IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 27, 2020, accepted August 4, 2020, date of publication August 7, 2020, date of current version August 20, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3015082

Early Evaluation of Mobile Applications' Resource Consumption and Operating Costs

JAVIER BERROCAL[1], JOSE GARCÍA-ALONSO[1], PABLO FERNANDEZ[2], ALEJANDRO PÉREZ-VEREDA[3], JUAN HERNANDEZ[1], (Associate Member, IEEE), CARLOS CANAL[3], JUAN MANUEL MURILLO[1], AND ANTONIO RUIZ-CORTES[2], (Member, IEEE)

[1]Department of Computer and Telematic Systems Engineering, University of Extremadura, 06006 Badajoz, Spain
[2]Department of Computer Languages and Systems, University of Sevilla, 41004 Sevilla, Spain
[3]Department of Computer Science, University of Malaga, 29016 Málaga, Spain

Corresponding author: Jose García-Alonso (jgaralo@unex.es)

This work was supported in part by the MCI/AEI/FEDER,UE, through Projects under Grant RTI2018-094591-B-I00 and Grant PGC2018-094905-B-I00, in part by the APOLO under Grant U.S.–1264651, in part by the HORATIO under Grant RTI2018-101204–B–C21, in part by the RCIS Research Network under Grant RED2018-102654-T, in part by the 4IE+ Project funded by the Interreg V-A España-Portugal (POCTEP) 2014–2020 Program under Grant 0499-4IE-PLUS-4-E, in part by the FEDER/Junta de Andalucia under Project UMA18-FEDERJA-180, in part by the Department of Economy, Science, and Digital Agenda of the Government of Extremadura under Grant GR18112 and Grant IB18030, and in part by the European Regional Development Fund.

ABSTRACT The explosive growth of the mobile application market in recent years has led to a large concomitant mobile software industry whose components are, in many cases, startups and small-size software providers. The success of these applications and the firms behind them depends on a subtle balance between different dimensions mainly affected by their architectural design, such as user satisfaction, resource consumption, operating costs, and timing. The present communication describes a framework with a specific set of practices for identifying the boundaries of different architectural designs —in this article we apply it to estimate both the smartphone's resource consumption and the operating costs in the cloud— and thus help in the architectural decision-making process. This will enable mobile software developers to predict at early stages which architectural design best suits their business model in accordance with the number of users and the expected use of the application and even provide an advance alert of when architectural choices will need to be reviewed, obviating the need for costly architectural re-design in further phases.

INDEX TERMS J.9 mobile applications, D.2.11 software architectures, D.2.18 software engineering process, M.2 services lifecycle.

I. INTRODUCTION

One of the most striking social changes in recent years has been the pervasive presence of mobile phones. Mobile-broadband subscriptions had been estimated to reach a penetration rate of over 120% in developed countries [1]. The implication was that this would involve a global mobile data traffic of 11.5 exabytes per month by the end of that year, with an estimated compound annual growth rate of 46% up to 2022 [2]. Consumers' habits have adapted accordingly, to the point where now major Internet sites receive most of their traffic through smartphones. Facebook, for instance, in its 2020 report states that it had 1.73 billion daily active users, and that 93% of its advertising revenues came from mobile advertising [3].

Such figures reflect the global dimension of the impact caused by smartphones, an impact which has led to the creation of a large mobile software development industry. Currently, there are more than 5.5 million applications in the leading app stores in 2020, with more than 200 billion downloads in 2019 [4]. This relatively new industry is expected to grow to over 400 billion US dollars in 2026 [5]. The rapid growth of this market has changed the way in which applications are being conceived and implemented.

This explosive development of the mobile apps market has been a challenge for the software industry. In contrast to standard software investments which focus on long-term issues such as feasibility, earned value, etc., the business

The associate editor coordinating the review of this manuscript and approving it for publication was Eyuphan Bulut.

models behind mobile applications are usually based on a very short time-to-market [6]. This allows investors to get rapid feedback about the application's profitability. Given this feedback, they can then work closely with the developers to analyse different alternatives for the ongoing development of the application. An application that becomes viral would follow a very different process than one which is merely profitable.

The analysis required to define the immediate future of a mobile application is anything but trivial. It will involve a trade-off between different closely related magnitudes [7] for improving the user experience, such as the user interface, the functionalities or the resource consumption [8], [9]. For instance, focused on the resource consumption, one is the number of users and their behaviour and engagement in the use of the application. That magnitude directly affects the consumption of resources in mobile devices and the operational costs of the cloud infrastructure and services. Traditionally, one has always wanted to maximize the number of users and their engagement. The reason is that these magnitudes in most cases determine the return on investment (monetization and retention of users). But more users and more activity per user usually lead to a greater consumption of resources in their devices, and this can cause a proportion of those users to abandon the application and other potential users to reject it [10], [11]. Alternatively, this increased resource consumption in mobile devices can be mitigated by applying architectural decisions that offload activities to the cloud, but this implies increased cloud operating costs [12] which may even exceed the benefit created by engaging a greater number of users. Both situations – user rejection and cloud operational cost overrun - could mean a step backwards in the mobile application's development plan.

Even though this kind of analysis is of crucial importance, there is little literature available about when and how it should be undertaken. Therefore, we here propose a set of practices to be integrated into the process of mobile application development with the aim of assisting with future architectural decisions. The practices involve the use of a framework that uses such magnitudes as the number of users and the activity per user to provide an early estimate of the boundaries of each architectural design. In particular, in this article those practices are focused on estimating resource consumption both in the mobile device and in the cloud of different architectural designs of social mobile applications. The goal of these practices is to provide additional guidance and help software architects to evaluate different architectural options in order to improve that dimension of the user satisfaction. This also enables investors to in turn make an early assessment of the return on investment that can be expected for different architectural decisions. In addition, the framework also provides an estimate of when and under which conditions an architecture should be re-engineered. This information could be used together with the existing architectural knowledge and expertise to better plan future evolution. To the best of our knowledge, no other work has proposed a framework for the early identification of a mobile application's architectural boundaries, allowing developers to plan when and how the architecture has to evolve depending on the users' behaviour.

The rest of this article is organized as follows. After this introduction, Sec. II presents the motivation for the work. Then Sec. III describes how the proposed practices can be included in a development process, and Sec. IV applies them for estimating both the mobile and the cloud consumption dimensions. Sections V and VI describe respectively a realistic case study and the experimental validation of the approach, measuring the actual consumption of the architecturally different variants of the application developed. Section VII discusses the most relevant related work in this field, and finally Sec. VIII presents the conclusions drawn from this study.

II. MOTIVATION

The early evaluation of software architectures has traditionally been the object of much attention from researchers in the area of software engineering. Various methodological approaches, techniques, and tools have been proposed to evaluate an architecture's formal aspects [13], their technical coherence [14], their adaptation to business requirements [15], and even the user experience [16]. The main objective of such proposals is to provide mechanisms that can help ensure the success of investments in software technologies. This is of real importance because such investments usually involve considerable financial effort and strategic planning on the part of the software's target business. Experience in the development of complex systems has shown that poor decisions made in early stages can lead to problems that will only be revealed when they are already almost irreparable [17]. So the industry has welcomed any aid to the early validation of decisions.

In the last fifteen years, the penetration of mobile technologies, smartphones in particular, has motivated the growth of the mobile application market. This market's rules are quite different from those applicable to traditional systems [18]. In the latter, software is developed for a well-known business in which users are clearly identified. The intention with the new software is to improve the services offered to users. User engagement is largely guaranteed by the business itself. In contrast, mobile applications are often born in the context of a startup with an as yet undeveloped but maybe innovative business idea behind it. Lean software development practices are then adopted by the development teams looking to shorten the time-to-market. The objective is to obtain feedback as soon as possible with a minimum investment, and then decide whether greater investment is justified and in which direction to aim [19]. Since how users will react and be engaged is almost unknown, there will be a great variability in the business plans.

Operating in a world guided by the law "the faster you deploy, the faster you get feedback" [20] often means that the product is released without being fully tuned. Indeed, fine tuning of the product is sometimes left to be done in response to the feedback provided by users. This can cause a false

impression that, in the development of mobile apps, the early evaluation of design decisions takes a back seat because anything goes. This could not be further from reality. For instance, early evaluation practices are explicitly considered as part of the lean methodology's build-measure-learn cycle so as to make development teams aware of the implications of technical decisions in terms of exploitation costs and user satisfaction, and of how the decisions made have to be constantly evaluated and reconsidered in order to increase user satisfaction. In fact, early evaluation is now as present as it has always been, but with a different focus. Instead of the focus being on addressing very large long-term investments, it is now on evaluating different future scenarios in terms of the amount of resources that have to be committed to them, and the outcome they will generate.

For example, in order to improve a dimension of the user satisfaction, the development team might evaluate the reduction of mobile devices' resource consumption by offloading some of the application's features to the cloud [21]. If the number of users remains under control, this could well be a good decision with an affordable cost. But if the number of users were to increase sharply, a startup might find that the exploitation cost of the service had become unaffordable, or the quality of service would have to be reduced to an inadmissible level. Hence, a reliable estimate of the boundaries of each architecture, depending on the number of users and their expected behaviour, is crucial to determine whether or not it is feasible. Likewise, monitoring those magnitudes would be a key to identifying when the architecture needs to evolve. There are known cases of this situation. One example is PokemonGO where the failure to make a good early evaluation led to server connection and access problems due to the heavy demand, problems that would have ruined such a major investment made by a startup [22].

To the best of our knowledge, there have been no studies focused on how to include early evaluation practices in the development processes of mobile apps. Given this lacuna, the present work provides a set of practices to be incorporated into these processes so as to support early evaluation during the preparation of each release, and provide additional help in the identification of the architectural design that best fits the requirements and the business model. Although, it can be applied to any application, this process is focused on social mobile applications (which are apps oriented to the interaction with other users and data sharing) [23]. To that end, in Sec. III we shall detail how these practices should be carried out, and discuss when they should be implemented within the processes of agile and lean software development. Measurements and evaluation need to be supported by some mathematical tools. As an example of how to estimate these boundaries, a framework is provided (Sec. IV) for estimating mobile applications' consumption of resources. This framework is evaluated and validated in Secs. V and VI.

III. INTEGRATING THE FRAMEWORK INTO A SOFTWARE DEVELOPMENT PROCESS

This section will address the activities that must be performed for the early evaluation of a mobile app's architectural boundaries. It will deal with their sequencing, the roles they each have to play, and the artifacts they might generate. To better detail this process, it was modeled using SPEM (Software and Systems Process Engineering Meta-Model) [24]. We shall also discuss how these activities integrate with agile and lean software development processes [25].

These processes have become very popular in recent years [26]. The Lean Methodology focuses on, first, reducing activities and code that are not clear or do not benefit the customer, and, second, continual learning using the customer's feedback [27], [28]. The term Agile was introduced in the 1990s to refer to flexible production systems [26]. Agile software process model concentrates on people and not technologies [29]. Agile tries to make the development process more flexible while Lean makes it more sustainable.

During the last few years, different works analysed and evaluated the usefulness of these processes to cope with the competitiveness of the mobile applications market and the restrictions of these devices. These works analyse the integration of different Agile methodologies (such as Scrum, XP, Kanban, etc.) with Lean [30]–[32].

Concretely, in this section, we discuss how the proposed activities are integrated with the Scrum development process [33] and the Lean Business model [34]. We selected Scrum as Agile methodology because it is widely used for the development of mobile applications [20], [27]. Please, note that the proposed activities do not substitute the development team's knowledge, but they provide additional guidance.

Instead of the planning activities defined by Scrum, i.e., specifying the sprint size, the team size, how to handle the sprint Backlogs or its integration with Lean (some works such as [27] defined how to perform these tasks for the development of mobile applications), Figure 1 shows how the whole process for estimating the architectural boundaries fits into a Scrum development workflow, with the Product Owner and the Scrum Team being the main roles involved. The main responsibilities of the Product Owner for increasing the stakeholder's satisfaction are to manage the relationship with the users, the backlog, the identification of the needs, and the business plan development [35]. Apart from the artifacts detailed in the figure, every task has an artifact *App Architecture* as input/output (not included in the figure for the sake of its readability). This documents the different architectural decisions considered and taken, and their impact on the user satisfaction and the business model. The process is iterative and fits the build-measure-learn feedback loop of the lean methodology. Briefly, in the first iteration, the Minimum Viable Product (MVP) [36] is defined. To that end, a full analysis of the MVP, its functionalities, the execution environment, and the predicted user behaviour is required to determine the architectural design that best fits the business model

and the user satisfaction (regarding resource consumption, for instance, or any other dimension). Each iteration results in a build of a new release of the product. Once the release has been launched, the Product Owner measures its acceptance, i.e., how users interact with it, their satisfaction, etc., and can thus learn how to improve the product to increase user satisfaction or to reduce costs. This information is stored in the Business Model artifact for use in future iterations of the process. For instance, every new release requires this information to be re-analysed in order to identify whether some parts of the system should be changed to reduce operational cost, consumption, and/or to improve the user experience.

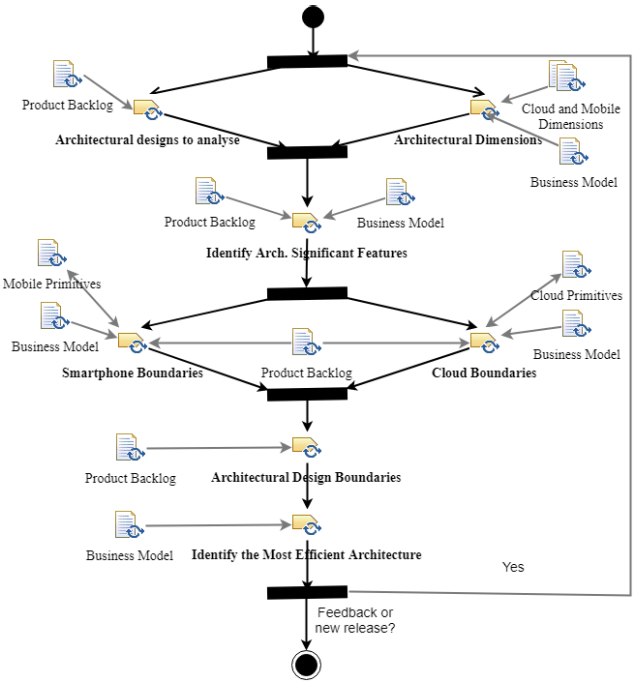


FIGURE 1. Activities for estimating the architectural boundaries.

Specifically, in the first task (*Architectural designs to analyse*), the Scrum Team reviews the Product Backlog artifact which was generated as a result of the Scrum process. It contains a list of functional requirements prioritized by their business value, from which the main features of the application will derive. Several architectural design alternatives may be identified by analysing these requirements, in particular those that will be included in the MVP. Different architectural designs can be defined depending on the applied architectural styles (which are the grouping of different responsibilities on loosely coupled layers or modules) [37], [38]. These designs may vary in how some functionalities have to be onloaded to the mobile devices or offloaded to the cloud, ranging from a pure server-centric application (where all the computation is done in the cloud, applying a client-server architectural style) to a mobile-centric one (in which it is done in the end devices, applying a peer-to-peer style) [38], or a mix of the two, with part of the functionality in the devices and part in the cloud. The architectural designs considered will be listed in the App Architecture artifact. The Scrum Team must decide which one is the most appropriate for the first release of the product.

At the same time, for the task *Architectural Dimensions*, the Product Owner studies the Business Model artifact to identify the dimensions to be estimated for. For instance, the Owner may decide to estimate the consumption, and concretely the battery and data traffic consumption, as they have a major impact on user satisfaction. In addition, if the Business Model sets constraints on the operating costs, they must also be evaluated. To assist in this decision-making task, the Cloud Dimensions and Mobile Dimensions artifacts list the different dimensions under consideration and how they can be estimated. If the Product Owner decides to estimate a dimension, or a specific resource, which is not dealt with in these artifacts, this must be noted appropriately so that the Scrum Team can later determine how it is to be estimated.

However, it would require too much effort to estimate the architectural impact of every feature of an application. Therefore, in the task *Identify Architectural Significant Features*, the Scrum Team has to determine which are the functional requirements that impact on the architecture. These will be the key requirements for the MVP, either because they are frequently used or because they underpin the essence of the application. Examples of architectural significant requirements are those that involve data acquisition from several sensors, require much processing, transfer large volumes of data, or are executed very frequently. As a result of this step, the Scrum Team will identify the most significant features for estimating the application's architectural boundaries and the parameters on which they depend (rates of use, data sizes, processing times, etc.).

Once the relevant requirements have been identified, one needs to estimate how they impact on each architectural option under consideration. To assist in this analysis, in the next section we shall present a framework that make use of a set of common primitive operations in order to estimate the application's boundaries for each design. Figure 1 shows two tasks — *Smartphone Boundaries* and *Cloud Boundaries*— for estimating the impact on the most important elements of mobile apps. They are performed in parallel by the Scrum Team according to the following steps:

1) The App Architecture and the Product Backlog artifacts are checked to learn the precise behaviour of each measured requirement for each architectural design.
2) The identified behaviours are then decomposed into the set of primitives that model their execution in each architecture, and those primitives are stored in the Mobile and Cloud Dimensions artifacts.
3) For each primitive, the Scrum Team estimates its execution rate and the values of its parameters, taking into account the architectural design being evaluated, the predicted execution scenarios (number of users, frequency of interaction with the app, etc.), and the measurements obtained from previous releases. Most of this data is detailed in the Business Model artifact.

4) If a new primitive has to be defined, the Scrum Team needs to characterize it and its parameters, measuring how it impacts on each architectural dimension.
5) Finally, the primitives are aggregated to get the boundaries of each feature for each architectural design under consideration.

Then, for the *Architectural Design Boundaries* task, the Scrum Team estimates the limits of the entire application for each architectural design and for each dimension selected by the Product Owner. For this, they simply have to aggregate the impact of feature multiplied by different parameters (such as the execution frequency).

Finally, one needs to identify which architectural design most efficiently satisfies the Product Owner's requirements. For the phone, the Scrum Team simply has to weight each dimension in accordance with its importance for the Product Owner and for the success of the mobile app. For instance, with regard to the resource consumption, various studies [39], [40] discuss the consumption that users expect —or accept— for different dimensions with different kinds of apps. Nevertheless, the Scrum Team may also use internal reports to set the weight and the limits for each dimension. As a result of this weighting, the different architectural options can be ranked by efficiency.

With regard to the cloud, the Scrum Team has to determine, for instance, which architectural options comply with the cost limits set out in the Business Model. To that end, the viability and success of any mobile app depends on two main parameters: Customer Acquisition Cost (CAC) and Customer Lifetime Value (LTV) [41]. The former refers to the resources that a business must allocate to acquire an additional customer. Typically this is a sum of marketing campaign costs, wages associated with marketing and sales personnel, and any additional professional services per customer acquired [42]. The latter is a prediction of the net profit attributed to the entire future relationship with a customer. To calculate this value, entrepreneurs analyse the average revenue per customer (ARPC), the gross margin (GM), the maintenance and evolution costs (EC), the customer retention cost (CRC), and the operating costs (OC) [43]:

$$LTV = \frac{(ARPC * \%GM) - (EC + CRC + OC)}{ChurnRate} \quad (1)$$

It is widely recognized that, for any form of recurring revenue model, LTV should be greater than three times CAC [44]. This can be achieved by reducing expenditure or increasing revenue. Whichever the case, the methodological approach presented here allows the entrepreneur to select an architectural design that reduces OC while maintaining the users satisfaction regarding the app resource consumption. To that end, the operating costs of the selected architecture should satisfy the following inequality:

$$OC \leq 3 * CAC * ChurnRate - ARPC * \%GM + EC + CRC \quad (2)$$

For a startup, one might think that cloud expenses would be a minor cost. But some analyses have indicated that they could surpass 80% of total operating costs [45].

Once the architectural designs satisfying the limit represented by Eq. 2 have been identified, the one that best addresses user satisfaction can be selected. If no architecture meets this condition, then a trade-off must be made in consensus with the Product Owner.

The estimation process discussed here may also be used to forecast when an app surpass some limits and to plan new releases. Developers would evaluate the trends in the usage of an app, identifying how new habits would affect the primitive operations, the use cases, their parameters, and the frequency at which they would be executed. They could then predict the evolution of each architectural dimension (such as the consumption) and plan the implementation of a more efficient architectural design for that usage.

In the following section, we shall present a conceptual framework assisting on the application of and validating the defined practices.This framework is specially focused on social mobile applications. Due to the space limitations and the authors' experience, this framework is focused on identifying the boundaries regarding both cloud and mobile consumption.

IV. ESTIMATING ARCHITECTURAL BOUNDARIES

In order to estimate the consumption boundaries of social mobile applications, both the phones and the cloud environment must be considered. On the mobile side, this analysis must be based on the resources that directly impact user satisfaction (such as battery consumption and data traffic) [11]. Instead, on the server side, one has to consider the operating costs of deploying the application's back end on a cloud environment. In the following subsections, we shall briefly recall how to estimate resource consumption on the mobile side, and then extend the analysis to estimating the operating costs on the cloud side.

A. ESTIMATING CONSUMPTION ON THE MOBILE SIDE

Three features of social mobile applications that have a direct impact on resource consumption are interacting with external entities, using specific sensors, and receiving information. Whether these operations are performed in the mobile devices or in cloud servers is an architectural decision that also affects consumption. In [46], we proposed a conceptual framework to assist developers during the architectural decision making process. This framework was based on a set of common primitive operations together with their associated resource consumptions. Most social application can be described as a combination of these primitives, thereby allowing the application's consumption boundaries to be estimated for each architectural option. Please, note that in order to estimate the consumption of other kind of applications, other primitive operations may be required. Some primitive operations are:

- Read the mobile's sensors (e.g., access the GPS sensor in order to geolocate the device).
- Store content in the mobile device.
- Get content from the cloud environment.
- Post content to the cloud environment.
- Receive a push notification.

Each primitive may have associated with it some parameters that directly influence the amount of resources that it consumes. Thus, for storing, posting, and getting content, the most important parameter is the size of the content since both data traffic and battery consumption depend on it.

TABLE 1. Average resource consumption of the primitives.

Primitives	Size (bytes)	Battery (μAh)	Data Traffic (bytes)
Operating System	n/a	3.56 ($\sigma = 5.22$)	0 ($\sigma = 0$)
getGPS()	n/a	7.20 ($\sigma = 11.27$)	0 ($\sigma = 0$)
store(content_size)	16	0.44 ($\sigma = 4.47$)	0 ($\sigma = 0$)
	140	0.59 ($\sigma = 4.95$)	0 ($\sigma = 0$)
	300 k	5.41 ($\sigma = 6.56$)	0 ($\sigma = 0$)
post(content_size)	16	16.83 ($\sigma = 5.94$)	1067 ($\sigma = 41$)
	140	18.14 ($\sigma = 5.87$)	1.16 k ($\sigma = 38$)
	300 k	71.39 ($\sigma = 22.91$)	318.46 k ($\sigma = 15.40k$)
get(content_size)	16	16.29 ($\sigma = 6.11$)	657 ($\sigma = 30$)
	140	16.26 ($\sigma = 5.47$)	782 ($\sigma = 32$)
	5.14 k	19.07 ($\sigma = 6.82$)	6.09 k ($\sigma = 129$)
	300 k	35.98 ($\sigma = 11.66$)	315.71 k ($\sigma = 31.20k$)
	668.34 k	47.41 ($\sigma = 13.70$)	700.87 k ($\sigma = 60.52k$)
push()	n/a	18.36 ($\sigma = 32.36$)	407 ($\sigma = 407$)

Table 1 lists each primitive's battery consumption and data traffic (whether received or transmitted) for specific sizes of the data being handled. To measure the resource consumption of each primitive, we developed an application that executes them and registers the battery consumed in microampere-hours (μAh) and the number of bytes transmitted and/or received. In order to get more precise measurements, each operation was repeated 3000 times, and the mean consumption was calculated in order to mitigate different consumption because of specific contextual situations. We also measured the consumption due to the operating system (OS) itself. The effective consumption of each operation was thus obtained by subtracting OS consumption from the operation's mean consumption.

Content size was selected based on the usual contents shared by social network users, i.e., geolocation (16 bytes to represent latitude and longitude), text comments (140 bytes for the average size of the messages posted), and images (300 KB, the average size of compressed images) [46]. If required, the measurements can be replicated for other content sizes, or their values can be extrapolated from those given in Table 1. Likewise, the activity performed by the user can also impact on the consumption of the sensor modules [47]. The tests have been performed to identify the usual consumption of the different sensors. If required, the measurements can also be replicated for different contexts. If additional primitives are required to estimate the consumption of a given application, their consumption can be calculated by performing similar experiments.

B. ESTIMATING THE OPERATING COSTS IN THE CLOUD

For the cloud environment, we propose a similar approach in order to estimate an application's operating cost and to forecast the architectural boundaries. In particular, there is a set of five primitive operations that characterize the common back-end logic of an application deployed in the cloud:

- Reserve an amount of cloud storage.
- Write content to the cloud and store it.
- Read content from cloud storage.
- Execute in the cloud a given algorithm that uses a certain amount of memory during a period of time.
- Invoke a cloud end-point, involving a certain amount of data transfer.

These primitives have parameters related either to the amount of data stored or transmitted or to the working memory used during the execution of a certain functionality. Based on these parameters, for each operation, a cost function must be defined according to the cloud provider's pricing structure. Here, we shall exemplify this process by considering Amazon Web Services (AWS), currently the largest major provider in terms of market share. In particular, we assume a microservice architecture implemented on Amazon's Function as a Service infrastructure (i.e., AWS Lambda and AWS API Gateway), and using AWS DynamoBD NoSQL storage for the persistence layer. In this context, it is important to stress the great complexity of AWS pricing schemes which address a wide variety of different situations. To this end, we shall need to make a number of assumptions:

- We use a single region (US_EAST) for the provision of all services. In a different scenario, the region choice could be more elaborate, based on such factors as reliability requirements (e.g., multi-region redundancy) or regulations that constrain storing sensitive data to a specific location.
- As DynamoDB uses 100 bytes of overhead for indexing purposes, the size of the items to be stored is considered to be no more than 924 bytes (i.e., 1 KB per stored item). This is important for the performance of the application in order to guarantee the contracted throughput while accessing the storage. In a different scenario it could be important to perform a throughput analysis if the data model requires larger items.

- We assume eventually consistent reads since typical mobile applications do not require an accurate state of the server data and could accept a fair amount of stale data. In time-critical applications this assumption should be reconsidered, and adjusted accordingly.
- We assume a maximum 10 TB of monthly response data. This does not include the data posted to storage, but is just the data generated as a result of the execution of the analysis in the micro-service layer.

Please, note that different cloud configurations can also be considered to better identity the operational costs and the architectural boundaries. Table 2 lists the cloud primitive operations together with their pricing and billing schemes as offered by AWS and for the selected configuration. The reserve operation depends on the allocation size in GB that must be indexed each month in the storage. The write operation is quantified in terms of a required throughput of item writes per second, and is charged hourly regardless of whether or not the total throughput is used. Similarly, the cost function for the read operation is based on the desired read throughput of items per second. The execute operation is parameterized with a combination of memory used in GB and the duration of the execution in seconds with a billing granularity of 10 seconds. Finally, the invoke operation's cost function corresponds to a combination of the cost of requesting a cloud end-point (cost per million requests per month) and the response data transferred (in GB per month) from the cloud back end to the front end as a result of executing the algorithm.

TABLE 2. AWS pricing model.

Primitives	Cost Function ($)	Billing period
reserve($storage_{GB/m}$)	$storage_{GB/m} * \$2.50\text{E}{-}01$	monthly
write($content_{item/s}$)	$content\_size_{item/s} * \$6.50\text{E}{-}04$	hourly
read($content_{item/s}$)	$(content\_size_{item/s}/2) * \$1.30\text{E}{-}04$	hourly
execute($memory_{GB/s}$)	$memory_{GB/s} * \$2.00\text{E}{-}07$	10 secs.
invoke($calls_m, resp_{GB/m}$)	$calls_m * \$3.50\text{E}{-}06 + resp_{GB/m} * \$9.00\text{E}{-}02$	monthly

Table 3 presents our analysis of the Free Tier offered by AWS. Again, we consider each of the primitives separately. These free quotas are permanent and reset monthly, except for the cloud-endpoint invocation which is limited to the first 12 months of operation.

Despite the realism of the presented model, we must point to three issues that were left out of consideration. First, the pricing model for the write/read throughput corresponds to an on-demand premise. A more general model would take into account estimation of the temporal provisioning sequence of the application. In such a context, there are other pricing models which reserve the infrastructure for a given period of time, and which involve an initial investment in exchange for cheaper hourly costs. Second, for the sake of simplicity, we have left out optional optimization and development services such as AWS DAX, AWS API Gateway Cache, and AWS Cloudtrail which provide extended features that can improve the performance or the development lifecycle.

TABLE 3. AWS monthly free tier.

Primitives	Quota	Duration
reserve($storage_{GB/m}$)	$storage_{GB/m} = 25$	permanent
write($content_{item/s}$)	$content\_size_{item/s} = 25$	permanent
read($content_{item/s}$)	$content\_size_{item/s} = 50$	permanent
execute($memory_{GB/s}$)	$memory_{GB/s} = 400000$	permanent
invoke($calls_m, resp_{GB/m}$)	$calls_m = 1000000$	12 months

And third, a uniform distribution is taken for the number of accesses to storage, which allows one to assume a linear provision of throughput. This is a limitation with respect to the auto-scaling possibilities of AWS DynamoDB.

C. USING THE PRIMITIVES TO ESTIMATE THE CONSUMPTION

In order to estimate the architectural boundaries of a particular application for the consumption dimension, in the *Identify Arch. Significant Features* activity one has to identify which features are executed in the mobile device and which in the cloud. This depends strongly on the application's architectural design. Then, in the *Smartphone Boundaries* and *Cloud Boundaries* activities, each feature must be decomposed into a set of primitive operations, taking into account specific values for their parameters. This provides an estimate of consumption for each particular feature of the application. Finally, in the *Architectural Design Boundaries* activity, the consumption of all the features on each side —mobile and cloud— are aggregated to get the resource consumption and the operating costs of the entire application.

For example, consider a mobile app that periodically gets and stores the location of its users. This feature is present for different purposes in almost any of today's social mobile applications. Once the user location has been obtained, it can be stored in the mobile device or uploaded to the cloud environment, thus resulting in two different architectural designs for the application, which one can call mobile-centric and server-centric, respectively. If, for instance, the information is stored in the cloud, then the primitives into which this feature can be decomposed are the following:

- First, the user location is obtained in the mobile phone (using the *getGPS* primitive), and then it is posted to the cloud (with the *post* primitive).
- In the cloud, the amount of data that is going to be used has to be *reserved*, and then the user location is stored in the database (using the *write* primitive).

By aggregating these primitives taking into account the rate at which they are executed and the specific values of their

parameters, one can estimate the resource consumption and operating costs of this feature.

We are currently working on the development of a tool to support this process. This tool allows developers to specify the different functionalities using the defined primitives, to define a range of values or parameters for each primitive or functionality, and to automatically generate the estimated consumption for each combination of parameters. This tool gets as input a JSON document with the specification of the application (i.e., the different architectural designs, the use cases for each architecture, their parameters, etc.). As a result, it provides a CSV with the estimated consumption for each architectural design and combination of parameters. From that CSV, developers can manually generate different charts showing the consumption trends, for instance, when the number of users increases. Currently, this tool only supports the generation of estimations for the mobile side of the application, we are working on also supporting the server side. More information on this tool is available in [7].

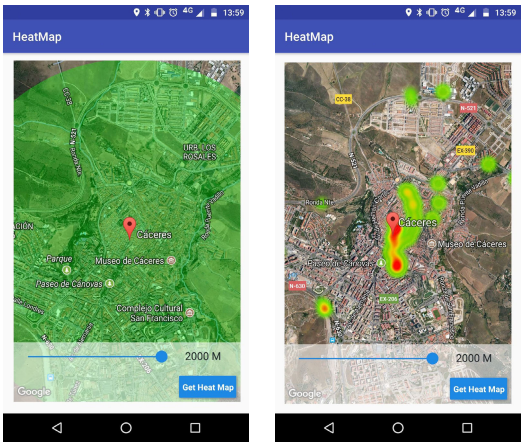
The following section presents a more extensive case study and uses it to validate both the proposed process and the framework.

V. CASE STUDY: HeatMaps

In this section we present a case study to illustrate how our proposal can help in the development of a mobile application by providing an early evaluation of different architectural options estimating their consumption. Then, in the following section, we shall compare these estimates with the actual figures of consumption of three variants of the application that were developed using different architectural approaches.

For this case study, we selected a social application that generates heat maps (i.e., shading matrices which, in this case, are overlain on a geographical map) from the geographical positions of its users. Such an application would provide interesting information for city authorities who would get precise information about the flow of tourists, the places they visit, etc., but also for the visitors themselves who could consult the most interesting spots for sightseeing, peak hours, etc., so that they would be able to better plan their visit.

The HeatMaps application entails a significant workload due both to obtaining information through the mobile phone's sensors and to algorithmic computing. Hence, it should be well-suited to displaying the impact that architectural decisions can have on the mobile applications' resources consumption. For this case study, we shall focus solely on the application's main architectural significant features: gathering and storing the location of each user in real time, and generating heat maps upon discretionary request of a user. Figure 2 shows what the app's user interface might be. To request a heat map, the user would mark its centre (Fig. 2a), and choose the radius and the time frame to be considered. The app would then present a map with all the users of the application that have been in the chosen area within that time frame (Fig. 2b). Obviously, a greater number of functional features would have to be considered for a production-ready application.



(a) Map request interface. (b) Example of heat map.

FIGURE 2. Interface of the HeatMaps mobile app.

In order to estimate the consumption dimension, we followed the seven tasks set out in the previous section.

A. ARCHITECTURAL DESIGNS TO ANALYSE

The two main requirements of the HeatMaps application are assumed to be present in the application's MVP: continuously tracking users' positions, and generation of heat maps. We shall consider three architectural designs implementing these features. First, a server-centric design (*cc*) that delegates both features to cloud servers, minimizing computation in the smartphone. Second, a mobile-centric architecture (*mc*) that takes advantage of the computing capabilities of smartphones by both storing user positions and generating the maps locally. And third, a hybrid (*hy*) architecture that attempts to balance the workload between the mobile devices and the cloud servers, gathering locations in the smartphone, but computing the heat maps in the back end.

B. ARCHITECTURAL DIMENSIONS

Let us assume that our business model wants to maximize user satisfaction in order to get information from a large number of users, while keeping operational costs as low as possible. Since the three architectural designs mentioned above will have the same functionality and user interface, we will consider the resource consumption dimension and, concretely, two issues which are always important while moving around — battery consumption and data traffic—, as the dimensions to optimize for user satisfaction. They will both be estimated for each of the three architectural options considered. For the operating costs, we will make an estimate of the costs of running the back end of the application in a cloud environment. In order to estimate the consumption of these resources, we will use the primitives described in Sec. IV.

C. IDENTIFY ARCHITECTURAL SIGNIFICANT FEATURES

As noted above, the two main features of the application are location gathering and heat map generation. Their implementation largely depends on which architectural design is chosen. In the server-centric architecture (Fig. 3), the mobile device gets the location from its GPS sensor and sends it to the cloud. In contrast, in both the mobile-centric and hybrid designs, the GPS location is kept in the device, requiring no transmission.

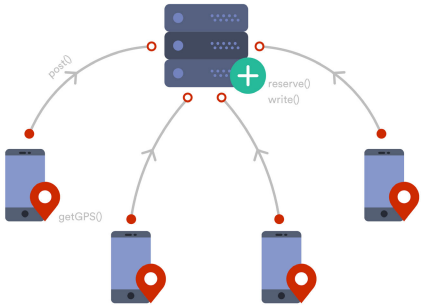


FIGURE 3. Location gathering in a server-centric design.

With respect to the heat map generation, in the server-centric architecture, the user requests a heat map from the cloud server, which filters and aggregates the data, and returns the results to the mobile device for them to be displayed to the user. In contrast, in the mobile-centric design (Fig. 4), the user requests location data from the rest of the app's users, whose devices filter their data to send back to the requesting device only those data that meet the conditions of the map, and the requesting device aggregates the results and displays the heat map. Finally, for the hybrid version (Fig. 5), the device asks the cloud server for a heat map, the server requests data from the rest of the users, their devices again filter the data according to the conditions of the map and send the data meeting the conditions back to the server which aggregates the results and returns them to the requesting device for display to the user.

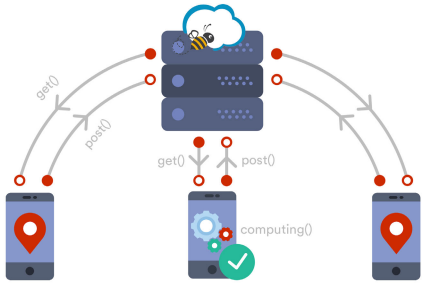


FIGURE 4. Heat map generation in a mobile-centric design.

D. SMARTPHONE BOUNDARIES

We need to estimate the consumption of the two selected functional features for each of the architectural designs considered. To this end, we use the primitives described in Sec. IV-A.

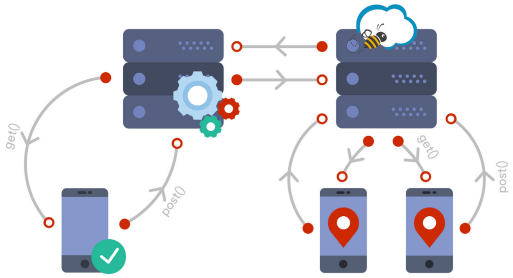


FIGURE 5. Heat map generation in a hybrid design.

For location gathering, in the server-centric design the consumption derives from reading the GPS position of the device, plus sending 48 bytes of information (latitude, longitude, and timestamp — 16 bytes each) to the cloud (Eq. 3).

$$mobile\_loc_{cc} = getGPS() + post(48B) \tag{3}$$

In both the mobile-centric and hybrid architectures, this consumption corresponds to obtaining the GPS location, plus storing 48 bytes of information locally (Eq. 4).

$$mobile\_loc_{mc,hy} = getGPS() + store(48B) \tag{4}$$

For the generation of the heat map, in both the server-centric and hybrid architectures, the consumption comes from sending the server a heat map request (comprising latitude, longitude, radius, init date, end date, and user id — 92 bytes), plus receiving the information needed to render the heat map. This information consists of a set of active positions (AP), i.e., GPS coordinates where at least one user was located in the specified period of time, plus the number of users present at that location (40 bytes). Hence, the amount of information depends on the radius of the map and the number of users that were present in that area in the time frame selected (Eq. 5).

$$mobile\_map_{cc,hy} = post(92B) + get(AP * 40B) \tag{5}$$

For the mobile-centric design, this consumption in the requesting smartphone is expressed by Eq. 6 — sending the request plus receiving the data to render the map, i.e., a set of active positions of each user (AP_u) multiplied by the number of users present in the map (U_M).

$$mobile\_map_{mc} = post(92B) + get(AP_u * 40B) * U_M \tag{6}$$

Additionally, both the mobile-centric and hybrid designs involve consumption in the rest of the devices whenever a heat map is requested. This corresponds to receiving a push notification with the request plus sending the information required for the heat map (Eq. 7). Note that if the device was not located in the area and time frame requested then the second term of this equation would be zero.

$$mobile\_map'_{mc,hy} = push(92B) + post(AP_u * 40B) \tag{7}$$

E. CLOUD BOUNDARIES

Similarly, the cloud costs of the application must be estimated for each of the architectures considered. To this end, we use the primitives described in Sec. IV-B.

For location gathering, the estimated costs of a server-centric design correspond to one invocation to the cloud services, plus reserving memory storage for the GPS position of the user, plus storing the information received, plus the cost of executing the function (Eq. 8). Note that, for the execute primitive, we have made the conservative assumption that each execution requires 1 GB and 1 second for its completion. This is an upper bound. In the estimates and tests that we carried out, no execution took more than 1 s nor required more than 1 GB of memory.

$$cloud\_loc_{cc} = invoke(1, 16B) + reserve(48B) + write(48B) + execute(1GB) \quad (8)$$

In contrast, as mentioned above, for the mobile-centric and hybrid architectures, the implementation of this feature does not need any cloud resources, and therefore has no associated consumption.

For the generation of the heat map, the estimated cost in a server-centric design corresponds to making an invocation to the cloud, plus reading the information needed from the cloud storage, plus the cost of executing the algorithm generating the map (Eq. 9).

$$cloud\_map_{cc} = invoke(1, AP * 40B) + read(AP * 40B) + execute(1GB) \quad (9)$$

For a mobile-centric system, this feature does not require cloud resources so that its cost is zero. For the hybrid design, however, the cost comes from invoking the corresponding cloud endpoint, triggering the information request to the mobile devices which will result in a data transfer of the information for the map, executing the algorithm to generate the heat map, and the costs generated for each of the users present in the map which consist of a single cloud invocation plus processing the corresponding data for each of those users (Eq. 10).

$$cloud\_map_{hy} = invoke(1, AP_u * 40B * U_M) + execute(1GB) + invoke(1, 16B) * U_M + execute(1GB) * U_M \quad (10)$$

F. ARCHITECTURAL DESIGN BOUNDARIES

Once the mobile and cloud costs of each of the app's features have been decomposed into those of primitive operations, the total cost of a given architectural design can be analysed.

For any given architectural design *arch* of the application, whether mobile-centric, server-centric, or hybrid, the total consumption in the mobile devices is expressed by Eq. 11. The terms on the right hand side correspond to the location gathering consumption multiplied by the location updating rate (PI_{freq}), the heat map generation consumption multiplied by the estimate of the users' heat map request rate (HM_{freq}), and the consumption due to providing the information for a heat map requested by other users multiplied again by the estimated users' heat map request rate and by the number of active users of the application (U_{app}).

$$mobile_{arch} = mobile\_loc_{arch} * PI_{freq} + mobile\_map_{arch} * HM_{freq} + mobile\_map'_{arch} * HM_{freq} * U_{app} \quad (11)$$

The total cloud operating cost can be estimated in a similar way, taking into account the corresponding rates and number of active users (Eq. 12):

$$cloud_{arch} = cloud\_loc_{arch} * PI_{freq} * U_{app} + cloud\_map_{arch} * HM_{freq} * U_{app} \quad (12)$$

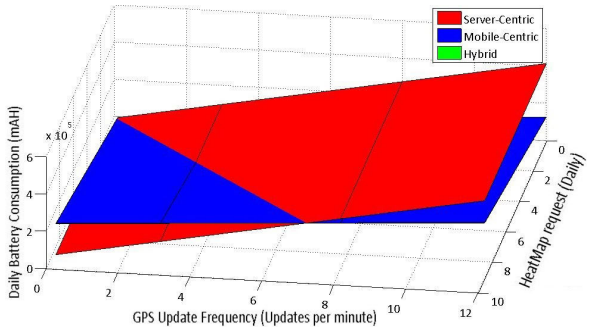
G. IDENTIFY THE MOST EFFICIENT ARCHITECTURE

The different architectural alternatives can now be assessed relatively in accordance with the business needs, and hence a decision made as to which to choose. To do so, the parameters in the above equations must be fixed. As explained in Sec. III, this information can be obtained from estimates made in the business model or from real measurements obtained from previous releases of the application. For the present case study, these parameters are the location gathering and heat map generation rates, the number of users, the time frame of the requested heat maps, the proportion of requests in which a given user is present, and the proportion of spatial coincidence between users in a given heat map.

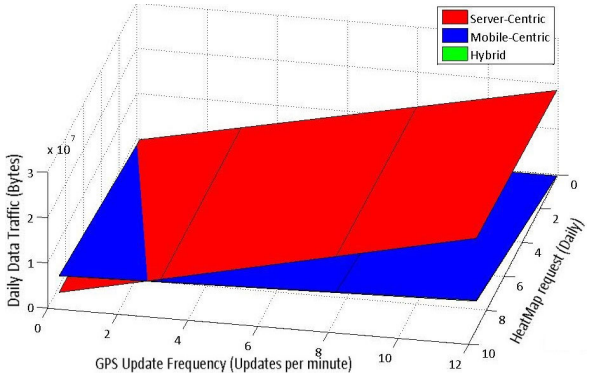
Figure 6 shows the consumption estimated for the three architectures. The charts assume frequency ranges of once every ten minutes to once every five seconds for the location gathering frequency, and once every ten days to ten times per day for the heat map generation. The values fixed for the rest of the parameters are 1000 active users, a 60-minute average time frame for the heat maps, a 1% proportion of the heat maps in which a given user is present in the requested area, and a 10% proportion of spatial coincidence between users. The values of all these parameters can be changed in the Matlab files included as additional material with this communication, leading of course to different estimates.

Figure 6a shows the battery consumption estimates for the three design alternatives. The consumption of the hybrid architecture is very similar to that of the mobile-centric system (in blue), and is therefore hidden from view. Nevertheless, the figure clearly shows that consumption increases significantly with the rate of location gathering for all three architectures, especially for the server-centric architecture (in red). The heat map request rate significantly increases consumption in the hybrid and mobile-centric designs, but has barely any effect in the server-centric case.

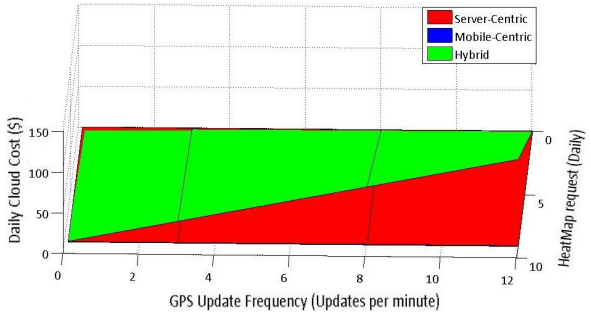
Similarly, Fig. 6b shows the estimated data traffic for the three architectures. In this case, increasing the location gathering rate is even more significant for consumption, and is particularly demanding for the server-centric architecture. The heat map request rate is far less relevant, although the



(a) Mobile consumption - battery in mAh.



(b) Mobile consumption - data traffic in bytes.



(c) Cloud consumption - daily costs.

FIGURE 6. **Estimates for 1000 active users.**

hybrid and mobile-centric architectures are slightly more sensitive to changes in this parameter than the server-centric case.

Finally, Fig. 6c shows the estimated daily cloud operating cost. Obviously, for the mobile-centric alternative, it is always zero, while for the server-centric design it increases slightly with both the location gathering and the heat map generation rates. One sees that the hybrid alternative (in green) is significantly affected by these parameters, incurring greater costs as they increase.

In sum, the possibility of establishing ranges, or actual and future values, allows one to observe the evolution of the different architectural options and thus analyse their limits. This in combination with the application business model, can be used to make architectural decisions which are better informed. For the HeatMaps application, under most circumstances a mobile-centric solution would incur lower cloud costs and mobile resource consumption. However, if the location gathering rate could be kept low to around two or three updates per minute, the server-centric architecture would present better behaviour, especially as the number of map requests increases.

VI. MEASURING ACTUAL CONSUMPTION

In order to validate the identified architectural boundaries, we implemented the HeatMaps application with the three architectural alternatives discussed above, and then measured their actual consumption of resources. The source code of the three implementations is provided as additional material with this communication.

We developed the apps for the Android OS since this was the operating system for which the primitive operations' consumptions were measured. For the implementations, various frameworks and libraries were used, abstracting from low level functionalities. We would emphasize that, while the use of different APIs may increase consumption with respect to the estimates, proportion-wise this increase should be the same for each architectural design.

Some of the libraries included are among those most widely used in commercial mobile apps. In this way, our procedure would be like that which any firm or startup would follow. In particular, we made use of the following:

- For communication with the cloud, we selected the widely used Retrofit framework.[1] This provides better performance than other alternatives [48].
- In order to facilitate communication between mobile devices in both the hybrid and the mobile-centric designs, we used nimBees.[2] This framework provides specific support for a mobile-centric design, with smart push notification capabilities based on the user's current or past locations.
- We used Realm[3] to store user locations in the smartphone, as this database offers better large entry writing and reading performance [49].
- To prepare the maps, we used the API provided by Google[4] for its ease of integration with Android apps.

For the hybrid and server-centric apps, the corresponding back ends were implemented in Node.js, and deployed on Amazon Lambda, storing all the information gathered in DynamoDB to create the heat maps. Again, the nimBees API was used to request information from the mobile devices, such as their locations during a specific time frame.

Once the apps had been developed, several experiments were conducted to compare our estimates with the apps' actual consumption. The main parameters of the experiments were:

- Five users were active during the experiments.

[1] Retrofit, Square Open Source. http://square.github.io/retrofit/
[2] nimBees. http://www.nimbees.com
[3] Realm for Android. https://realm.io/
[4] Google Maps API. Heatmaps. https://developers.google.com

- All the users were present in all the heat maps requested, and the spatial coincidence between users was set at 10%.
- Each experiment lasted 30 minutes in order to obtain stable consumption measurements.
- To measure consumption under different situations, each app was evaluated with different frequencies of location gathering (from 0.4 to 1.2 times per minute) and heat map requests (from 0.06 to 0.16 times per minute).

The measurements were made on five different smartphones, running different versions of the Android OS: one Motorola Moto E 1st generation (Android 5), one Motorola Moto E 2nd generation (Android 5), one Honor 8 (Android 8), one Xiaomi RedMi 5 (Android 8.1), and one Xiaomi RedMi 4 (Android 7.0). Each experiment involved having the devices in movement, being carried by the user while he or she was moving around the university campus, and requesting heat maps at the pre-established rates. In addition, in order to get some stable values, the mobile devices were reset to the factory values before each experiment in order to eliminate any interference or residual consumption of other applications. Figure 7 shows two of the maps generated as a result of the experiments.

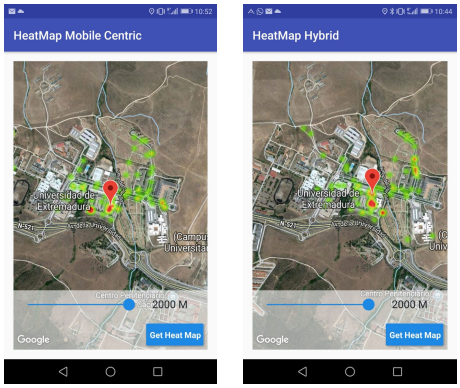
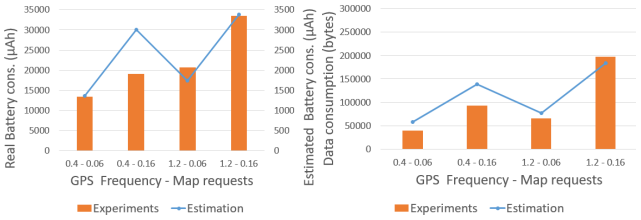


FIGURE 7. Heat maps generated in the experiments.

In order to obtain the data traffic and battery consumption during each test, we used Android's bug report functionality [50].This generates logs with information about battery consumption, data traffic, sensors used, etc. by the different applications installed in the device. This information was reset before each test and the logs were generated after each execution. In order to help with the reproducibility of our experiments, the 60 logs generated are provided as additional material with this communication. All the logs were processed using the tool Battery Historian [51], which among other information provides the estimated battery and data traffic consumption of each installed app.

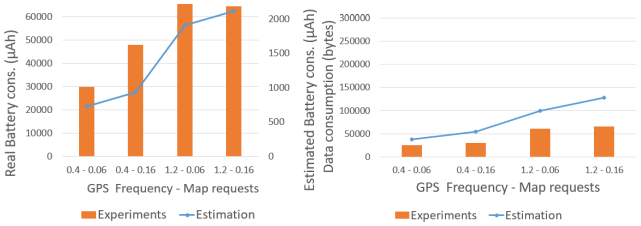
Figure 8 shows the comparison between the estimated (blue line) and the actual (orange bars) consumption for the mobile-centric architecture. As one can see in Fig. 8a, the real battery consumption is always greater than the estimates. Some of the reasons are that the estimates were made assuming that the smartphone screen is always off, whereas it had to be on during the experiments to start/stop the app and to request maps. Moreover, since the smartphones were in movement, real consumption would also have depended on other issues such as network coverage, etc. However, when the results are scaled appropriately (the reason for the two different Y-axes in Fig. 8a), one sees that the differences scale linearly and the two shapes match. Indeed, configurations with lower location gathering and map request frequencies show lower battery consumption, and the growth of consumption with increases in these frequencies is similar in the two charts. On the contrary, for data consumption (Fig. 8b), the actual measurements are very close to the estimates.



(a) Battery consumption. (b) Data traffic consumption.

FIGURE 8. Consumption for the mobile-centric architecture.

Figure 9 compares the estimates with the real consumption for the server-centric architecture. Again, the differences for battery consumption were found to scale linearly, but for each configuration the ratio between the real and the estimated data was similar for both the battery and the data traffic measurements. Apart from the screen consumption, we would attribute some of the deviations to optimizations of the different libraries used to exchange information in the three architectural designs.

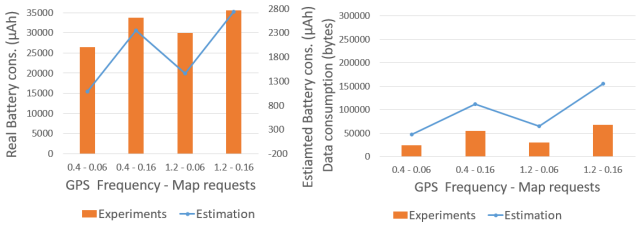


(a) Battery consumption. (b) Data traffic consumption.

FIGURE 9. Consumption for the server-centric architecture.

Figure 10 shows the comparison for the hybrid architecture case. Once again, there were similar linear differences for battery consumption, but the consumption trends for the different configurations were very much alike.

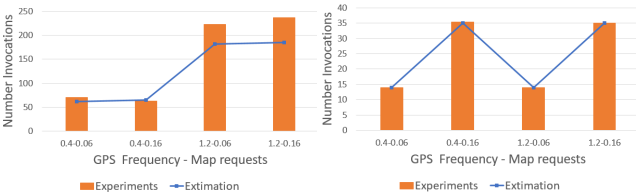
We did not analyse actual cloud costs because Amazon's free tier was not exhausted during the experiments. Nevertheless, the number of requests to the cloud environment (shown in Fig. 11 for the server-centric option) correlates linearly with cloud costs, as may be deduced from the equations in Sec. V. The figure shows that the actual number of cloud



(a) Battery consumption. (b) Data traffic consumption.

FIGURE 10. Consumption for the hybrid architecture.

invocations was almost the same as the estimate, with minor deviations for the greater location gathering rates. These deviations can be attributed to the fact that the estimates were calculated for exactly 30 minutes, while the actual experiments where just approximately for this duration, with small variations for each user. Figure 11b shows the cloud requests for the hybrid architecture. As can be seen, here the number of invocations is exactly as estimated. This is because for the hybrid design the figures depend only on the number of heat maps requested, not on the exact duration of the experiments.



(a) Server-centric architecture. (b) Hybrid architecture.

FIGURE 11. Number of requests to the cloud environment.

A. DISCUSSION

To synthesize, Figs. 8–11 showed that the real consumption measured during the experiments roughly corresponded to the estimates, with both following the same trends for the scenarios considered. These results indicate that, within a certain margin of error, the process proposed here can be used to estimate the consumption, and to identify the boundaries of different architectural options for a given application.

The evolution of the consumption followed different trends for each design. For the mobile-centric and hybrid architectures the consumption mainly depended on the location gathering rate, but, as was foreseen, for the server-centric case it depended on the number of maps requested.

As expected, the least accurate of the three types of estimate was that of battery consumption. Indeed, battery consumption depends on various factors that were not under control during the experiments (screen consumption, external libraries used in the development, etc.). Also, the values of consumption of the primitive operations in Table 1 were measured for a particular device with a specific version of Android OS, while consumption in the field will depend on each device's hardware and OS version. We could have obtained more accurate estimates by measuring the consumption of the primitives for the particular devices used during the experiments, and even more accurate measurements could have been made using an external power monitor. However, our interest was in trends rather than actual consumption figures, i.e., how, for each of the architectural designs considered, resource consumption grows with increasing numbers of users and rates of execution of the app's different functional features.

With respect to data traffic, the results showed the estimates to be very close to the real figures for all three versions of the app. Indeed, data traffic is unaffected by which particular device or OS version is used.

For the cloud costs, the experiments showed the estimates of the number of requests in the three versions of the app. As cloud costs depend mainly on this parameter and on the size of the data involved (see Sec. IV-B), one can argue that these estimates are fairly acceptable as proxies.

The set of practices incorporated into the development process and the conceptual framework that we have presented allow one to obtain consumption trends of each architectural design. Also, similar frameworks can be used to obtain the architectural boundaries for different dimensions. The objective of this work was to be able to provide a development team with these trends in the early stages of their work, so that they can better plan in the medium term the evolution of the app in accordance with its expected growth and use. Of particular interest is for the team to be able to identify the boundaries at which a given architecture ceases to be advantageous, and changes need to be made for part or the whole of the system to evolve to another design. For instance, for the running example, these boundaries can be clearly seen in Fig. 6 in which, if the GPS information is obtained more frequently than six times per minute, the efficiency of the server-centric architecture is overtaken by that of the other two types of design, and the system should be migrated to a mobile or hybrid design.

The consumption of mobile resources from the Heatmaps case study described in this section is very dependent on the use of GPS positioning, as mentioned above. While the use of GPS positioning is a very common feature of social mobile apps, the framework for mobile consumption presented here has been tested in other applications that do not rely so heavily on positioning. Specifically, in [46] the authors examine the resource consumption of a mobile app for exchanging information between contacts in which GPS positioning is not considered. Similarly, in [21] the framework is used for estimating the consumption of a context aware application.

B. THREATS TO VALIDITY

The presented methodology was evaluated in one case study and with three different implementations for different architectural designs in order to validate the estimations obtained by the conceptual framework. Data was collected to evaluate its feasibility, completeness and precision. In this section,

the possible threats to validity are discussed according to the four types of possible threats reported in [52].

Construct Validity is concerned with the relation between a theory and its observation. Threats to construct validity refer to the extent to which the setting of an empirical study actually corresponds to the construct under study. The framework defined in section IV has been applied to a case study in order to identify the architectural boundaries. Three different designs of the case study have also been implemented obtaining similar consumption trends. Therefore, we can conclude that the application of the framework is feasible.

Conclusion Validity is concerned with the relationship between a treatment and the conclusions drawn from it. Threats to conclusion validity refer to the ability to draw correct conclusions about relationships between the treatments and the results of an empirical study. The main source of information used during this validation was the results obtained from the execution of the case study for the three different architectural designs. The results obtained and the comparison between the estimated and the actual consumption obtained show that they are not the same, but they follow the same trend. Therefore, an exact precision of the consumption cannot be obtained. Nevertheless, since the consumption trends are the same, these deviations do not impact the conclusions we draw. We can conclude that the framework is perfectly valid to estimate the boundaries of different architectural designs in early phases.

Internal Validity is concerned with the causal relationship between a treatment and its results. In the validation of this conceptual framework, these threats are related to truth of the metrics obtained, and the application of the methodology. First, the case study has been developed by the researchers themselves, so the implementation may be influenced by the defined conceptual model. This threat to validity was directly addressed by the researchers during the design and implementation of the systems. The behaviour of the application and the technologies to be used were defined at the beginning of the experiment. During the implementation, the same technologies and defined behaviour were implemented for the different designs. Second, the use of the application during the validation could be conditioned by the estimations. To address this threat, specific routes were defined for each user in which the percentage of overlapping with other users was the same to the one defined for the estimation. Also, the frequency of execution of different functionalities (such as the generation of heatmaps) was the same to the one defined for the estimation. Finally, the characteristics of the devices chosen to validate the system may involve a consumption that could be different for other devices with different properties. To address this threat, the validation was performed with five different devices, with a range of operating system versions from Android 5 to Android 8.1 and, in addition, from three different manufacturers. Therefore, it can be concluded that the different internal threats to validity were addressed to mitigate and / or eliminate them.

External Validity is concerned with the generalisation of the conclusions of the validation. Threats to external validity refer to the ability to generalise the results and conclusions beyond the setting of the study. First, the conceptual framework presented can be applied to measure the architectural boundaries for different dimensions. However, in the case study, only consumption, battery and operational cost have been evaluated. The framework presented is independent of the dimensions measured for the mobile environment or for the cloud environment. If developers want to measure other dimensions, the consumption of the different primitive operations for those dimensions should be measured and added to the framework. The methodology presented and evaluated would be the same. Second, only a subset of primitives has been estimated. Estimating the architectural boundaries of other applications may require the use of other primitives. The presented conceptual model is independent of the measured primitives and, even, encourages developers to measure and add new primitives. To solve this problem public repositories would be created for developers to add and share the measured primitives. Finally, the conceptual model has only been validated for one case study, so the generalisation of the results could be limited. Although this article only presents the application of the framework to one case study, it has also been applied for estimating the consumption of social networks and to recommendation systems based on air quality. However, they have not been presented in this article, avoiding making it too complex. Therefore, it can be concluded that the different external threats to validity have been addressed by making the conceptual framework independent of the application domain.

VII. RELATED WORK

A large body of research has dealt with measuring or estimating some boundaries of an architectural design [53]. Some studies propose offloading resource-consuming tasks to cloud servers. As has been demonstrated here, these techniques may reduce the consumption or the response-time in some circumstances, but not in all. Therefore, the situations in which offloading certain tasks to the cloud may be beneficial need to be identified.

With regard to methodological approaches and practices, in [54], the authors indicate that cloud computing has generated a new global market thanks to the increasing adoption of devices such as smartphones and tablets, and this in turn has led to the birth of a new generation of startup firms. The work analyses the impact of cloud technology for entrepreneurial activities, finding that cloud computing allows a new business to be ''born global'', to reduce its initial investment and operating costs, and to achieve high growth rates. They also find that it reduces diversification costs by eliminating the barriers to entry. Gupta *et al.* [55] analyse the adoption of cloud technology by small- and medium-sized enterprises. They find that the most important factors influencing cloud usage by such firms are ease of use, security and privacy, and cost reduction. All of these studies, however, compared the costs

of deploying applications using cloud computing with those of purchasing all the required hardware and software. Current mobile phones have sufficient resources to store and process all the data gathered. This allows startups to reduce even further the operational costs of deploying a new application. There is thus a requirement for new approaches to identifying when to use cloud environments or other architectural styles depending on the business model and the budget.

In this sense, there are some works focused on identifying the best cloud provisioning plan within a budget. Hasan and Hossain [56] indicate that startups are often failing due to not taking into account the risks involved in investing money without being aware of the actual resources they can avail themselves of given their budget. In order to obviate these risks, the authors construct a model which a firm can use as a guide by specifying its budget as a model parameter, and then applying a methodical process to determine what cloud resources are within its possibilities. In particular, they identify the driving factors to the cost measurement, such as number of additional elastic IPs, number of elastic IP remaps, data transfer out, data transfer in, etc., they build a fitness function and apply genetic algorithms in order to solve the optimization problem focused on what cloud resources can be obtained from the specified budget. Nevertheless, this model does not consider to obtain some computation capabilities by onloading some services, or part of the application, to end devices by designing a mobile-centric or a hybrid architecture. Startups also need a method that they can apply to guide them in decision-making by allowing them to analyse the architectural alternatives if their budget does not meet the cost of the required resources or just to reduce the operational expenses maintaining the quality of service.

In addition, in [57] and [58], some of the authors of this article presented a solution that analyses the user needs to support the decision-making process of selecting the most suitable cloud service provision plans for different cloud providers. To that end, first, the requirements of the software to be deployed (CPU, memory, IO performance, and storage) and the usage schedule have to be specified. Then, the optimal configuration plan is generated for different providers. This solution made use of feature models in order to define the characteristics of the different providers. Nevertheless, it was focused on identifying the optimal configuration in terms of infrastructural requirements and cost. In this work, we present a framework that takes into account both the smartphone's resource consumption and the operating costs in the cloud. In addition, the presented solution can be applied in the early phases while the previous one was applied once the different services and modules are implemented.

Focused on measuring consumption in applications, first, we can find some works focused on measuring the exact resource consumption. In [47], the authors measure the power consumption of different components and sensors by instrumentation at the circuit level and considering different configurations. Reference [59] develops a fine-grained energy profiler. This allows the authors to measure the energy spent performing such tasks as rendering images on the device's screen, or building an application's internal database. This information is extremely useful for developers seeking to improve resource consumption, but the application has to be built before it can be analysed. In addition, the source code has to be instrumented for system-call tracing and routine tracing. These traces are recorded and analysed in order to generate the energy profiles. A step further, taking into account the wide array of sensors installed in current mobile devices, is taken in [60]. This proposes a solution optimizing the sensing requirements of all the applications running on the device, and thus reduce the overall consumption. Concretely, they propose to use an optimizing sampling scheduler in order to schedule the sampling rates and sharing the obtained sampling among different applications when possible. Again, these approaches allow developers to reduce the resource consumption. Nevertheless, they only can be used once the application is developed. A methodology for designing efficient applications regarding different dimensions (battery consumption, data consumption, operational cost, etc) is needed. Nevertheless, both proposals could be integrated in order to analyse the resource consumption both in the early and in the late phases.

Also focused on measuring the precise consumption of a mobile application, some studies compared the consumption information provided by devices with the measurements made using an external power monitor [61]. Others take this information from the device's battery [62] or from a modified version of the kernel [63]. These studies show that the consumption information obtained from the devices is sufficiently accurate for the analyses and experiments that have been proposed in the present work to be carried out.

On the other hand, there are also approaches closer to the presented conceptual model focused on providing some guidance to developers in order to design applications reducing the resource consumption. In [64], the literature on the consumption patterns of mobile applications on Android platforms is systematically reviewed. The authors find there to be only limited work assisting developers to choose the most suitable software architecture, application interface, and behaviour of an application in terms of resource consumption. In their analysis, they identify architectural design and the network technologies used for communications to be among the main features affecting consumption. In addition, they identified that the energy conservation of mobile devices is of great interest for the research community since it is one of the most limited resources of these devices. Nevertheless, this review reports no work on trade-offs between battery and data traffic consumption, or on cloud infrastructure costs. It focuses on detailing the different approaches that can be used to optimize the energy consumption.

In [46], some of the present authors proposed a conceptual framework for measuring patterns of consumption of mobile applications in early stages of their development. The framework also estimates the trends in consumption as the number of users increases or when how they use the application

changes. This framework has also being applied for estimating the consumption of context-aware mobile applications [21]. The study does not, however, analyse operating costs when some tasks are offloaded to the cloud.

In [65] the authors indicate that developers need to estimate their apps' energy consumption. Therefore, they propose, first, a machine learning-based energy consumption model trained by running different applications and that can estimate the energy consumption from their source code. In addition, these authors also state that developers need energy optimization guidelines. Therefore, they propose different techniques that can be applied in the communication and security layers to create more efficient applications and a new design pattern for making view updates more energy efficient [66]. Nevertheless, these works only propose guidelines for improving the energy consumption dimension.

Finally, other approaches focus on offloading some part of a mobile application depending on its context and the consumed resources. For instance, AndroidOff [67], collects the execution costs of a subset of the methods and then uses static analysis to predict the execution costs of the remaining methods. Then, optimization algorithms decide when some methods should be offloaded to the cloud. Nevertheless, the authors do no provide guidelines to developers in order to design applications in which some modules can be offloaded.

In sum, firms need to have a clear picture of the (mobile and cloud) boundaries an application. This is crucial for a product to be sustainable and successful. Most research in this area has focused on optimizing applications behaviour after they have already been developed, and on determining the reduction of the costs of using cloud computing. To the best of the authors' knowledge, however, there has been no work assisting developers in choosing the software architecture that is best suited to both their applications and business models.

VIII. CONCLUSION

The business models behind mobile applications are often based on a very short time-to-market. The success of these apps and of the firms behind them depends on a subtle balance between user satisfaction, operating costs, and timing. It is not surprising therefore that an application's architectural design has a major impact on the firm's operating costs and on the users' satisfaction. In this communication, we have detailed a set of practices that should be done to estimate when each architectural design should be applied. In addition, we have presented a conceptual framework assisting on the estimation of the consumption dimension.

In our proposal, we considered three different architectural design for a given application —from a server-centric design at one extreme to a mobile-centric design at the other— and we have shown how the consumption of different resources can be estimated for these architectural designs. We then built the three versions of the app, and measured their actual consumption figures, comparing them with our estimates. The results showed that the process we propose can be of use for estimating the architectural boundaries at early stages of development of an application, assisting the development team with their decision making during the design stage and with their planning of the app's architectural evolution.

As future work, we work on building a tool for providing support to the presented framework. This tool will take as input an initial definition of the application and the architectural designs to evaluate in order to provide the most suitable cloud configuration and estimations on the resource consumption and operating costs for each option.

REFERENCES

[1] ITU Telecommunication Development Bureau. (2019). *Measuring Digital Development. Facts and Figures*. [Online]. Available: https://www.itu.int/en/ITU-D/Statistics/Documents/facts/FactsFigures2019.pdf

[2] Cisco, "Global mobile data traffic forecast update, 2017–2022," White Paper, Feb. 2019.

[3] Facebook. (Jan. 2019). *Facebook Reports First Quarter and Full Year 2018 Results*. [Online]. Available: https://s21.q4cdn.com/399680738/files/doc_financials/2018/Q4/Q4-2018-Earnings-Release.pdf

[4] Statista. (2020). *Statistics and Market Data on Mobile Internet and Apps*. [Online]. Available: https://www.statista.com/markets/424/topic/538/mobile-internet-apps/

[5] Allied Market Research. (2020). *Mobile Application Market Statistics–2026*. [Online]. Available: https://www.alliedmarketresearch.com/mobile-application-market

[6] B. Bergvall-Kåreborn and D. Howcroft, "Persistent problems and practices in information systems development: A study of mobile applications development and distribution," *Inf. Syst. J.*, vol. 24, no. 5, pp. 425–444, Sep. 2014.

[7] J. Berrocal, J. García-Alonso, and J. M. Murillo, "Architectures server-centric vs mobile-centric for developing WoT applications," in *Proc. 19th Int. Conf. Web Eng. (ICWE)*, in Lecture Notes in Computer Science, vol. 11496, M. Bakaev, F. Frasincar, and I. Ko, Eds. Daejeon, South Korea: Springer, Jun. 2019, pp. 578–581, doi: 10.1007/978-3-030-19274-7_48.

[8] K. De Moor, I. Ketyko, W. Joseph, T. Deryckere, L. De Marez, L. Martens, and G. Verleye, "Proposed framework for evaluating quality of experience in a mobile, testbed-oriented living lab setting," *Mobile Netw. Appl.*, vol. 15, no. 3, pp. 378–391, Jun. 2010.

[9] A. Vagrani, N. Kumar, and P. V. Ilavarasan, "Decline in mobile application life cycle," *Procedia Comput. Sci.*, vol. 122, pp. 957–964, 2017.

[10] A Technologies. (2014). *Android App Performance Report*. [Online]. Available: http://now.avg.com/wp-content/uploads/2015/02/avg_android_app_performance_report_q4_2014.pdf

[11] H. Qian and D. Andresen, "Extending mobile device's battery life by offloading computation to cloud," in *Proc. 2nd ACM Int. Conf. Mobile Softw. Eng. Syst.*, May 2015, pp. 150–151.

[12] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing," EECS Dept., Univ. California, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2009-28, 2009.

[13] A. Patidar and U. Suman, "A survey on software architecture evaluation methods," in *Proc. 2nd Int. Conf. Comput. Sustain. Global Develop. (INDIACom)*, Mar. 2015, pp. 967–972.

[14] A. Mokni, M. Huchard, C. Urtado, S. Vauttier, and H. Y. Zhang, "Towards automating the coherence verification of multi-level architecture descriptions," in *Proc. 9th ICSEA*, Nice, France, 2014, pp. 416–421.

[15] N. M. Devadiga, "Tailoring architecture centric design method with rapid prototyping," in *Proc. 2nd Int. Conf. Commun. Electron. Syst. (ICCES)*, Oct. 2017, pp. 924–930.

[16] Z. Hao, E. Novak, S. Yi, and Q. Li, "Challenges and software architecture for fog computing," *IEEE Internet Comput.*, vol. 21, no. 2, pp. 44–53, Mar. 2017.

[17] T. O. A. Lehtinen, M. V. Mäntylä, J. Vanhanen, J. Itkonen, and C. Lassenius, "Perceived causes of software project failures—An analysis of their relationships," *Inf. Softw. Technol.*, vol. 56, no. 6, pp. 623–643, Jun. 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950584914000263

[18] J. Škrabálek and C. Böhm, "Why modern mobile and Web-based development need a lean agile Web approach (lawa)," in *Proc. IDIMT*, 2013, p. 225.

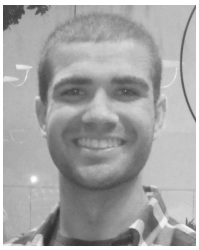
[19] B. Li, "Mobile app to promote and assist entrepreneurial activities," Ph.D. dissertation, Dept. Commerce, Law Manage., Univ. Witwatersrand, Johannesburg, Johannesburg, South Africa, 2018.
[20] C. Scharff and R. Verma, "Scrum to support mobile application development projects in a just-in-time learning context," in *Proc. ICSE Workshop Cooperat. Hum. Aspects Softw. Eng. (CHASE)*, 2010, pp. 25–31.
[21] G. Ortiz, A. García-de-Prado, J. Berrocal, and J. Hernández, "Improving resource consumption in context-aware mobile applications through alternative architectural styles," *IEEE Access*, vol. 7, pp. 65228–65250, 2019, doi: 10.1109/ACCESS.2019.2918239.
[22] L. Stone. (Sep. 2016). *Bringing Pokémon GO to Life on Google Cloud*. [Online]. Available: https://cloud.google.com/blog/products/gcp/bringing-pokemon-go-to-life-on-google-cloud
[23] T. Giridher, A. Bulchandani, R. Kim, P. Naik, A. Wasilewska, and J. L. Wong, "Social mobile applications," in *Proc. IEEE Long Island Syst., Appl. Technol. Conf.*, May 2010, pp. 1–6.
[24] G. Baumgarten, M. Rosinger, A. Todino, and R. de Juan Marin, "SPEM 2.0 as process baseline meta-model for the development and optimization of complex embedded systems," in *Proc. IEEE Int. Symp. Syst. Eng. (ISSE)*, Sep. 2015, pp. 155–162.
[25] X. Wang, "The combination of agile and lean in software development: An experience report analysis," in *Proc. AGILE Conf.*, Aug. 2011, pp. 1–9.
[26] D. Leffingwell, *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*. Reading, MA, USA: Addison-Wesley, 2010.
[27] T. F. V. D. Cunha, V. L. L. Dantas, and R. M. C. Andrade, "SLeSS: A scrum and lean six sigma integration approach for the development of sofware customization for mobile phones," in *Proc. 25th Brazilian Symp. Softw. Eng.*, Sep. 2011, pp. 283–292.
[28] J. P. Womack, D. T. Jones, and D. Roos, *The Machine That Changed the World: The Story of Lean Production–Toyota's Secret Weapon in the Global Car Wars That Is Now Revolutionizing World Industry*. New York, NY, USA: Simon and Schuster, 2007.
[29] K. Beck *et al.*, "Manifesto for agile software development," Tech. Rep., 2001. [Online]. Available: https://moodle2019-20.ua.es/moodle/pluginfile.php/2213/mod_resource/content/2/agile-manifesto.pdf
[30] R. Vallon, L. Wenzel, M. E. Brüggemann, and T. Grechenig, "An agile and lean process model for mobile app development: Case study into austrian industry," *J. Softw.*, vol. 10, no. 11, pp. 1245–1264, Nov. 2015.
[31] A. Ashishdeep, J. Bhatia, and K. Varma, "Software process models for mobile application development: A review," *Comput. Sci. Electron. J.*, vol. 7, no. 1, pp. 150–153, 2016.
[32] H. K. Flora and D. S. V. Chande, "A review and analysis on mobile application development processes using agile methodologies," *Int. J. Res. Comput. Sci.*, vol. 3, no. 4, pp. 8–18, Jul. 2013.
[33] K. Schwaber and M. Beedle, *Agile Software Development With SCRUM*. Upper Saddle River, NJ, USA: Prentice-Hall, 2001.
[34] A. Maurya, *Running Lean: Iterate from Plan A to a Plan That Works* (The Lean Series). Newton, MA, USA: O'Reilly Media, Inc., 2012.
[35] S. Oomen, B. De Waal, A. Albertin, and P. Ravesteyn, "How can scrum be succesful? Competences of the scrum product owner," in *Proc. 25th Eur. Conf. Inf. Syst. (ECIS)*, Guimaraes, Portugal, Jun. 2017. [Online]. Available: http://aisel.aisnet.org/ecis2017_rp/9?utm_source=aisel.aisnet.org%2Fecis2017_rp%2F9&utm_medium=PDF&utm_campaign=PDFCoverPages
[36] V. Lenarduzzi and D. Taibi, "MVP explained: A systematic mapping study on the definitions of minimal viable product," in *Proc. 42th Euromicro Conf. Softw. Eng. Adv. Appl. (SEAA)*, Aug. 2016, pp. 112–119.
[37] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 3rd ed. Reading, MA, USA: Addison-Wesley, 2012, p. 624.
[38] A. Sharma, M. Kumar, and S. Agarwal, "A complete survey on software architectural styles and patterns," *Procedia Comput. Sci.*, vol. 70, pp. 16–28, Jan. 2015.
[39] *M2AppInsight Quarterly App Performance Report: Global Android App Category Averages*, M2AppInsight, Aliso Viejo, CA, USA, Jun. 2015.
[40] *Avast AndroidTM App Performance and Trend Report*, Avast, Aliso Viejo, CA, USA, Jan./Apr. 2017.
[41] P. D. Berger and N. I. Nasr, "Customer lifetime value: Marketing models and applications," *J. Interact. Marketing*, vol. 12, no. 1, pp. 17–30, Jan. 1998.
[42] W. Jianxun, "A study on customer acquisition cost and customer retention cost: Review and outlook," in *9th Int. Conf. Innov. Manage.*, 2012, pp. 799–803.
[43] D. Skok. (Jan. 2018). *Calculating LTV and CAC for a SaaS Startup*. [Online]. Available: http://www.forentrepreneurs.com/saas-metrics-2-definitions-2/
[44] *Customer Lifetime Value to Customer Acquisition Ratio (CLV:CAC)*, Klipfolio, Ottawa, ON, Canada, Jan. 2018.
[45] K. Fietkiewicz. (Jan. 2018). *A Review of Monthly Operating Costs for a Startup's Engineering Infrastructure*. [Online]. Available: https://kubaf.wordpress.com/2013/10/05/an-analysis-of-monthly-operating-costs-for-a-startups-engineering-infrastructure/
[46] J. Berrocal, J. Garcia-Alonso, C. Vicente-Chicote, J. Hernández, T. Mikkonen, C. Canal, and J. M. Murillo, "Early analysis of resource consumption patterns in mobile applications," *Pervas. Mobile Comput.*, vol. 35, pp. 32–50, Feb. 2017.
[47] A. Carroll and G. Heiser, "The systems hacker's guide to the galaxy energy usage in a modern smartphone," in *Proc. 4th Asia–Pacific Workshop Syst. (APSys)*, 2013, pp. 1–7.
[48] J. Ruesch. (2013). *Android Async HTTP Clients: Volley vs Retrofit*. [Online]. Available: http://instructure.github.io/blog/2013/12/09/volley-vs-retrofit/
[49] K. Kussainov and B. Kumalakov, "Mobile data store platforms: Test case based performance evaluation," in *Proc. 8th Int. Joint Conf. Knowl. Discovery, Knowl. Eng. Knowl. Manage.*, 2016, pp. 95–99.
[50] Android. (Feb. 2019). *Capture and Read Bug Reports*. [Online]. Available: https://developer.android.com/studio/debug/bug-report
[51] Android Developer. (2020). *Profile Battery Usage With Batterystats and Battery Historian*. [Online]. Available: https://developer.android.com/topic/performance/power/battery-historian
[52] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering: An Introduction*. Norwell, MA, USA: Kluwer, 2000.
[53] N. Vallina-Rodriguez and J. Crowcroft, "Energy management techniques in modern mobile handsets," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 1, pp. 179–198, 1st Quart., 2013.
[54] L. Ferri, M. Maffei, G. Mangia, and A. Tomo, "Analyzing cloud-based startups: Evidence from a case study in Italy," *Int. Bus. Res.*, vol. 10, no. 5, pp. 73–85, 2017.
[55] P. Gupta, A. Seetharaman, and J. R. Raj, "The usage and adoption of cloud computing by small and medium businesses," *Int. J. Inf. Manage.*, vol. 33, no. 5, pp. 861–874, Oct. 2013.
[56] M. Hasan and S. Hossain, "An approach to estimating cloud resources prior to new business startup," *Amer. Sci. Res. J. Eng., Technol., Sci.*, vol. 26, no. 3, pp. 172–187, 2016.
[57] J. M. García, O. Martín-Díaz, P. Fernandez, A. R. Cortés, and M. Toro, "Automated analysis of cloud offerings for optimal service provisioning," in *Proc. 15th Int. Conf. Service-Oriented Comput. (ICSOC)*, in Lecture Notes in Computer Science, vol. 10601, E. M. Maximilien, A. Vallecillo, J. Wang, and M. Oriol, Eds. Malaga, Spain: Springer, Nov. 2017, pp. 331–339, doi: 10.1007/978-3-319-69035-3_23.
[58] J. García-Galán, P. Trinidad, O. F. Rana, and A. Ruiz-Cortés, "Automated configuration support for infrastructure migration to the cloud," *Future Gener. Comput. Syst.*, vol. 55, pp. 200–212, Feb. 2016, doi: 10.1016/j.future.2015.03.006.
[59] A. Pathak, Y. C. Hu, and M. Zhang, "Where is the energy spent inside my app?: Fine grained energy accounting on smartphones with eprof," in *Proc. 7th ACM Eur. Conf. Comput. Syst. (EuroSys)*, 2012, pp. 29–42.
[60] A. A. Moamen and N. Jamali, "Share sens: An approach to optimizing energy consumption of continuous mobile sensing workloads," in *Proc. IEEE Int. Conf. Mobile Services*, Jun. 2015, pp. 89–96.
[61] W. Jung, C. Kang, C. Yoon, D. Kim, and H. Cha, "DevScope: A nonintrusive and online power analysis tool for smartphone hardware components," in *Proc. 8th IEEE/ACM/IFIP Int. Conf. Hardw./Softw. Codesign Syst. Synth. (CODES+ISSS)*, 2012, pp. 353–362.
[62] M. Curti, A. Merlo, M. Migliardi, and S. Schiappacasse, "Towards energy-aware intrusion detection systems on mobile devices," in *Proc. Int. Conf. High Perform. Comput. Simulation (HPCS)*, Jul. 2013, pp. 289–296.
[63] C. Yoon, D. Kim, W. Jung, C. Kang, and H. Cha, "Appscope: Application energy metering framework for Android smartphone using kernel activity monitoring," in *Proc. USENIX Annu. Tech. Conf.*, 2012, pp. 387–400.
[64] H. S. A. Al Nidawi, K. T. Wei, K. A. Dawood, and A. Khaleel, "Energy consumption patterns of mobile applications in Android platform: A systematic literature review," *J. Theor. Appl. Inf. Technol.*, vol. 95, no. 24, pp. 6776–6787, Dec. 2017.

[65] S. Chowdhury, S. Borle, S. Romansky, and A. Hindle, "GreenScaler: Training software energy models with automatic test generation," *Empirical Softw. Eng.*, vol. 24, no. 4, pp. 1649–1692, Aug. 2019, doi: 10.1007/s10664-018-9640-7.
[66] S. Chowdhury, S. Di Nardo, A. Hindle, and Z. M. Jiang, "An exploratory study on assessing the energy impact of logging on Android applications," *Empirical Softw. Eng.*, vol. 23, no. 3, pp. 1422–1456, Jun. 2018, doi: 10.1007/s10664-017-9545-x.
[67] X. Chen, J. Chen, B. Liu, Y. Ma, Y. Zhang, and H. Zhong, "Android-Off:Offloading Android application based on cost estimation," *J. Syst. Softw.*, vol. 158, Dec. 2019, Art. no. 110418.

JAVIER BERROCAL received the Ph.D. degree in computer science from the University of Extremadura, Spain, in 2014. In 2016, he obtained an Associate position at the University of Extremadura. His main research interests are mobile computing, context awareness, pervasive systems, crowd sensing, the Internet of Things, and fog computing. He is currently a Co-Founder of the company Gloin, which is a software-consulting company.

JOSE GARCÍA-ALONSO received the Ph.D. degree in software engineering from the University of Extremadura, in 2014. He is currently an Associate Professor with the University of Extremadura and a Co-Founder of Gloin, a software-consulting company. His research interests include software engineering, mobile computing, pervasive computing, and eHealth, gerontechnology.

PABLO FERNANDEZ received the Ph.D. degree in computer science from the University of Sevilla, Spain. He is currently a Lecturer and a member of the Applied Software Engineering Group (ISA), University of Sevilla. His current research is focused on the automated governance of organizations based on service level agreements and commitments.



ALEJANDRO PÉREZ-VEREDA is currently pursuing the Ph.D. degree with the University of Malaga, Spain. He is also working with the Languages and Computer Science Department, University of Malaga. His research interests are mobile computing, context-awareness, pervasive systems, crowd sensing, and the Internet of Things.

JUAN HERNANDEZ (Associate Member, IEEE) received the Ph.D. degree in computer science from the Technical University of Madrid, Spain, in 1995. He is currently a Full Professor at the University of Extremadura and the Head of the Quercus Software Engineering Group. His research interests include service-oriented computing, ambient intelligence, and model-driven development.

CARLOS CANAL received the Ph.D. degree in computer science from the University of Malaga, Spain. He is currently a Full Professor with the University of Malaga. His research interests include mobile and cloud development.

JUAN MANUEL MURILLO received the Ph.D. degree in computer science from the University of Extremadura. He is currently a Co-Founder of Gloin and a Full Professor with the University of Extremadura. His research interests include software architectures, mobile computing, and cloud computing.

ANTONIO RUIZ-CORTES (Member, IEEE) is currently a Full Professor of software and service engineering and also heads the Applied Software Engineering Group, University of Sevilla. His current researches focus on service-oriented computing, business process management, testing, and software product lines. He is an Associate Editor of *Computing* (Springer).