SN Computer Science manuscript No.
(will be inserted by the editor)

A Survey on Mobile Applications for Smart Agriculture

Making Use of Mobile Software in Modern Farming

*‡**Isaac Nyabisa Oteyo** · ‡**Matteo Marra** · †**Stephen Kimani** · ‡**Wolfgang De Meuter** · ‡**Elisa Gonzalez Boix**

Received: date / Accepted: date

Abstract The increasing global demand for food and nutrition security has raised the need to automate processes in modern farming. As such, a promising way to automate those processes is by using smart agriculture applications (SAAs). Different studies in the literature classify these applications based on agricultural themes, agricultural domains, and farming scenarios. However, this classification is not sufficient for researchers and industry to gain deeper insights on software engineering issues pertaining to SAAs. In this survey, we explore SAAs and further classify them based on architectural models, supported software engineering issues, and target mobile platforms. The survey results show that SAAs in general (i) follow different architectural models, (ii) are targeted for different mobile platforms, and (iii) satisfy different software engineering issues. Most importantly, the key findings from this study reveal that SAAs can fail to meet their intended purpose if developers ignore key software engineering issues. These findings can be used as a starting point for researchers and industry to implement smart agriculture related mobile applications.

Keywords mobile applications · cloud computing · smart farming · internet of things · smart agriculture applications

*Corresponding author. E-mail: isaac.nyabisa.oteyo@vub.be; *Tel.:* +32 484 739 336;

‡I. N. Oteyo (*ORCID iD:* 0000-0002-2682-605X)
· ‡M. Marra (*ORCID iD:* 0000-0002-8037-0567)
· ‡W. De Meuter (*ORCID iD:* 0000-0002-5229-5627)
· ‡E. Gonzalez Boix (*ORCID iD:* 0000-0002-9966-6421)
Software Languages Lab; Department of Computer Science; Vrije Universiteit Brussel; Pleinlaan 2, 1050, Elsene, Brussels, Belgium

†S. Kimani
School of Computing and Information Technology; Jomo Kenyatta University of Agriculture and Technology; Postal address: 62000 00200, Nairobi, Kenya

1 Introduction

Recently, efforts are being harnessed to increase food production to address food and nutrition security [**?**,**?**,**?**,**?**]. This has put a demand for automation of processes in modern farming to improve farm efficiency and increase productivity [**?**,**?**]. Broadly, the processes in modern farms as identified by Wolfert *et al.* [**?**] can be condensed into four categories [**?**]: (i) data collection, (ii) data transformation and processing, (iii) data dissemination, and (iv) evaluation and impact. In practice, these processes are a chain of systematic, repetitive, and time-dependent tasks that can be performed on devices via smart agriculture applications (SAAs) [**?**,**?**]. These applications can function as standalone or as distributed applications exploiting technologies such as Internet of Things, cloud computing etc.

In the latest years, SAAs have become popular because of their versatility to be used in one or more of the smart agriculture processes [**?**,**?**,**?**,**?**,**?**]. In fact, SAAs can be used to collect data, inserted directly by farmers in the applications or in communication with sensors, to process such data and provide indicative suggestions on the required resources to the farmers, as well as allowing them to easily interact with farm equipment and machinery [**?**]. Whereas, using SAAs can improve the data collection process, such data collection can happen in remote and rural areas that are characterised with no internet access or poor quality connections [**?**]. The collected data can be compared with set thresholds to trigger indicative suggestions when the limits are exceeded. These indicative suggestions can be sent to the end-users as timely notifications via SAAs. Additionally, the different farming activities may require these applications to be adaptable for different functions in the farm. The above aspects should be part of the software engineering design considerations for SAAs.

Different studies in the literature have attempted to classify these applications based on agricultural themes, agricultural domains, application scenarios, and agricultural functions [**?**,**?**,**?**,**?**,**?**,**?**,**?**,**?**]. However, to the best of our knowledge, none of the existing studies classify these applications based on architectural models, target platforms, and software engineering issues. As such, the existing classification is not sufficient for researchers and industry to gain deeper insights on software engineering issues pertaining to SAAs. For example, it is important to understand how different SAAs handle network connection issues in farms with poor quality connections.

In this paper, we combine the criteria used in the existing literature with architectural models, software engineering issues, and target mobile platforms to survey SAAs. In particular, in our proposed classification, we group SAAs into the following categories: (i) 1-tier (standalone), (ii) 2-tier, (iii) 3-tier, and (iv) edge computing applications. Furthermore, we identify three issues that should be addressed by developers of SAAs: (i) offline accessibility, (ii) reactivity, and (iii) reconfigurability. Offline accessibility allows SAAs to continue functioning when there is no internet at all or the connection is limited due to poor quality. On the other hand, reactivity allows SAAs to generate and send timely notifications to the end users. Lastly, reconfigurability allows end-users to add new services to SAAs or extend existing services and adapt them for different farming activities. In this survey, we show how different SAAs have addressed the above issues and identify opportunities for further research. The survey also investigates the mobile platforms that SAAs target. Lastly, the survey findings provide a state-of-the-art snapshot on classifying SAAs and insights in the domain. We believe that those insights can help researchers and industry towards the implementation of SAAs. The rest of the paper is organised as follows. Section 2 gives the motivation for this study and describes the state-of-the-art, while Section 3 presents our classification of SAAs and discussions. This is followed by Section 4 that presents the open issues and gives directions for further research. Lastly, Section 5 presents our conclusions.

2 Motivation and Background

Smart agriculture refers to the incorporation of information and communication technologies into modern farming processes for improved management of farm activities [**?**,**?**,**?**]. In literature, the terms agriculture 4.0 [**?**,**?**], facility agriculture [**?**,**?**], order agriculture [**?**], smart agriculture/farming [**?**,**?**], precision agriculture/farming [**?**,**?**], digital agriculture/farming [**?**], and intelligent agriculture/farming [**?**] have been used interchangeably. Other terms that have been used to refer to smart agriculture include [**?**]: (i) prescription farming, (ii) farming-by-the-foot, (iii) site-specific crop management, (iv) satellite farming, and (v) precision livestock farming. From the definition, the unifying factor for all these terms is using information and communication technologies in "farm management concepts". In this article, we define *smart agriculture applications* (SAAs) as mobile software applications that can be used in any of the smart agriculture processes such as data collection, processing and dissemination as illustrated in Figure 1. As such, in this article, we focus on SAAs that can execute on mobile devices.

2.1 Smart Agriculture Processes

In smart agriculture, farm processes are managed on a site-specific basis; farm inputs are applied at variable rates at each site on a farm field depending on the agronomic needs of that particular location. As such, for the purpose of this paper, we summarise the processes in modern farming as identified by Wolfert *et al.* [**?**] into four categories as illustrated in Figure 1, namely: (i) data collection, (ii) data transformation and processing, (iii) data dissemination, and (iv) evaluation and impact.

Data collection processes can use sensors that are connected within modern farms e.g., motion cameras to capture crop images or temperature and humidity sensors to monitor environmental conditions. The data can also be entered directly by farmers manually via forms or dedicated applications.

Data transformation and processing converts raw data into meaningful information e.g., converting accelerometer values collected from wearable devices to determine animal location and movements. This transformation and processing relies on statistical tools available on data analytic platforms. Additionally, the data transformation and processing may happen using big data frameworks and techniques to process high volumes of data. It can further include agricultural prediction using machine learning techniques. Lastly, this process can be performed locally or in the cloud.

Data dissemination serves to sensitise different end-users on the kind of meaning and value they can derive from the processed data. The data dissemination process entails using web dashboards, mobile applications, and interactive maps to present processed data to the end-users. Lastly, the dissemination can also be done via alerts or notifications.

Evaluation and impact is performed to ascertain and give perspective to farming activities. This serves to provide feedback and inform the farmer on subsequent data collection phases e.g., the dimension, volume, and quality of data to be collected.

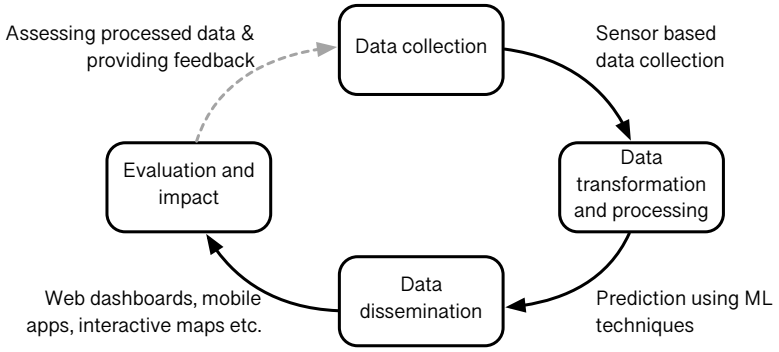

Fig. 1: Key smart agriculture processes based on Wolfert *et al.* [**?**].

SAAs are diverse and can target one or more of the above processes.

2.2 Software Engineering Issues in SAAs

Recall from Section 1 that for this work, there are three software engineering issues that developers should address when programming SAAs: (i) offline accessibility, (ii) reactivity, and (iii) reconfigurability.

Offline accessibility: Modern farming activities can happen in remote and rural areas. As noted in the literature, these areas continue to experience a digital-divide in terms of internet and broadband access [**?**]. For instance, the penetration of broadband access in rural areas for developing regions has been noted to be significantly low and of poor quality [**?**]. As a result, using SAAs that require internet connectivity in such areas can be challenging. This issue can limit and make the end-users shy away from adopting and using SAAs in their farming activities. As such, to support distributed applications from a software engineering perspective, some SAAs need to be offline accessible, hence capable to continue functioning when the network connection is not available or the connection quality is poor.

Reactivity: While using SAAs, end-users may expect to receive feedback on farming activities. For instance, when collecting data using remote sensors, the end-users may need to receive timely notifications when the sensors go off or when other crucial farming activities require some attention such as when crops need pesticides, fertilisers, and irrigation [**?**,**?**]. The end-users can utilise the received notifications to take decisions such as turning on irrigation systems when the soil-water levels drop below certain thresholds or applying fungicides to crops. To handle such notifications in a timely way, SAAs require to be reactive.

Reconfigurability: SAAs may require new services to enrich the existing functionalities. Additionally, SAAs may require adapting the existing functionalities to different farming scenarios. For instance, the workload conditions for SAAs may change such as connecting to new data sources that never existed before. Such changes can often require adding new functionalities to SAAs or re-adapting the existing functionalities. As such, SAAs require extensibility features such as APIs to add new functionalities and configurability features such as widgets that can be used to adapt existing functionalities to different scenarios. Additionally, modern farming activities can vary based on context e.g., collecting data for dairy cows and on a different date collecting data on dairy goats. In this regard, the end-users can use widgets to reconfigure data collection forms that suits both situations. In order to support these requirements, some SAAs should be reconfigurable.

Different architectural models exists in software engineering to handle the above issues differently.

2.3 Architectural Models for SAAs

SAAs consist of interconnected components that can handle different application concerns such as information presentation and display, data fetching and manipulation, and data storage. These components can be layered based on their functionalities from which architectural models for SAAs can be derived. These layers can be broadly grouped into: (i) presentation layer, (ii) business layer, and (iii) database layer. The presentation layer communicates and passes information given by the end-user to the business layer. On the other hand, the business layer performs the business logic by fetching data from the database layer and manipulating the data based on the specified business rules. In fact, the business logic performed in this layer controls the application functionality. Additionally, this layer can perform computations based on input that received from the end-users via the presentation layer. Lastly, the database layer performs stor-

age functionalities for both raw and processed application data.

The architectural models for SAAs are similar to the ones found in distributed systems. In this article, we base ourselves on the architectural roles from Tanenbaum and Van Steen [**?**] that we later use to classify SAAs. As such, depending on where the above layers are deployed, SAAs present different architectural models, namely: (i) 1-tier (standalone), (ii) 2-tier, (iii) 3-tier, and (iv) edge computing models.

1-tier (standalone) model: In this model, the presentation, business, database layers are contained in a single application. The data in this model is stored in the local system. Applications that follow this model often execute entirely on mobile devices. As such, these applications do not require network connections to perform their functionalities.

2-tier model: This model has two tiers i.e., the client-tier and the server-side tier illustrated in Figure 2 as *Tier-1* and *Tier-2* respectively. In Figure 2(a) the client-tier contains only the presentation layer, while in Figure 2(b), Figure 2(c) and Figure 2(d), the client-tier contains both the presentation and business layers. In Figure 2(d), the client-tier contains a database layer that executes on the client-side. The database tier serves as the server that receives client requests, processes them, and sends back responses to the clients. Generally, in this model, the database tier becomes the server to the client tier. The business and data logic can be stored at either the client-side or the server-side. When the business and data logic are stored on the server-side, the architecture is referred to as thin-client fat-server model (Figure 2(a)). Similarly, when the business and data logic are stored on the client-side, the architecture is referred to as fat-client thin-server model (Figure 2(b), Figure 2(c) and Figure 2(d)). This model can be useful where clients talk directly to the servers with no intervening server (middleware). Data processing can be split between the user interface environment and the database management server environment.

3-tier model: This model has the client tier, middle tier, and the database tier. The client tier handles the presentation and application layer, while the middle tier handles the business layer and application server, and lastly the database tier handles data storage as illustrated in Figure 3. In this model, both the application and database tiers become servers to the client tier. As illustrated in Figure 3, the middle tier runs on the application server and sits in between the client and the server. Client requests go to the server through this middle tier. Similarly, responses from the server first go through the middle tier before they can reach the client. Additionally, the business and data access logic can be stored in the middle tier; and if there are multiple business and data access logic, then it is referred to as *n-tier* architecture. The middle tier performs various tasks such as database staging, queuing/scheduling tasks, and executing applications. Though this model improves flexibility and gives better performance, the development environment is more difficult to use.

Edge computing model: This model is a variation of the 3-tier model that brings computational resources closer to the end-users. In particular, the model brings data processing closer in proximity to the sources of data generation. The data processing can be performed in the same devices that are used for data collection or in computing infrastructure that is close to the devices used in data collection. This model is aimed at reducing the number of requests made to and responses received from distant servers over communication networks.

The SAAs that follow the 2-tier, 3-tier, and edge computing model are generally client/server applications.

2.4 Target Platforms for SAAs

From our definition of SAAs, these applications can be targeted to different platforms such as Android, iOS, Windows mobile etc. In this work, we note that the choice of a target platform for SAAs depends on a number of factors such as existing mobile operating systems, mobile software development kits, mobile device models, and existing mobile application development technologies. In the case for development kits, there not only exists dedicated software development kits (SDKs) that are unique for particular platforms, but also cross platform development kits and technologies. The cross platform kits are based on the principle of 'develop once, and deploy many'. In addition, some cross platform technologies such as HTML5 yield hybrid applications. Also, some technologies such as mobile-first technologies for developing mobile-web applications require that SAAs should be designed for smaller devices first before scaling up and adapting to larger devices as responsive mobile designs. For mobile device models, it is worth to note that some device models like iPhones are only meant for iOS applications.

Lastly, in terms of space and time dimensions, some platforms are well adopted in some regions than others. For example, the Android platform has been noted to be more popular in developing regions [**?**,**?**]. Application developers should the take the above factors into considerations when programming SAAs.

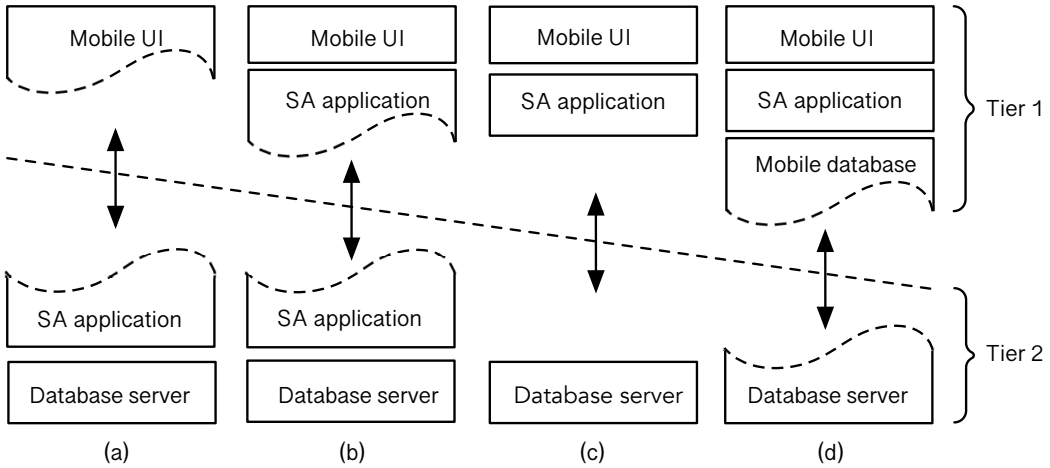

Fig. 2: 2-tier architectural models for SAAs adapted from Tanenbaum and Van Steen [**?**]. The arrows show the communication happening between the tier 1 and tier 2. This figure depicts 2-tier thin client fat server and fat client thin server applications.

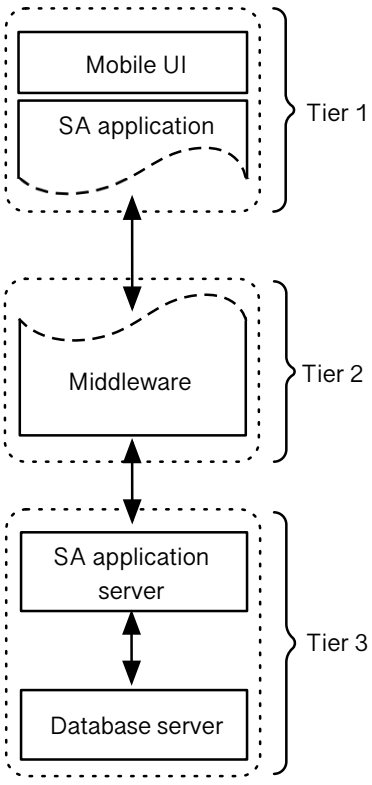

Fig. 3: 3-tier architectural model adapted from Tanenbaum and Van Steen [**?**] with arrows showing the communication happening between the different tiers.

2.5 Existing Classifications

From the literature surveyed, existing studies classify SAAs based on agricultural themes, domains, and functions [**?**,**?**,**?**,**?**,**?**,**?**,**?**,**?**].

Ayaz *et al.* [**?**], Minet *et al.* [**?**], and O'Grady *et al.* [**?**] base their classifications on agricultural domains such as irrigation management, weed and pest control, yield prediction, animal welfare, aqua farming etc. Ayaz *et al.* [**?**] note that Internet of Things sensors can be used to collect data on important farm characteristics such as soil type, nutrient presence, flow of irrigation, pest resistance etc. Using sensors can lead to achieving higher accuracy levels in data collection for timely reporting. The higher accuracy levels of collected data can then be used to detect early stages of unwanted states in modern farming activities. Minet *et al.* [**?**] show that different agricultural data can be collected through crowdsourcing such as land-use, soil, weather, yields, prices, and crop phenological data. However, the authors note that crowdsourcing applications in agriculture can be hindered by privacy issues. O'Grady *et al.* [**?**], in addition to classifying SAAs, show how different edge computing techniques (e.g., latency-sensitive analytics, edge mining for data compression, reducing data traffic and latency, and offloading computation) have been used in different agricultural domains such as animal surveillance, forest fire detection etc. Lastly, Bacco *et al.* [**?**] suggest application scenarios that can be used to classify SAAs such as crop health monitoring, yield prediction, weed mapping, soil assessment etc. The study focused on digitising agriculture without delving much into the actual classification of SAAs. The study findings show that pushing technology into rural areas can encounter several challenges such as data security, data privacy, and network capabilities.

Pongnumkul *et al.* [**?**], Inwood *et al.* [**?**], Barh and Balakrishnan [**?**], and Patel and Patel [**?**] base their classifications on agricultural functions such as farm management, extension services, journaling farming activities, equipment

tracking, managing human resource in farms etc. In their study, Pongnumkul *et al.* [**?**] focused on applications that use built-in smartphone sensors to provide agricultural sensing solutions. The authors note that when designing applications that utilise smartphone sensors, requirements and behaviours of end-users can be different across different farming communities. For instance, when designing applications for pest prediction, alerts can be necessary to notify the end-users on pest outbreaks. Inwood *et al.* [**?**] focused on tools for information provisioning and sharing that can connect decision-makers to knowledge in support of sustainable agriculture, while Barh and Balakrishnan [**?**] examined how smartphone mobile applications can help in agricultural development. Additionally, Barh and Balakrishnan [**?**] argue in support of SAAs as being part of the transformative technologies that can be used to improve productivity in modern farming especially in developing regions. Lastly, Patel and Patel [**?**] observe that most of the SAAs are used in farm management activities.

2.6 Our Taxonomy

Based on the above state-of-the-art analysis, the existing studies do not consider the architectural models, target mobile platforms, and software engineering issues as classification criteria for SAAs. From an application design and development point of view, classifying SAAs based on architectural models can help researchers and industry in implementing SAAs targeting different mobile platforms for different modern farming scenarios. As such, the architectural models can help determine how to structure different parts of SAAs and how these different parts can seamlessly talk to each other. Moreover, classifying these applications based on how the previously mentioned software engineering issues are handled can help in understanding the features and limitations of SAAs. In particular, some SAAs might fit well to the end-user requirements and some may be better than others. Furthermore, the software engineering aspects can help developers give proper attention to required application properties for different farming scenarios and activities. Additionally, the classification based on software engineering issues can help end-users to evaluate a set of alternative applications to identify a candidate target application based on their needs. In conclusion, in this survey, we classify the state of the art based on (i) the architectural model of the applications, (ii) the targeted platform, and (iii) how they handle the identified software engineering issues.

3 Classifying Smart Agriculture Applications

In this section, we combine the agricultural themes, domains, architectural models, target platforms, and software engineering issues to classify SAAs as shown in Table 1. The surveyed applications cover different agricultural themes e.g., irrigation management, geographic information systems, fertiliser management etc. Furthermore, the applications also cover different agricultural domains e.g., irrigation scheduling, leaf area index estimation, soil assessment etc. Most of the applications surveyed (at 71.70%) are based on the client/server model, while few applications (at 28.30%) are based on the standalone (*1-tier*) model. There are more SAAs that follow the client/server model because the model is scalable.

From the literature surveyed, some 2-tier and 3-tier components such as application and database servers can execute in the cloud to support cloud computing applications. Based on their nature, these applications are best suited in scenarios that require large storage spaces, advanced cloud-based data analytics and computational capabilities that cannot be readily available on mobile devices. Most of the client/server SAAs surveyed are based on the 2-tier model, while few applications often follow the edge computing and 3-tier models.

3.1 Classification Based on Architectural Models

Table 1 shows classified SAAs based on standalone and client/server architectural models. As mentioned before, the client/server model encompasses 2-tier, 3-tier, and edge computing models that were described in Section 2.3.

3.1.1 1-tier Applications

From the literature surveyed, 1-tier (standalone) SAAs serve different agricultural themes such as irrigation management [**?**,**?**,**?**], pest control and management [**?**,**?**,**?**], and risk management [**?**]. The examples of these applications are as indicated in Table 1. Some of the SAAs in this category are used to estimate water needs for crops. Such applications are implemented to exploit sensors in mobile phones such as phone cameras, accelerometer, ambient light sensors etc. For instance, PocketLAI [**?**,**?**], VitiCanopy [**?**,**?**,**?**], and pCAPS [**?**] use the phone cameras to capture crop images that are processed within the applications to determine water requirements for crops. On the other hand, SmartfLAIr [**?**] uses the ambient light sensor in phones to estimate crop leaf area index which is then used to estimate the water requirements for crops.

Other applications in this category serve as decision support tools to farmers. For instance, AgroDecisor EFC [**?**] that recommends to farmers on proper application of fungicides by comparing the user input against threshold. On the other hand, SnapCard [**?**] and DropLeaf [**?**] use image analysis techniques to compare fungicide spray nozzles against end-user specifications. The analysed outcome is then used to recommend to the farmer whether to apply fungicide or

Table 1: Classifying smart agriculture application based on themes, domains, architectures, and target mobile platforms.

Application	Agricultural theme	Agricultural domain	Architectural model	Target mobile platform
PocketLAI [?,?]	Irrigation management	Estimating crop water needs	1-tier	Android
WISE [?]	Irrigation management	Irrigation scheduling	2-tier	iOS
EVAPO [?]	Irrigation management	Evapotranspiration estimation	2-tier	Android
PIS [?]	Irrigation management	Soil moisture sensing	2-tier	iOS, Android
SoilWaterApp [?]	Irrigation management	Soil moisture monitoring	2-tier	iOS
RaGPS [?]	Irrigation management	Extraterrestrial solar radiation	1-tier	Windows
SmartIrrigation [?]	Irrigation management	Irrigation scheduling	2-tier	iOS, Android
Crop Water Stress [?]	Irrigation management	Vine water stress	2-tier	Android
pCAPS [?]	Irrigation management	Crop water needs	1-tier	Android
VitiCanopy [?,?,?]	Irrigation management	Plant water requirement	1-tier	iOS, Android
SmartfLAIr [?]	Yield management	Crop yield estimation	1-tier	Android
PETEFA [?]	Geographic information systems	Geo-referenced soil analysis	2-tier	Android
eFarm [?]	Geographic information systems	Geo-tagging land data	2-tier	Android
PAMS [?]	Geographic information systems	Managing farmland spatial data	2-tier	iOS, Android, Windows
AMACA [?]	Farm machinery/tools	Estimating machinery cost	2-tier	iOS, Android
Ecofert [?]	Fertiliser management	Crop fertiliser estimation	2-tier	Android
cFertigUAL [?]	Fertiliser management	Crop fertiliser estimation	2-tier	Android
BaoKhao [?]	Fertiliser management	Leaf colour estimation for N fertiliser	1-tier	Android
SnapCard [?]	Weed and pest control	Crop spraying	1-tier	Android
VillageTree [?]	Weed and pest control	Gathering pest incidence reports	2-tier	iOS, Android
MobiCrop [?,?]	Weed and pest control	Sharing information on pesticides	3-tier	iOS
AgroDecisor EFC [?]	Weed and pest control	Fungicide treatment estimation	1-tier	Android
DropLeaf [?]	Weed and pest control	Crop health	1-tier	Android
AgriMaps [?]	Land management	Crop and land management recommendation	2-tier	Android
LandPKS [?]	Land management	Soil assessment	2-tier	Android
SOCiT [?]	Land management	Soil assessment	2-tier	Android
SIFSS [?]	Land management	Soil assessment	2-tier	Android
GeoFoto [?]	Land management	Land parcel identification	2-tier	Android
MapIT [?]	Farm management	Equipment tracking	2-tier	Android
SafeDriving [?]	Farm management	Equipment tracking	1-tier	iOS
FarmManager [?]	Farm management	Capturing farm data	2-tier	Android
SmartHof [?]	Animal welfare	Monitoring animal health	Edge computing	Android
SmartFarm [?]	Animal welfare	Monitoring animal health	Edge computing	Android
Agri-IoT [?]	Animal welfare	Livestock fertility management	2-tier	–
BioLeaf [?]	Crop health	Leaf health monitoring	1-tier	Android
Plant Disease [?,?]	Crop health	Plant disease diagnosis	1-tier	iOS, Android, Windows
Canopeo [?]	Crop health	Estimating canopy development	2-tier	iOS, Android
vitisFlower [?]	Crop health	Flower assessment	1-tier	Android
vitisBerry [?]	Crop health	Berry assessment	1-tier	Android
FruitSize [?]	Crop health	Fruit size assessment	2-tier	Android
PulAm [?]	Crop health	Crop pest monitoring	2-tier	Android
UbiQON [?]	Greenhouse monitoring	Oyster mushroom monitoring	2-tier	Android
Blynk [?]	Storage monitoring	Paddy rice monitoring	2-tier	iOS, Android
iDee [?]	River monitoring	Water assessment	2-tier	Android
ConnectedFarm [?]	Environment monitoring	Remote monitoring and control	2-tier	Android
SmartFarmKit [?]	Environment monitoring	Oyster mushroom and maize monitoring	2-tier	–
Nitrogen Index [?]	Environment monitoring	Nitrogen monitoring	1-tier	Android
WheatCam [?]	Risk management	Crop insurance	2-tier	Android
SMILEX [?]	Risk management	Tracking sick plants	2-tier	Android
AgDataBox [?]	Data collection	Field data recording	2-tier	Android
IRIS [?]	Data collection	Field data recording	2-tier	–
iFarm [?]	Data collection	Field data recoding	2-tier	–
GeoFarmer [?]	Data collection	Field data recording	2-tier	Android

not. BioLeaf [?] exploits image processing to measure foliar damage in crop leaves, which farmers can also use to determine whether to apply fungicides or not. Similarly, Plant Disease [?,?] uses image analysis to help end-users detect crop diseases against predefined disease signatures. Additionally, SAAs such as vitisFlower [?] and vitisBerry [?] use image analysis to help farmers count the number of flowers for grapes and berries respectively. This count can be useful in predicting possible crop yields.

Lastly, Nitrogen Index [?] application can be used to assess the risk of nitrogen losses. The application allows on-site analysis of nitrogen levels to reduce the potential risk of nitrate leaching. The analysis is done via nitrogen leaching simulations on the application. The simulations are saved on the device storage in XML-based file format. Assessed nitrogen loss risk is correlated with observed values and the results are saved locally on the device.

3.1.2 2-tier Applications

As previously mentioned, applications that follow this model have a client-tier and database tier. These applications can serve different functions in smart agriculture such as data

presentation and display, data fetching, and information dissemination. Other SAAs in this category are multipurpose and can serve different functions in the smart agriculture processes as illustrated in Figure 1.

Data presentation and display: Some of the SAAs that use this model such as WISE [**?**] and AMACA [**?**] can be used as data monitors to display information to the end-users. The displaying can be done either on the applications themselves or web interfaces. Thus, such SAAs should have comprehensive graphical user interfaces that can be useful to end-users in viewing farming information such as soil moisture levels, location of farm machines etc. Furthermore, to get value out of the displayed information, some of the SAAs permit data input from the end-users. Meaningful information can be derived when the input data is compared to the application data. One such application that performs such comparison is EVAPO [**?**]. This application receives location coordinates as end-user input and performs irrigation scheduling after correlating the user input with weather data fetched from a server. On the other hand, SoilWaterApp [**?**] communicates to online climate, crop, and soil database to fetch data that it compares to on-farm data to generate meaningful infographics. Generating these infographics is done within the application. As such, the application can allow simulations on crop water balance levels to be performed on smartphones or tablets. In particular, the application uses multiple threads to perform parallel processing of the received data. The processed data is then saved to a cloud-hosted database which is periodically synchronised for updates.

Data fetching: In order to fetch data from the server, some SAAs such as SmartIrrigation [**?**], PETEFA [**?**], and Ecofert [**?**] employ pull-based techniques. The data is pulled at regular intervals to make it real-time i.e., almost at the rate at which it is generated from weather stations and pushed to the cloud. The received data can be used to trigger different services within the application such as generating notifications. For instance, in SmartIrrigation [**?**], a notification is generated and sent to the end-user whenever rainfall data that is recorded at the nearest weather station is received. On the other hand, in PETEFA [**?**], field images pulled from a geoserver are used to visualise information on evapotranspiration. Additionally, in some SAAs such as Ecofert [**?**] and cFertigUAL [**?**], data is pulled from cloud-hosted databases to help end-users compute the amount of fertiliser and water that can be applied to different crops. Such computations are often performed within the applications.

Data dissemination: On the issue of data dissemination, some SAAs such as LandPKS [**?**] uses mobile phones to exchange knowledge and information. Furthermore, this particular application uses cloud computing to integrate, interpret, and access relevant knowledge and information about land with similar potential. Also, the application primarily supports icon-based end-user inputs that are uploaded to the cloud databases. These databases provide input for predictive models. On the other hand, GeoFoto [**?**] uses smartphone sensors like GPS or camera to collect data and broadcast it to a remote server for analysis in real-time. The data collected by this application can be used in identifying farm fields. The real-time communication ensures and supports verifying that the end-user collecting data has visited the correct farm fields or ascertain that the sufficient number of data samples has been collected.

Multi-process applications: These applications are at the intersection of different domains and smart agriculture processes. For instance, AgDataBox [**?**] allows farmers in rural areas to collect and send environmental data for storage remotely. The application can also serve as a support tool for recording rainfall and scheduling tasks. Furthermore, as a software tool, this application was initially implemented as an API testing tool to send requests and receive responses depending on the selected operation. On the other hand, IRIS [**?**] is an integrated smart agriculture application with a web and mobile application. The application measures ambient parameters below the soil, crop level, and the ambient environment. Collected data is sent to the cloud for processing and visualisation is done on the web dashboard. End-users can then use the mobile application to view the processed and visualised data. These applications focus and emphasise on real-time monitoring of environmental conditions. In, GeoFarmer [**?**] data collection surveys are pre-created and then assigned to registered end-users. The collected data include spatial observations. The application uses local phone storage to support offline accessibility. When the application is online the central database and the local phone storage can be synchronised, and after which the application can be used offline.

Other multipurpose SAAs include eFarm [**?**] and VillageTree [**?**]. The eFarm [**?**] application can be used by the end-users to collect geo-tagged agricultural land system information based on remotely sensed images. The main functionalities of this application include visualising base-maps, data management (both for land parcels and users), and data sensing. Data processing is done locally on a desktop machine and sent to a cloud server for storage. Candidate basemaps are processed in data centres before they can be visualised in the application. On the other hand, VillageTree [**?**] offers intelligent pest management by gathering pest incidence reports from farmers. The application gathers crowdsourced reports on pest incidences from end-users. The collected data is analysed on a backend-end server using spatial-

temporal analytics and image recognition algorithms. Analysed information is then used to send contextualised notifications for end-users to take preventive measures. Also, the application uses the crowdsourcing approach to send images together with location information to other farmers that may be affected. This application offers different mobile interfaces designed for different stakeholders on different mobile platforms.

3.1.3 3-tier Applications

From the SAAs surveyed, MobiCrop [**?**,**?**] falls in this category. This application is designed to follow a mobile distributed architecture pattern with a three-layered deployment approach. In particular, this application is composed of mobile clients, a cloud-based middleware, and a cloud-based database server. The middleware runs in the middle tier as depicted in Figure 3 and is mainly used to shield the database server from the mobile clients. For this reason, the middleware is responsible for data routing, pre-fetching, and caching for offline accessibility. To achieve this, a policy-based system was implemented in the middleware as configuration file with a set of rules. These rules aid the system to control information being sent to the end-user; new updates are pushed to the end-user through the pre-fetching technique. The data being sent to the end-user follows a personalised content adaptation delivery approach. In this approach, content adaptation was done based on user preferences and device specific content adaptation. On the other hand, for offline accessibility, the application uses caching both at the middleware and the mobile client. Furthermore, in this application, data is modelled following the REST design standard. As such, it is manageable to process the end-user requests in the middleware.

3.1.4 Edge Computing Applications

From the literature surveyed, the SAAs that follow this model offload part of their computations to the edge to reduce delays that can be experienced when communicating to cloud-hosted services [**?**]. The SAAs surveyed that fall in this category include SmartHof [**?**] and SmartFarm [**?**] that were designed to monitor animal welfare.

The implementation of SmartHof follows a standard hierarchical structure with three components i.e., cloud computing, edge computing, and sensing (and actuation). The cloud provides data processing and storage, while at the edge, Raspberry Pi devices are used for data collection via sensors. A part of the client is worn by the animal to collect body temperature and animal movements using temperature and accelerometer sensors. Unlike, temperature data that can be used without processing, accelerometer data is processed first in the wearable component to derive meaningful information about the physical location of the animal and the number of steps moved. The processed data is then used by the application to trigger an alarm about the animal health when it is unable to move by correlating data in the cloud processing component. The edge component is implemented as a wearable and environment client to process data as it is being collected by sensors. The mobile application acts as an interface to manage farm configurations and evaluate animal welfare factors in the cloud. SmartFarm [**?**] follows a similar approach to SmartHof [**?**]. In particular, for this application, the environment and wearable clients are mostly used for data collection. Both applications enable the end-users to visualise and interact with the farm in real-time.

3.2 Classification Based on Software Engineering Issues

Table 2, which is a subset of Table 1, classifies SAAs based on how they deal with the software engineering issues identified in Section 2.2. Most of the applications in Table 1 are not 1-tier applications because they support features that are closely related to distributed SAAs. As mentioned before, such applications have different parts that talk to each other over communication networks. In Table 2, we write (✓) if the application supports the software engineering issue and (✗) if the application does not.

3.2.1 Offline Accessible Applications

Client/server applications can employ different techniques to ensure that they remain offline accessible and continue to function when the network becomes unavailable. Such techniques include buffering, synchronisation, client-side databases, and caching. Based on the literature surveyed, some SAAs employ caching for offline accessibility, while others use local databases e.g., SQLite and persistent databases at the client-side. These databases are synchronised to the server-side when the network becomes available.

In this regard, MobiCrop [**?**,**?**] uses a cloud-hosted middleware for data caching to offer offline accessibility. As noted before, this application exploits a dual caching technique where data is cached both on the mobile client and on the middleware. On the other hand, PAMS [**?**] uses a database at the client-side to store data upon network disconnection, while GeoFarmer [**?**] uses the local phone storage that is synchronised to a central database when the application is online. cFertigUAL [**?**] uses a persistent database on the client-side to minimise the number of requests that can be made to the server. This way the application can function offline in-between server request windows. PulAm [**?**] exploits an SQLite database that runs locally on the mobile device to minimise network connection issues. This application consist of four modules: (i) capturing crop inspec-

Table 2: Classifying smart agriculture applications based on software engineering issues. The applications in this table are a subset of Table 2 for distributed SAAs that support at least one of the software engineering issues.

Application	Agricultural theme	Agricultural domain	Offline accessible	Reactive	Reconfigurable
EVAPO [?]	Irrigation management	Evapotranspiration estimation	✗	✓	✗
PIS [?]	Irrigation management	Soil moisture sensing	✗	✓	✗
SmartIrrigation [?]	Irrigation management	Irrigation scheduling	✗	✓	✗
PAMS [?]	Geographic information systems	Managing farmland spatial data	✓	✗	✗
VillageTree [?]	Weed and pest control	Gathering pest incidence reports	✗	✓	✗
MobiCrop [?,?]	Weed and pest control	Sharing information on pesticides	✓	✗	✓
SmartHof [?]	Animal welfare	Monitoring animal health	✗	✓	✓
SmartFarm [?]	Animal welfare	Monitoring animal health	✗	✗	✓
Agri-IoT [?]	Animal welfare	Livestock fertility management	✗	✓	✓
SafeDriving [?]	Farm management	Equipment tracking	✗	✓	✗
UbiQON [?]	Greenhouse monitoring	Oyster mushroom monitoring	✗	✓	✗
Blynk [?]	Storage monitoring	Paddy rice monitoring	✗	✓	✗
SmartFarmKit [?]	Environment monitoring	Oyster mushroom and maize monitoring	✗	✓	✓
cFertigUAL [?]	Fertiliser management	Crop fertiliser estimation	✓	✗	✗
ConnectedFarm [?]	Environment monitoring	Remote monitoring and control	✗	✗	✓
PulAm [?]	Crop health	Crop pest monitoring	✓	✗	✗
AgDataBox [?]	Data collection	Field data recording	✗	✗	✓
IRIS [?]	Data collection	Field data recording	✗	✓	✗
iFarm [?]	Data collection	Field data recoding	✓	✗	✗
GeoFarmer [?]	Data collection	Field data recording	✓	✓	✗

tion data, (ii) database management, (iii) crop verification, and (iv) database synchronisation. The database management module is responsible for consulting, editing, deleting, and adding new crop inspection registers and pests. The changes that are made to the database when in offline mode are inserted into a queue. When internet connection becomes available, the contents of the queue are sent and synchronised to the server via *http* by the database synchronisation module. Lastly, iFarm [?] uses synchronisation to ensure offline accessibility. The client-side is synchronised to the server when the network is available to guarantee up-to-date data is available for local computations.

3.2.2 Reactive Applications

In order to offer reactivity, SAAs often use notifications and visualisations to give real-time feedback to end-users. The notifications are triggered upon external events that can happen in modern farming such as changes in weather conditions and market prices. The visualisations can be displayed on dashboards to help in tracking trends for different activities in modern farming. Notifications and visualisations can be generated by push-based or pull-based mechanisms, often via external services employed by the applications. For instance, Blynk [?] uses text messages to notify system administrators when sensing devices get disconnected from the system. Sensors that are used in this application automatically push the collected data to the application. Failure to receive such data triggers sending a text message to the end-user. In this example, push-based notifications and visualisations are automatically triggered once the desired events occur. On the other hand, pull-based notifications and visualisations rely on query fetching and processing which can have an implication on managing the network bandwidth [?]. The choice of either approach depends entirely on the application context. From the SAAs presented in Table 2, SmartHof [?] and SafeDriving [?] use push-based notifications, while SmartFarmKit [?] and IRIS [?] use pull-based approach to fetch data from sensors. Lastly, VillageTree [?] uses SMS texting to send contextualised alerts and relevant solutions to farmers for them to take preventive measures in pest control and management.

3.2.3 Reconfigurable Applications

Recall from Section 2.2 that building comprehensive SAAs that can cover all business domains may not be feasible since the workload conditions for SAAs can change over time and as such reconfigurability is important. As such, reconfigurable SAAs provide mechanisms that end-users can use to adapt them to different farming activities. In particular, reconfigurability in these applications can be achieved through APIs to add and extend existing application services, and widgets to change application user interfaces such data collection forms and alert messages. Additionally, APIs can be used to integrate SAAs to other systems in the farm. For instance, in AgDaBox [?], APIs are used to integrate to any other application that allows *http* communication with the server. Similarly, Agri-IoT [?] connects to other applications using APIs, while in SmartFarmKit [?], farm services are published as RESTful APIs with JSON data format. On the other hand, ConnectedFarm [?] uses APIs to allow third-party developers to implement their own farm services. Similarly, in other SAAs such as SmartHof [?], SmartFarm [?] and GeoFarmer [?], REST APIs are used to communicate to web servers in order to access backend application func-

tionalities. Lastly, as mentioned above, some SAAs such as SmartFarm [**?**], SmartHof [**?**], Agri-IoT [**?**] use widgets as a reconfiguration mechanism.

3.3 Discussion

In this article, we classified SAAs based on (i) architectural models and (ii) software engineering issues that we identified as important for software developers to deal with when building these applications. The combination between architectural models and mechanisms to deal with software engineering issues can determine which SAAs end-users will choose for their modern farming tasks. In what follows, we discuss the different trade-offs in order to guide those choices.

From the literature reviewed, common uses for applications that exploit the 1-tier model include journaling farming activities (data collection) and in-application data processing such as image processing that may not require network access. As such, SAAs that follow the 1-tier model adapt better to rural areas that have zero or poor quality network connections since, by design, they do not require network access to function. However, these SAAs have a single point of failure and as such they may not be scalable. Additionally, such applications may suffer from resource constraints to handle large datasets, support advanced data analytics and computational capabilities that are necessary in data transformation, processing, and storage. These limitations notwithstanding, as software developers, it should be important to make 1-tier applications rich in functionality to benefit the end-user. For instance, 1-tier SAAs that rely on historical data for prediction can be pre-loaded with such data to support machine learning and prediction.

From the state-of-the-art analysis, few client/server SAAs follow the 3-tier and edge computing model, while most SAAs follow the 2-tier model. These models can be used when computational resources required by SAAs to process and store high volume data are not sufficient locally, and as such rely on cloud-hosted computational resources. In addition, both 3-tier and edge computing models can be used when communication latency is an issue in the client/server setting. In this regards, the middleware in the 3-tier model can be used for caching responses to repetitive client requests to minimises queries that go directly to the server, while edge computing can be used to bring computational recourses closer to the data sources. Moreover, edge computing can be used to perform data analysis at the source before such data is sent to the cloud for further processing or storage.

In terms of supported software engineering issues, Table 2 does not show any SAAs that support all the three software engineering issues considered in this work. From the survey analysis, few client/server applications are both offline accessible and reconfigurable, offline and reactive, and lastly, reactive and reconfigurable. We think that these groupings are based on developer design choices based on contextual constraints such as communication and latency limits. In addition, most reactive applications use text and email messaging. Moreover, reconfigurable applications use APIs to allow addition of new services and extending existing application services. As such, the quality of these APIs should be enhanced to have minimal impact on application performance and user experience. Similarly, some applications use widgets to make them easier for end-users to reconfigure. Such widgets can facilitate modifying parts of an application that do not require hard-coding e.g., modifying data entry forms. Lastly, from this analysis, we observe that some SAAs address domain specific issues, while others address technological and software engineering issues. As technology advances, new techniques of implementing different software parts are designed. These techniques have to be implemented in the future versions of SAAs in an evolving manner. In terms of mobile platforms, most of the SAAs surveyed run on Android. We believe this is the case because it is cheaper to acquire an Android phone than an iPhone and those applications are sometimes used in developing countries where Android devices are increasingly being adopted.

From the perspective of smart agriculture processes described in Section 2, some SAAs can be dedicated to particular processes, while others can be multipurpose to cut across all the processes. In addition, the SAAs also cut across multiple research domains such big data applications in smart agriculture, Internet of Things, artificial intelligence applications in smart agriculture, and agricultural robotics. As such, we think that SAAs maybe widely spread than perceived from the literature.

Lastly, the distributed nature of client/server SAAs can allow end-users to monitor farming activities and receive timely notifications. Such notifications can be generated using push-based or pull-based techniques. The choice of which technique depends on the constraints at hand such as network bandwidth and the cost of processing queries sent or received from a remote server.

4 Open Issues and Research Directions

Based on the survey results, we note that SAAs are gaining traction both in research and industry. However, some issues and challenges are still open that can hinder adoption of SAAs especially in developing countries. In the following sections, we describe some of those open issues, related to some of the smart agriculture processes identified in Section 2.1, as well as to the different software engineering issues defined in Section 2.2.

4.1 Architectural Models

Data collection is an essential smart agriculture process, and is often implemented by using sensors, or by letting end-users (i.e., farmers) enter data directly into SAAs. The SAAs that present a more than 1 tier architecture, however, often rely on remote connections to one (or more) server(s) for processing and storage. If applications are not designed for providing offline accessibility, this can present a problem since broadband internet access in rural areas, where many farms are located, still remains a problem globally [**?**,**?**]. As a result, problematic network connections often lead to SAAs not functioning when not available. In Section 3.2.1, we provided an overview on how some offline accessibility techniques (e.g., database synchronisation) have been used in SAAs to deal with this problem. We believe, however, that not enough effort has been done in this direction, and that many farms, especially in developing countries, are pretty limited in the use of such applications.

For instance, to the best of our knowledge, no attention has been given to peer-to-peer (P2P) as an architectural model for SAAs. By providing a direct communication between different devices connected in one or more local networks, a P2P architecture could help in solving some data transmission problems brought about when communication networks are not available. In this regard, the P2P model can be used to transform mobile devices into data transmission nodes using wireless communication techniques such as Bluetooth and ZigBee. As such, we think that P2P applications can be exploited over ad-hoc wireless sensor networks to relay data to an edge server from where it can be uploaded to the cloud. This could lead to further research on specific P2P issues in the context of smart agriculture, such as the placement of edge devices, fault tolerance for edge devices, and deployment strategies according to farm segmentations and end-user requirements.

4.2 Scalability and Performance

Considering that part of the smart agriculture processes need often to be deployed on one or more low-resource devices, we believe that scalability and performance issues should also be addressed. None of the surveyed literature, however, focusses on analysing the application performance and scalability. For instance, it is interesting to evaluate and compare the effect of different offline availability techniques on memory as it is a scarce resource in mobile and embedded devices that can be used for data collection. Furthermore, from the surveyed SAAs, we noted that big amounts of data can be generated at global scales with diverse and multiple sensors. We think that this data can be used to validate big data processing techniques for smart agriculture. Lastly, in smart agriculture, SAAs rely on data fetched from sensors or servers through push-based and pull-based mechanisms. None of the surveyed literature compares these approaches in a large scale smart agriculture setting and their effects on application performance.

4.3 Security and Privacy

Overall, security and privacy issues are still a concern in all smart agriculture processes, since it has not been given much attention[**?**]. For instance, sending data to common cloud platforms, especially when using crowdsourcing techniques, may require end-users to share information that can be intercepted in cyber-security bridges. This can be related to information flow control, but, to the best of our knowledge, that has never been contextualised or adapted to SAAs.

5 Conclusion

In smart agriculture, SAAs are varied and spread over different agricultural domains such as irrigation management, fungicide management, risk management, fertiliser management, etc. Our study classifies SAAs based on (i) architectural models and (ii) on how they tackle specific software engineering issues such as offline accessibility, reactivity, and reconfigurability. We believe that these issues are important in the context of SAAs, since they represent problems often happening in smart farms; hence we present how different surveyed applications deal with them.

Based on the survey findings, most of the SAAs follow the 2-tier client/server model for data collection, data processing, data storage and information dissemination. From a software engineering perspective, these applications can serve different roles in a modern smart farm such as providing interfaces for farmers to interact with sensors, as functional blocks for Internet of Things implementations, and integrating with farm management information systems to help end-users in decision-making. However, none of the SAAs surveyed tackles all the three software engineering issues considered in this work.

Finally, we suggest how some issues such as scalability, performance, security, and privacy are all possible open avenues for future research, since they all represent important concerns of SAAs, that have not been investigated in detail.

Acknowledgements This work is supported by the Legumes Centre for Food and Nutrition Security (LCEFoNS) programme which is funded by VLIR-UOS. The programme is a North-South Collaboration between the Katholieke Universiteit Leuven, Vrije Universiteit Brussel (both in Belgium) and Jomo Kenyatta University of Agriculture and Technology (Kenya).

Conflict of interest: The authors declare that they have no conflict of interest.