✓	✓	✓	✓	✓	Required	None	N/D[f]	✓	✗	N/D[f]	Established	[83,84]

[a] A = Android; I = iOS; WB = Web Browser.
[b] Allows for easy implementation of health decisions as simple functions taking collected and/or analysed data as input or using custom machine learning models.
[c] Feature demo available.
[d] Not all features are available.
[e] Uses system-wide inexact, non-deterministic, delayed alarms.
[f] No data. Source code not available.

- AwarNS is the only solution that offers complete freedom to run any code right after data capture (and before it is stored), a feature that can be used to filter and transform data or implement on-device just-in-time analyses for momentary interventions. AWARE and CAMS offer limited support for just-collected data analysis. In AWARE, developers can implement plugins to perform actions on top of the acquired data, but the hard coupling between data capture, storage and upload prevents these plugins from avoiding the upload of certain sensitive data before it is transformed. In CAMS, post-capture code execution is limited to performing basic data transformation and filtering before data storage and uploading. Conceptually, mCerebrum also offers full flexibility but forces the developer to implement a dedicated app for that purpose.
- AwarNS is the only solution to offer opportunistic, alarm-based, background scheduling of tasks. AWARE, mCerebrum, BEIWE and LAMP also function in the background; however, they use a continuous service to gather data while the application view is not open or in the background, a method that has its disadvantages, i.e., it is not reliable because the OS regularly kills long-lived services and it is more battery intensive [20]. RADAR-base relies on alarms for task scheduling too but uses the inexact type, which raises based on OS convenience, thus making data sampling tasks unreliable [20]. In CAMS, all sensing modules but the location module require the main application screen to be open to work, which may lead to missing data if the user forgets to open the app or inadvertently closes it. Furthermore, it increases battery consumption and overall phone resource usage and is arguably more intrusive than triggering the app to run in the background for periodic short-lived sensing tasks. Sensus works in a similar way, where all the data collection features are tied to user interface components that must be disabled to implement custom interfaces [45].
- AWARE and mCerebrum partly follow a modular design, while AwarNS and CAMS follow a completely modular design. AWARE exposes a monolithic architecture, with support for external plugins to extend its core functionality. At the other end of the spectrum, mCerebrum encapsulates each module in a separate, independent app, which means it is not self-contained and requires multiple apps to work. Installation is cumbersome, and technically, the effort to successfully configure a series of apps to consistently run in the background on recent Android versions is not negligible [20,21]. Added to this, keeping users interested in individual applications is already difficult [85], therefore, having multiple apps to install, configure and interact with can be a decisive factor in the abandonment of the end users.
- All open-source solutions, except AWARE, mCerebrum, BEIWE and LAMP, include automated tests along with source code, thus increasing trust that the solutions will work as expected in the conceived usage scenarios and reducing developer onboarding friction by exposing how the individual software components work internally in case any adjustments or extensions must be implemented. Close-sourced solutions could not be analysed in this regard.

6.4. Choosing a solution based on intended use and limitations

In summary, the result of this comparison shows that the AwarNS framework stands out when context-based reactive features are required, as is the case for advanced mobile health interventions in, but not limited to, behavioural, mental health and physiological health monitoring. mCerebrum stands as a strong competitor to AwarNS, but AwarNS outperforms mCerebrum in development and deployment simplicity, background behaviour and on-device analysis, which are key features for just-in-time interventions as highlighted by [41]. Nevertheless, in less demanding usage scenarios, other solutions may be a better fit. For example, for mHealth apps that only require active data collection, possibly in combination with passive data collection tied to the actively collected data, any of the commercial platforms (i.e., CommCare, movisensXS and mEMA), ODK, PACO or Sensus are probably a better option due to the availability of ready-to-use components for manual input. For applications mainly involving the collection of passive data, visualised and exported quickly (i.e., via a web dashboard), and requiring none or almost no custom functionality, RADAR-base, BEIWE, LAMP or AWARE with server deployment and the default client app are a good option. Apps that are centred on the delivery of well-established health tests or data analyses should particularly consider LAMP, movisensXS, mEMA or mCerebrum, as these have some pre-defined health-specific artefacts available. Studies or applications in which it is not necessary to monitor, evaluate or intervene in the patient's behaviour on a regular basis, i.e. context-independent mHealth apps such as patient management apps, treatment diaries, etc., should simply consider standard reusable user interface component solutions. The AwarNS Framework is the tool of choice for mHealth apps that require reliable passive and active data collection, possibly involving local (on-phone) analysis of the collected data to provide timely interventions, and/or have high expectations to bring them into clinical practice.

7. Conclusion

In this article, we presented the AwarNS Framework, developed in the context of health and mental health. It implements the sense-analyse-act paradigm, and is aimed at easing the development of mobile applications that regularly sample the users and their environment (sense), analyse this data on-device or server-side (analyse), and perform reactive interventions based on this analysis (act). The framework is based on the design principles of reusability, reliability, privacy, versatility and testability. It follows a completely modular architecture, where a core package provides reliable, background task scheduling, and on top of this, optional modules provide specific sensing, analysis or intervention tasks. The framework comes with built-in modules to facilitate the sensing tasks of passively capturing the phone's location, scanning nearby Wi-Fi and Bluetooth devices, detecting physical activity changes, collecting samples from smartphone's and smartwatch's sensors, sampling the phone's battery level and actively requesting users' input through various pre-defined forms (e.g. confirmation, multiple choice, free answers). The framework also comes with modules to perform geofencing analysis over the captured locations, and run machine-learning models on top of the collected samples for pattern recognition and prediction. It includes a module to locally persist, export and remotely synchronise the information generated by the framework. It comes with a notifications module that facilitates the delivery of assessments and interventions. In addition, it includes a tracing module that facilitates debugging and error reporting in complex task workflows. Last but not least, it allows the developers to implement their own sensing, analysis, acting and common features, i.e., the framework is completely extensible.

AwarNS stands out for working transparently in the background, allowing to perform complex custom data analysis, data filtering and transformation tasks right on the phone, even offline, acquiring both active and passive data and delivering both assessments and interventions. These can be triggered time-based, or based on the collected data or the results of the conducted analyses. AwarNS is not an out-of-the-box or no-code solution, i.e., it does not come with a ready-made and (limitedly) configurable mobile application and Web dashboard. Instead, it is a software framework that provides developers with the basic building blocks to develop sense-analyse-act context-aware reactive (mHealth) mobile applications.

AwarNS has been used to implement a variety of mobile applications applied in clinical practice, addressing different health and mental health assessments and interventions. Among them, presented in this article, a mobile application that applies geofencing based on the smartphone's GPS sensor, to deliver psycho-educational notifications when the patients enter or leave an area of interest (i.e. panic disorder, agoraphobia, gambling disorder), an application that instruments the TUG test using smartwatch IMU sensors, which show promising results when compared to manual tests, and an application that allows detailed guided exposure therapy in places that provoke emotional discomfort.

As future work, we plan to expand the framework's features, both in sensing (i.e. support additional sensors), analysis (i.e. built-in vs external analysis) and acting (i.e. novel intervention modules). We will also evaluate the possibility of extracting and publishing, as new independent modules, concrete health features from the implemented and future applications. All are possible due to the extensibility of the AwarNS framework. In this sense, we also plan to continue maintaining and improving the current functionality of the framework, both at the core and in the different modules. Application-wise, we foresee the use of AwarNS as a basis of a variety of modern mobile health solutions, which sense the users and their environment, provide custom analysis to yield (mental) health markers, and perform just-in-time interventions based on the performed analysis. Nevertheless, AwarNS is a versatile framework, not limited to health. We foresee applications in other areas where the sense-analyse-act paradigm is applicable, not only to track people (e.g., to conduct behaviour studies, passive crowdsourcing, etc.), but also to monitor and activate objects with embedded or attached Android-based sensing devices. For example, in fleet management, asset tracking or B2C services; scenarios that involve observing and reacting to changes in the status of certain assets (e.g. goods) throughout their lifecycle.

Software access and code availability

The AwarNS framework is completely open for anyone to use. It has been published both to GitHub[1] and Zenodo [37]. The code repository contains extensive documentation on the use and extension of the framework, and a demo application, to immediately test the features that the framework offers.

CRediT authorship contribution statement

Alberto González-Pérez: Conceptualization, Methodology, Software, Formal analysis, Investigation, Writing – original draft, Visualization. **Miguel Matey-Sanz:** Software, Validation, Writing – review & editing, Visualization. **Carlos Granell:** Conceptualization, Methodology, Validation, Resources,Writing – review & editing, Supervision, Project administration, Funding acquisition. **Laura Díaz-Sanahuja:** Conceptualization, Resources,Writing – review & editing. **Juana Bretón-López:** Conceptualization, Resources, Writing – review & editing, Supervision. **Sven Casteleyn:** Conceptualization, Methodology, Validation, Resources,Writing – review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

Alberto González-Pérez and Miguel Matey-Sanz are funded by the Spanish Ministry of Universities [grants FPU17/03832 and FPU19/05352]. This study was supported by grant PID2020-120250RB-I00 (SyMptOMS-ET) funded by MCIN/AEI/10.13039/501100011033.

1 https://github.com/GeoTecINIT/awarns-framework

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.jbi.2023.104359.

References

[1] Smartphone users 2026, 2022, https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide. (Accessed 21 June 2022).
[2] U.S. and world population clock, 2022, https://www.census.gov/popclock/world. (Accessed 21 June 2022).
[3] S. Jusoh, A survey on trend, opportunities and challenges of mHealth apps, Int. J. Interact. Mob. Technol. 11 (6) (2017) 73–85, http://dx.doi.org/10.3991/ijim.v11i6.7265.
[4] L. Marzano, A. Bardill, B. Fields, K. Herd, D. Veale, N. Grey, P. Moran, The application of mhealth to mental health: opportunities and challenges, Lancet Psychiatry 2 (10) (2015) 942–948, http://dx.doi.org/10.1016/S2215-0366(15)00268-0.
[5] A.E. Kazdin, Technology-based interventions and reducing the burdens of mental illness: Perspectives and comments on the special series, Cogn. Behav. Pract. 22 (2015) 359–366, http://dx.doi.org/10.1016/j.cbpra.2015.04.004.
[6] J. Li, A. Brar, The use and impact of digital technologies for and on the mental health and wellbeing of indigenous people: a systematic review of empirical studies, Comput. Hum. Behav. 126 (2022) 106988, http://dx.doi.org/10.1016/S2215-0366(15)00268-0.
[7] K.K. Weisel, L.M. Fuhrmann, M. Berking, H. Baumeister, P. Cuijpers, D.D. Ebert, Standalone smartphone apps for mental health—a systematic review and meta-analysis, npj Digit. Med. 2 (1) (2019) 1–10, http://dx.doi.org/10.1038/s41746-019-0188-8.
[8] Covid-19 growth in medical app downloads by country 2020, 2020, https://www.statista.com/statistics/1181413/medical-app-downloads-growth-during-covid-pandemic-by-country/. (Accessed 21 June 2022).
[9] S.J. Iribarren, K. Cato, L. Falzon, P.W. Stone, What is the economic evidence for mHealth? A systematic review of economic evaluations of mHealth solutions, PLoS One 12 (2) (2017) e0170581, http://dx.doi.org/10.1371/journal.pone.0170581.
[10] J. Kim, D. Marcusson-Clavertz, K. Yoshiuchi, J.M. Smyth, Potential benefits of integrating ecological momentary assessment data into mhealth care systems, BioPsychoSoc. Med. 13 (1) (2019) 1–6, http://dx.doi.org/10.1186/s13030-019-0160-5.
[11] Top funded digital health technologies 2020, 2020, https://www.statista.com/statistics/736163/top-funded-health-it-technologies-worldwide/. (Accessed 21 June 2022).
[12] J. Torous, J. Nicholas, M.E. Larsen, J. Firth, H. Christensen, Clinical review of user engagement with mental health smartphone apps: evidence, theory and improvements, Evid.-Based Ment. Health 21 (3) (2018) 116–119, http://dx.doi.org/10.1136/eb-2018-102891.
[13] K. Huckvale, J. Nicholas, J. Torous, M.E. Larsen, Smartphone apps for the treatment of mental health conditions: status and considerations, Curr. Opin. Psychol. 36 (2020) 65–70, http://dx.doi.org/10.1016/j.copsyc.2020.04.008.
[14] A.J. Siegler, J. Knox, J.A. Bauermeister, J. Golinkoff, L. Hightow-Weidman, H. Scott, Mobile app development in health research: pitfalls and solutions, mHealth 7 (2021) http://dx.doi.org/10.21037/mhealth-19-263.
[15] D.D. Luxton, R.A. McCann, N.E. Bush, M.C. Mishkind, G.M. Reger, mHealth for mental health: Integrating smartphone technology in behavioral healthcare., Prof. Psychol. Res. Pract. 42 (6) (2011) 505, http://dx.doi.org/10.1037/a0024485.
[16] J. Torous, S. Bucci, I.H. Bell, L.V. Kessing, M. Faurholt-Jepsen, P. Whelan, A.F. Carvalho, M. Keshavan, J. Linardon, J. Firth, The growing field of digital psychiatry: current evidence and the future of apps, social media, chatbots, and virtual reality, World Psychiatry 20 (3) (2021) 318–335, http://dx.doi.org/10.1002/wps.20883.
[17] I. Miralles, C. Granell, L. Díaz-Sanahuja, W. Van Woensel, J. Bretón-López, A. Mira, D. Castilla, S. Casteleyn, et al., Smartphone apps for the treatment of mental disorders: systematic review, JMIR mHealth uHealth 8 (4) (2020) e14897, http://dx.doi.org/10.2196/14897.
[18] D. Colombo, J. Fernández-Álvarez, A. Patané, M. Semonella, M. Kwiatkowska, A. García-Palacios, P. Cipresso, G. Riva, C. Botella, Current state and future directions of technology-based ecological momentary assessment and intervention for major depressive disorder: a systematic review, J. Clin. Med. 8 (4) (2019) 465, http://dx.doi.org/10.3390/jcm8040465.
[19] C. Baxter, J.-A. Carroll, B. Keogh, C. Vandelanotte, Assessment of mobile health apps using built-in smartphone sensors for diagnosis and treatment: Systematic survey of apps listed in international curated health app libraries, JMIR mHealth uHealth 8 (2) (2020) e16741, http://dx.doi.org/10.2196/16741.
[20] A. González-Pérez, M. Matey-Sanz, C. Granell, S. Casteleyn, Using mobile devices as scientific measurement instruments: reliable android task scheduling, Pervasive Mob. Comput. 81 (2022) 101550, http://dx.doi.org/10.1016/j.pmcj.2022.101550.

[21] S. Bähr, G.-C. Haas, F. Keusch, F. Kreuter, M. Trappmann, Missing data and other measurement quality issues in mobile geolocation sensor data, Soc. Sci. Comput. Rev. (2021) 0894439320944118, http://dx.doi.org/10.1177/0894439320944118.

[22] A. González-Pérez, I. Miralles, C. Granell, S. Casteleyn, Technical challenges to deliver sensor-based psychological interventions using smartphones, in: Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers, 2019, pp. 915–920, http://dx.doi.org/10.1145/3341162.3346271.

[23] S. Liu, H. La, A. Willms, R.E. Rhodes, et al., A "No-Code" app design platform for mobile health research: Development and usability study, JMIR Form. Res. 6 (8) (2022) e38737, http://dx.doi.org/10.2196/38737.

[24] C. Jacob, A. Sanchez-Vazquez, C. Ivory, et al., Factors impacting clinicians' adoption of a clinical photo documentation app and its implications for clinical workflows and quality of care: qualitative case study, JMIR mHealth uHealth 8 (9) (2020) e20203, http://dx.doi.org/10.2196/20203.

[25] A. Ahmad, K. Li, C. Feng, S.M. Asim, A. Yousif, S. Ge, An empirical study of investigating mobile applications development challenges, IEEE Access 6 (2018) 17711–17728, http://dx.doi.org/10.1109/ACCESS.2018.2818724.

[26] Software developers' biggest challenges 2022, 2022, https://www.revealbi.io/whitepapers/software-developers-biggest-challenges. (Accessed 21 June 2022).

[27] K.L. Fortuna, M.C. Lohman, L.E. Gill, M.L. Bruce, S.J. Bartels, Adapting a psychosocial intervention for smartphone delivery to middle-aged and older adults with serious mental illness, Am. J. Geriatr. Psychiatry 25 (8) (2017) 819–828, http://dx.doi.org/10.1016/j.jagp.2016.12.007.

[28] S. Barnett, K. Huckvale, H. Christensen, S. Venkatesh, K. Mouzakis, R. Vasa, et al., Intelligent sensing to inform and learn (InSTIL): A scalable and governance-aware platform for universal, smartphone-based digital phenotyping for research and clinical applications, J. Med. Internet Res. 21 (11) (2019) e16399, http://dx.doi.org/10.2196/16399.

[29] M. Andrachuk, M. Marschke, C. Hings, D. Armitage, Smartphone technologies supporting community-based environmental monitoring and implementation: a systematic scoping review, Biol. Cons. 237 (2019) 430–442, http://dx.doi.org/10.1016/j.biocon.2019.07.026.

[30] A. Kuerbis, A. Mulliken, F. Muench, A.A. Moore, D. Gardner, Older adults and mobile technology: Factors that enhance and inhibit utilization in the context of behavioral health, CUNY Acad. Work. (2017) http://dx.doi.org/10.15761/MHAR.1000136.

[31] J.M. Robillard, T.L. Feng, A.B. Sporn, J.-A. Lai, C. Lo, M. Ta, R. Nadler, Availability, readability, and content of privacy policies and terms of agreements of mental health apps, Internet Interv. 17 (2019) 100243, http://dx.doi.org/10.1016/j.invent.2019.100243.

[32] M. Bauer, T. Glenn, J. Geddes, M. Gitlin, P. Grof, L.V. Kessing, S. Monteith, M. Faurholt-Jepsen, E. Severus, P.C. Whybrow, Smartphones in mental health: a critical review of background issues, current status and future concerns, Int. J. Bipolar Disord. 8 (1) (2020) 1–19, http://dx.doi.org/10.1186/s40345-019-0164-x.

[33] L. Nurgalieva, D. O'Callaghan, G. Doherty, Security and privacy of mHealth applications: a scoping review, IEEE Access 8 (2020) 104247–104268, http://dx.doi.org/10.1109/ACCESS.2020.2999934.

[34] A. van Haasteren, F. Gille, M. Fadda, E. Vayena, Development of the mHealth app trustworthiness checklist, Digit. Health 5 (2019) 2055207619886463, http://dx.doi.org/10.1177/2055207619886463.

[35] D.A. Adler, F. Wang, D.C. Mohr, D. Estrin, C. Livesey, T. Choudhury, A call for open data to develop mental health digital biomarkers, BJPsych Open 8 (2) (2022) e58, http://dx.doi.org/10.1192/bjo.2022.28.

[36] C. Granell, A. Kamilaris, A. Kotsev, F.O. Ostermann, S. Trilles, Internet of things, in: H. Guo, M.F. Goodchild, A. Annoni (Eds.), Manual of Digital Earth, Springer Singapore, Singapore, 2020, pp. 387–423, http://dx.doi.org/10.1007/978-981-32-9915-3_11.

[37] A. González, M. Matey, C. Granell, GeoTecINIT/Awarns-Framework, Zenodo, 2022, http://dx.doi.org/10.5281/zenodo.7100416.

[38] J.B. Torous, Focusing on the future of mobile mental health and smartphone interventions, Psychiatr. Serv. 69 (9) (2018) 945, http://dx.doi.org/10.1176/appi.ps.201800308.

[39] B. Aryana, L. Brewster, J.A. Nocera, Design for mobile mental health: an exploratory review, Health Technol. 9 (4) (2019) 401–424, http://dx.doi.org/10.1007/s12553-018-0271-1.

[40] L. Piwek, D.A. Ellis, A. Sally, Can programming frameworks bring smartphones into the mainstream of psychological science? Front. Psychol. 7 (2016) 1252, http://dx.doi.org/10.3389/fpsyg.2016.01252.

[41] D. Kumar, S. Jeuris, J.E. Bardram, N. Dragoni, Mobile and wearable sensing frameworks for mHealth studies and applications: a systematic review, ACM Trans. Comput. Healthc. 2 (1) (2020) 1–28, http://dx.doi.org/10.1145/3422158.

[42] P. Loola Bokonda, K. Ouazzani-Touhami, N. Souissi, Mobile data collection using open data kit, in: International Conference Europe Middle East & North Africa Information Systems and Technologies to Support Learning, Springer, 2019, pp. 543–550, http://dx.doi.org/10.1007/978-3-030-36778-7_60.

[43] D. Ferreira, V. Kostakos, A.K. Dey, AWARE: mobile context instrumentation framework, Front. ICT 2 (2015) 6, http://dx.doi.org/10.3389/fict.2015.00006.

[44] S.M. Hossain, T. Hnat, N. Saleheen, N.J. Nasrin, J. Noor, B.-J. Ho, T. Condie, M. Srivastava, S. Kumar, mCerebrum: a mobile sensing software platform for development and validation of digital biomarkers and interventions, in: Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems, 2017, pp. 1–14, http://dx.doi.org/10.1145/3131672.3131694.

[45] J.E. Bardram, The CARP mobile sensing framework–A cross-platform, reactive, programming framework and runtime environment for digital phenotyping, 2020, http://dx.doi.org/10.48550/arXiv.2006.11904, arXiv preprint arXiv:2006.11904.

[46] K.K. Baxter, A. Avrekh, B. Evans, Using experience sampling methodology to collect deep data about your users, in: Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems, 2015, pp. 2489–2490, http://dx.doi.org/10.1145/2702613.2706668.

[47] H. Xiong, Y. Huang, L.E. Barnes, M.S. Gerber, Sensus: a cross-platform, general-purpose system for mobile crowdsensing in human-subject studies, in: Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing, 2016, pp. 415–426, http://dx.doi.org/10.1145/2971648.2971711.

[48] Y. Ranjan, Z. Rashid, C. Stewart, P. Conde, M. Begale, D. Verbeeck, S. Boettcher, R. Dobson, A. Folarin, R.-C. Consortium, et al., RADAR-base: open source mobile health platform for collecting, monitoring, and analyzing data using sensors, wearables, and mobile devices, JMIR mHealth uHealth 7 (8) (2019) e11734, http://dx.doi.org/10.2196/11734.

[49] J.-P. Onnela, C. Dixon, K. Griffin, T. Jaenicke, L. Minowada, S. Esterkin, A. Siu, J. Zagorsky, E. Jones, Beiwe: a data collection platform for high-throughput digital phenotyping, J. Open Source Softw. 6 (68) (2021) 3417, http://dx.doi.org/10.21105/joss.03417.

[50] A. Vaidyam, J. Halamka, J. Torous, et al., Enabling research and clinical use of patient-generated health data (the mindLAMP platform): digital phenotyping study, JMIR mHealth uHealth 10 (1) (2022) e30557, http://dx.doi.org/10.2196/30557.

[51] CommCare by Dimagi: Data collection app, 2022, https://www.dimagi.com/commcare/. (Accessed 21 June 2022).

[52] Experience sampling - movisensXS, 2022, https://www.movisens.com/en/products/movisensXS/. (Accessed 21 June 2022).

[53] mEMA app, 2021, https://ilumivu.com/solutions/ecological-momentary-assessment-app/. (Accessed 21 June 2022).

[54] I. Miralles, C. Granell, L. Díaz-Sanahuja, W. Van Woensel, J. Bretón-López, A. Mira, D. Castilla, S. Casteleyn, Smartphone apps for the treatment of mental disorders: Systematic review, JMIR Mhealth Uhealth 8 (4) (2020) e14897, http://dx.doi.org/10.2196/14897.

[55] R.C. Martin, Agile Software Development: Principles, Patterns, and Practices, Prentice Hall PTR, 2003, http://dx.doi.org/10.5555/515230.

[56] K. Sahu, R. Srivastava, Revisiting software reliability, Data Manag. Anal. Innov. (2019) 221–235, http://dx.doi.org/10.1007/978-981-13-1402-5_17.

[57] M.T. Baldassarre, V.S. Barletta, D. Caivano, M. Scalera, Integrating security and privacy in software development, Softw. Qual. J. 28 (3) (2020) 987–1018, http://dx.doi.org/10.1007/s11219-020-09501-6.

[58] S.U. Khan, A.W. Khan, F. Khan, M.A. Khan, T.K. Whangbo, Critical success factors of component-based software outsourcing development from vendors' perspective: A systematic literature review, IEEE Access 10 (2021) 1650–1658, http://dx.doi.org/10.1109/ACCESS.2021.3138775.

[59] H. Toivakka, T. Granlund, T. Poranen, Z. Zhang, Towards RegOps: A DevOps pipeline for medical device software, in: International Conference on Product-Focused Software Process Improvement, Springer, 2021, pp. 290–306, http://dx.doi.org/10.1007/978-3-030-91452-3_20.

[60] L.S. Vailshery, Most used languages among software developers globally 2021, 2022, Statista, https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/.

[61] P. Pascacio, S. Casteleyn, J. Torres-Sospedra, E.S. Lohan, J. Nurmi, Collaborative indoor positioning systems: A systematic review, Sensors 21 (3) (2021) 1002, http://dx.doi.org/10.3390/s21031002.

[62] N.C. Jacobson, B. Feng, Digital phenotyping of generalized anxiety disorder: using artificial intelligence to accurately predict symptom severity using wearable sensors in daily life, Transl. Psychiatry 12 (1) (2022) 1–7, http://dx.doi.org/10.1038/s41398-022-02038-1.

[63] I. Miralles, C. Granell, A. García-Palacios, D. Castilla, A. González-Pérez, S. Casteleyn, J. Bretón-López, Enhancing in vivo exposure in the treatment of panic disorder and agoraphobia using location-based technologies: A case study, Clin. Case Stud. 19 (2) (2020) 145–159, http://dx.doi.org/10.1177/1534650119892900.

[64] L. Díaz-Sanahuja, I. Miralles, C. Granell, A. Mira, A. González-Pérez, S. Casteleyn, A. García-Palacios, J. Bretón-López, Client's experiences using a location-based technology ICT system during gambling treatments' crucial components: A qualitative study, Int. J. Environ. Res. Public Health 19 (7) (2022) 3769, http://dx.doi.org/10.3390/ijerph19073769.

[65] M. Matey-Sanz, A. González-Pérez, S. Casteleyn, C. Granell, Instrumented timed up and go test using inertial sensors from consumer wearable devices, in: International Conference on Artificial Intelligence in Medicine, Springer, 2022, pp. 144–154, http://dx.doi.org/10.1007/978-3-031-09342-5_14.

[66] B. Hashemi, S. Ali, R. Awaad, L. Soudi, L. Housel, S.J. Sosebee, Facilitating mental health screening of war-torn populations using mobile applications, Soc. Psychiatry Psychiatr. Epidemiol. 52 (1) (2017) 27–33, http://dx.doi.org/10.1007/s00127-016-1303-7.

[67] A. Bante, A. Mersha, Z. Zerdo, B. Wassihun, T. Yeheyis, Comorbid anxiety and depression: Prevalence and associated factors among pregnant women in Arba Minch zuria district, gamo zone, southern Ethiopia, PLoS One 16 (3) (2021) e0248331, http://dx.doi.org/10.1371/journal.pone.0248331.

[68] A. Doryab, D.K. Villalba, P. Chikersal, J.M. Dutcher, M. Tumminia, X. Liu, S. Cohen, K. Creswell, J. Mankoff, J.D. Creswell, et al., Identifying behavioral phenotypes of loneliness and social isolation with passive sensing: statistical analysis, data mining and machine learning of smartphone and fitbit data, JMIR mHealth uHealth 7 (7) (2019) e13209, http://dx.doi.org/10.2196/13209.

[69] S. Bae, D. Ferreira, B. Suffoletto, J.C. Puyana, R. Kurtz, T. Chung, A.K. Dey, Detecting drinking episodes in young adults using smartphone-based sensors, Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 1 (2) (2017) 1–36, http://dx.doi.org/10.1145/3090051.

[70] I. Moshe, Y. Terhorst, K. Opoku Asare, L.B. Sander, D. Ferreira, H. Baumeister, D.C. Mohr, L. Pulkki-Råback, Predicting symptoms of depression and anxiety using smartphone and wearable data, Front. Psychiatry 12 (2021) 625247, http://dx.doi.org/10.3389/fpsyt.2021.625247.

[71] S.L. Battalio, D.E. Conroy, W. Dempsey, P. Liao, M. Menictas, S. Murphy, I. Nahum-Shani, T. Qian, S. Kumar, B. Spring, Sense2Stop: a micro-randomized trial using wearable sensors to optimize a just-in-time-adaptive stress management intervention for smoking relapse prevention, Contemp. Clin. Trials 109 (2021) 106534, http://dx.doi.org/10.1016/j.cct.2021.106534.

[72] D.A. Rohani, A. Quemada Lopategui, N. Tuxen, M. Faurholt-Jepsen, L.V. Kessing, J.E. Bardram, MUBS: A personalized recommender system for behavioral activation in mental health, in: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, 2020, pp. 1–13, http://dx.doi.org/10.1145/3313831.3376879.

[73] C.S. Chan, W.A. Van Tilburg, E.R. Igou, C. Poon, K.Y. Tam, V.U. Wong, S. Cheung, Situational meaninglessness and state boredom: Cross-sectional and experience-sampling findings, Motiv. Emot. 42 (4) (2018) 555–565, http://dx.doi.org/10.1007/s11031-018-9693-3.

[74] P.I. Chow, K. Fua, Y. Huang, W. Bonelli, H. Xiong, L.E. Barnes, B.A. Teachman, Using mobile sensing to test clinical models of depression, social anxiety, state affect, and social isolation among college students, J. Med. Internet Res. 19 (3) (2017) e6820, http://dx.doi.org/10.2196/jmir.6820.

[75] S. Sun, A.A. Folarin, Y. Ranjan, Z. Rashid, P. Conde, C. Stewart, N. Cummins, F. Matcham, G. Dalla Costa, S. Simblett, et al., Using smartphones and wearable devices to monitor behavioral changes during COVID-19, J. Med. Internet Res. 22 (9) (2020) e19992, http://dx.doi.org/10.2196/19992.

[76] C.L. Stewart, Z. Rashid, Y. Ranjan, S. Sun, R.J. Dobson, A.A. Folarin, RADAR-base: major depressive disorder and epilepsy case studies, in: Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers, 2018, pp. 1735–1743, http://dx.doi.org/10.1145/3267305.3267540.

[77] M. Straczkiewicz, H. Wisniewski, K.W. Carlson, Z. Heidary, J. Knights, M. Keshavan, J.-P. Onnela, J. Torous, Combining digital pill and smartphone data to quantify medication adherence in an observational psychiatric pilot study, Psychiatry Res. 315 (2022) 114707, http://dx.doi.org/10.1016/j.psychres.2022.114707.

[78] J. Melcher, S. Patel, L. Scheuer, R. Hays, J. Torous, Assessing engagement features in an observational study of mental health apps in college students, Psychiatry Res. 310 (2022) 114470, http://dx.doi.org/10.1016/j.psychres.2022.114470.

[79] A.M. Bauer, S. Hodsdon, J.M. Bechtel, J.C. Fortney, Applying the principles for digital development: case study of a smartphone app to support collaborative care for rural patients with posttraumatic stress disorder or bipolar disorder, J. Med. Internet Res. 20 (6) (2018) e10048, http://dx.doi.org/10.2196/10048.

[80] E. Mühlbauer, M. Bauer, U. Ebner-Priemer, P. Ritter, H. Hill, F. Beier, N. Kleindienst, E. Severus, Effectiveness of smartphone-based ambulatory assessment (SBAA-BD) including a predicting system for upcoming episodes in the long-term treatment of patients with bipolar disorders: study protocol for a randomized controlled single-blind trial, BMC Psychiatry 18 (1) (2018) 1–9, http://dx.doi.org/10.1186/s12888-018-1929-y.

[81] I.H. Bell, S.F. Fielding-Smith, M. Hayward, S.L. Rossell, M.H. Lim, J. Farhall, N. Thomas, Smartphone-based ecological momentary assessment and intervention in a coping-focused intervention for hearing voices (SAVVy): study protocol for a pilot randomised controlled trial, Trials 19 (1) (2018) 1–13, http://dx.doi.org/10.1186/s13063-018-2607-6.

[82] D.R. Kolar, F. Hammerle, E. Jenetzky, M. Huss, Smartphone-enhanced low-threshold intervention for adolescents with Anorexia Nervosa (SELTIAN) waiting for outpatient psychotherapy: study protocol of a randomised controlled trial, BMJ Open 7 (10) (2017) e018049, http://dx.doi.org/10.1136/bmjopen-2017-018049.

[83] C. Nuij, W. van Ballegooijen, J. Ruwaard, D. De Beurs, J. Mokkenstorm, E. van Duijn, R.F. de Winter, R.C. O'Connor, J.H. Smit, H. Riper, et al., Smartphone-based safety planning and self-monitoring for suicidal patients: Rationale and study protocol of the CASPAR (Continuous assessment for suicide prevention and research) study, Internet Interv. 13 (2018) 16–23, http://dx.doi.org/10.1016/j.invent.2018.04.005.

[84] B.L. Stevenson, C.E. Blevins, E. Marsh, S. Feltus, M. Stein, A.M. Abrantes, An ecological momentary assessment of mood, coping and alcohol use among emerging adults in psychiatric treatment, Am. J. Drug Alcohol Abuse 46 (5) (2020) 651–658, http://dx.doi.org/10.1080/00952990.2020.1783672.

[85] L. Ceci, Mobile app user retention rate by category 2020, 2022, Statista, https://www.statista.com/statistics/259329/ios-and-android-app-user-retention-rate/.