Computers and Electronics in Agriculture 125 (2016) 12–28

Contents lists available at ScienceDirect

Computers and Electronics in Agriculture

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

Original papers

A reference architecture for Farm Software Ecosystems

J.W. Kruize [a,b,*], J. Wolfert [a,b], H. Scholten [a], C.N. Verdouw [a,b], A. Kassahun [a], A.J.M. Beulens [a]

[a] Information Technology, Wageningen University, Hollandseweg 1, 6706 KN Wageningen, The Netherlands
[b] Wageningen UR, Agricultural Economics Research Institute, Alexanderveld 5, 2585 DB Den Haag, The Netherlands

ARTICLE INFO

Article history:
Received 11 June 2015
Received in revised form 24 December 2015
Accepted 16 April 2016
Available online 4 May 2016

Keywords:
Farm Management Information Systems
Software Ecosystems
Open Software Enterprise
Interoperability
Precision agriculture
Smart farming

ABSTRACT

Smart farming is a management style that includes smart monitoring, planning and control of agricultural processes. This management style requires the use of a wide variety of software and hardware systems from multiple vendors. Adoption of smart farming is hampered because of a poor interoperability and data exchange between ICT components hindering integration. Software Ecosystems is a recent emerging concept in software engineering that addresses these integration challenges. Currently, several Software Ecosystems for farming are emerging. To guide and accelerate these developments, this paper provides a reference architecture for Farm Software Ecosystems. This reference architecture should be used to map, assess design and implement Farm Software Ecosystems. A key feature of this architecture is a particular configuration approach to connect ICT components developed by multiple vendors in a meaningful, feasible and coherent way. The reference architecture is evaluated by verification of the design with the requirements and by mapping two existing Farm Software Ecosystems using the Farm Software Ecosystem Reference Architecture. This mapping showed that the reference architecture provides insight into Farm Software Ecosystems as it can describe similarities and differences. A main conclusion is that the two existing Farm Software Ecosystems can improve configuration of different ICT components. Future research is needed to enhance configuration in Farm Software Ecosystems.

1. Introduction

Agri-food supply chain networks are confronted with a growing world population and increasing prosperity and associated changing demands. These developments are challenging because the demand on food is increasing while there are stricter requirements regarding food safety, sustainable food production and transparent supply chains. Therefore, farm enterprises[1] are pushed to improve their production processes by smart monitoring and control. Smart monitoring, -planning and -control of production processes, which can be referred to as smart farming, can be supported by a broad spectrum of technologies, ICT components, and their constituent hard- and software systems (Aubert et al., 2012; Cox, 2002; Lamb et al., 2008; Wolfert et al., 2010). Examples of these ICT components are all kinds of sensors, terminals, implement assemblies, computers and software applications. For smart monitoring and control an integrated information system is required that enables seamless interaction and sharing of data between different ICT components. However, a lack of interoperability is currently severely hindering smart farming because ICT components of multiple vendors do not operate as one integrated farm information system (Aubert et al., 2012; Fountas et al., 2005; Pedersen et al., 2004; Pierce and Nowak, 1999).

To overcome this, Wolfert et al. (2014) identified five main challenges (i) handling the increasingly large amounts of data, especially from all kind of agricultural equipment, (ii) interoperability between various systems at farm level and in the whole supply chain network surrounding the farm, (iii) standardization of data, (iv) go beyond the small scale and the regional focus of farm software development while at the same time (v) comply with national or regional differences in farming practices. More specifically for interoperability, the systematic analysis of Kruize et al. (2013) showed that ICT components used within the same farm enterprise (i) have partly overlapping and partly unique services, functions and interfaces, (ii) are missing required application services, functions and interfaces, (iii) have separated data repositories and (iv) have inadequate and incomplete data exchange. In conclusion, most of the available ICT components are lacking both technical and semantic interoperability, resulting in data sharing

* Corresponding author at: Information Technology, Wageningen University, Hollandseweg 1, PO Box 8130, 6700 EW Wageningen, The Netherlands.
E-mail address: janwillem.kruize@wur.nl (J.W. Kruize).

[1] A farm enterprise can be an arable farm, livestock farm or horticultural farm. In this paper we focus on arable farm enterprises however it is expected that the concept Farm Software Ecosystem can address software integration challenges for the other type of farms as well.

http://dx.doi.org/10.1016/j.compag.2016.04.011
0168-1699/

issues and non-coherent user interfaces (Kruize et al., 2013). Consequently, current ICT components often hamper farm enterprise integration as they do not sufficiently support the monitoring, planning and control processes to enable smart farming. Supporting these processes by making a combination of multiple ICT Components is currently challenging. In addition, the creation of one overarching system developed by one software vendor that overcomes all mentioned challenges is neither a feasible nor – from a competitive point of view – a desirable solution. Hence, a promising method to achieve such integrated solutions is a best of breed approach, which allows users to configure customized software systems from standardized components that are supplied by multiple vendors (Light et al., 2001; Verdouw et al., 2010). As a consequence, software systems are not supplied by single companies, but by a set of independent actors which collaborate and can compete via an integration platform (Light et al., 2001). This integration approach requires an advanced infrastructure that covers both organizational and technological aspects (Wolfert et al., 2010). An organizational infrastructure is required that enables and facilitates both collaboration and competition between actors. In such infrastructure, actors collaborate in their development to provide interoperable ICT components that are based on their core competences and compete with ICT components that provide similar functionalities. A technological infrastructure is required that can support the linkage of ICT components into integrated FMISs. Both the organizational and technological infrastructure should enable and ensure a sustainable collaboration and competition in which all actors, including software developers, farm enterprises, contractors, technology providers and others, can flourish.

A concept that addresses such an infrastructure is nowadays called a Software Ecosystem. Currently, Software Ecosystems are becoming more widespread as they are increasingly considered to provide an effective way to construct large software systems on top of a software platform by combining components, developed by actors that are part of different organizations (Bosch, 2009; Manikas and Hansen, 2013; te Molder et al., 2011). Examples of current Software Ecosystems are, among others, Eclipse, Linux/Linux kernel and Android (Manikas and Hansen, 2013). At the moment there are no well-established Software Ecosystems for farming available, although several developments go into this direction. Large agricultural machinery vendors have setup their own proprietary platforms (e.g. John Deere's Farmsight[2] or AGCO's Fuse Technology[3]). With these platforms it is still difficult to establish interoperability with other components that come from other manufacturers. Several multi-vendor platforms (e.g. 365FarmNet[4], Crop-R, AgroSense, FIspace) are recently introduced, but these are still in an early stage of development and sometimes regionally oriented lacking a large international user base.

To gain deeper insights into these developments and to support further development of Farm Software Ecosystems, this paper proposes a reference architecture that can be used to map, assess, design and implement Farm Software Ecosystems that contribute to integrated FMISs. The purpose of the reference architecture is to improve communication and collaboration between multiple actors that are part of real-world Farm Software Ecosystems. It will help them to understand Software Ecosystems and enable them to join, form or improve Farm Software Ecosystems that lead to integrated farm information systems.

The remainder of this paper first introduces literature about Software Ecosystems and the relation to software development for farming. Second, the methodology for designing the reference architecture for Farm Software Ecosystems is described. Next, the requirements for the reference architecture, the reference architecture itself and an example farm information system that can result from a Farm Software Ecosystem is described. This is followed by an evaluation to verify the Reference Architecture based on the requirements and to validate if it can map existing Farm Software Ecosystems to provide insight how it matches and in what extend. This paper concludes with a discussion and outlook for future research and development.

2. Software Ecosystems and software development for farming

In the Internet of Services (IoS) software components are available as interoperable services on the internet. The IoS allows to decouple the possession and ownership of software from its usage and thus to use Software as a Service (Turner et al., 2003). Users do not need to buy and install a large software system, but required functionality is delivered as a set of distributed web services that can be configured and executed when needed. In contrast to traditional non-modular software systems, it is no longer necessary that components are delivered by the same software vendor. Software companies can concentrate on the development of components that fit best to their core competences. Users can configure customized software systems from standardized components that are supplied by multiple vendors that interact via a common technological platform. Such collaborative environments are nowadays referred to as Software Ecosystems. Software Ecosystems are defined as the interaction of a set of actors on top of a common technological platform that results in a coherent set of ICT components or services (Manikas and Hansen, 2013). These components include hardware, software and service modules, along with an architecture that specifies how they fit together (Eisenmann et al., 2008).

In practice, a Software Ecosystem is usually started by a single- or a group of software producing organizations that open up their business processes to become an Open Software Enterprise (Jansen et al., 2012). Such an Open Software Enterprise provides a technical platform and additional (collaboration) artefacts that are essential for the coherence of the software components and for collaboration between multiple actors (Seichter et al., 2010). There are various reasons why actors with different perspectives would like to collaborate in such an environment (Bosch, 2009; Wolfert et al., 2010):

- It increases the value of the core offering to existing users and increases the attractiveness for new users.
- Increase "stickiness" of the technology platform, i.e. it is harder to change the platform when it is widely used (cf. PC operating systems e.g. Windows, iOS, etc.).
- It creates and facilitates a structural and independent environment, developed by partners in the ecosystem that potentially offers a large critical mass of users (once success has been proven).
- Share the costs of innovation by collaborating with other actors and accelerate innovation through open innovation in the ecosystem.
- Decrease total costs of ownership and risks for commoditizing functionality by sharing the maintenance with networking partners.

The concept of Software Ecosystems is new for the agricultural domain. Related literature focuses on the integrating capabilities of farm ICT components by proposing a standardized infrastructure that supports the integration of ICT components of multiple vendors (Iftikhar and Pedersen, 2011; Kaloxylos et al., 2012; Nash et al., 2009; Steinberger et al., 2009; Wolfert et al., 2010).

[2] www.myjohndeere.deere.com.
[3] www.agcotechnologies.com.
[4] www.365farmnet.com.

Most of these papers focus on semantic aspects of the data to improve interoperability of application components (Iftikhar and Pedersen, 2011; Nash et al., 2009; Steinberger et al., 2009). Examples of available standards for the agricultural domain that facilitates data exchange between application components are the international ISO-11783 standard[5], the Dutch EDI-Teelt standard[6], and the German AgroXML standard[7]. Most papers also focus on application integration in which there is a focus on the design of an integrated FMIS (Kaloxylos et al., 2012; Kaloxylos et al., 2014; Nikkilä et al., 2010; Sørensen et al., 2010a; Sørensen et al., 2010b). A recent implementation of a platform that can be used to develop an integrated Farm Management Information System (FMIS) is described in Kaloxylos et al. (2012) and Kaloxylos et al. (2014). Yet, this literature misses a specification on how to operationalize and organize a Farm Software Ecosystem. The reference architecture in this paper will address this shortcoming.

Furthermore, the Software Ecosystem literature is in an early stage since the concept is coined relatively recently (Messerschmitt and Szyperski, 2005). There is still little consensus on what precisely constitutes a Software Ecosystem, a few analytical models of Software Ecosystems exist, and little research is done in the context of real-world Software Ecosystems (Manikas and Hansen, 2013). Hence, this paper also aims to contribute to the theoretical basis of Software Ecosystems in general.

3. Material and methods

3.1. The Future Internet Program and existing Farm Software Ecosystems

The research presented in this paper was carried out as a part of SmartAgriFood and FIspace project which are part of the European Future Internet Public–Private Partnership programme (FI-PPP)[8]. In SmartAgriFood, the needs from the agri-food sector for Future Internet ICTs were identified while at the same time the capabilities of Future Internet were described by potential use case scenarios in agri-food (Eigenmann et al., 2012; Sebök et al., 2012). This has resulted in a conceptual platform architecture and several prototype applications. The FIspace project is currently implementing these concepts into a software platform for business collaboration for the agri-food, transport and logistics domain. This platform and its architecture will enable collaboration of application components and can be a basis to form several Software Ecosystems. More detailed background information about these projects can be found in Kruize et al. (2014), Verdouw et al. (2014) and Wolfert et al. (2014).

As a conceptual validation to test the mapping functionality of the reference model two Dutch initiatives in which Farm Software Ecosystems are being established were selected: Crop-R and AgroSense. Crop-R is a Dutch organization developing an online platform offering GIS-based crop-recording applications on the web, smartphones and tablets. The system has currently more than one thousand users such as farm enterprises and contractors. AgroSense is an open source platform on which a modular and open source FMIS can be configured. The modules can come from different independent organizations.

3.2. Methodology

The reference architecture for Farm Software Ecosystems was developed by a design-oriented research approach (March and Smith, 1995). Aligned with the guidelines and framework of Hevner et al. (2004), the reference architecture was designed in four steps: (i) ontology definition, (ii) requirements analysis, (iii) development of a basic design and (iv) evaluation of this design within case studies (see Fig. 1).

Ontologies are important tools to enable seamless communication and mutual understanding about a domain and can reduce misunderstanding between people and enable interoperability between software components, e.g. apps, data, etc. (Scholten et al., 2007). In this research an initial basis for an ontology has been laid down and is used to describe the requirements analysis, basic design and the application of the design. The ontology has been iteratively developed within this research of which the basis was the architectural language ArchiMate (TheOpenGroup, 2011). The ontology is available in the Appendix and all words in capitals are referring to this ontology.

The requirements analysis started with the identification and definition of the scope the object system for Farm Software Ecosystems based on Sørensen et al. (2010b) and Wolfert et al. (2010). Next, the requirements for Farm Software Ecosystems were derived from the research that was carried out in the SmartAgriFood and FIspace projects. In SmartAgriFood a survey was carried out to assess the user's expectations on future internet Functions and Services. In total 135 questionnaires in 6 countries and 8 focus group discussions with 69 participants in 5 countries have been performed and analyzed, collecting feedback on the interpretation of the future internet capabilities from the user's perspective, the current use of internet applications in their daily work and today's problems or limitations resulting in expectations and requirements for the future internet. These were compared to the Future Internet's capabilities that were identified in the FIWARE project[9]. FIWARE provides enhanced OpenStack-based cloud hosting capabilities and a rich library of components – so-called Generic Enablers (GEs) – implementing a number of added-value functions offered 'as-a-service'. The Generic Enablers concern among others Context Management, easy connection to the Internet of Things, Open Data support and Big Data processing and analysis. A technical team consisting of a number of software architects from leading ICT companies and technical universities - including the authors – developed the conceptual architecture into the FIspace platform. A detailed documentation of this platform can be found online[10]. Beside the empirical results from these projects, a literature analysis was done on the development of Software Ecosystems in relation to smart farming. Based on the ontology and the requirements analysis, the reference architecture for Farm Software Ecosystems was designed.

The evaluation of the Reference Architecture contains two parts. First the Reference Architecture was verified based on the requirements. Second, as a conceptual validation the mapping functionality of the Reference Architecture was tested using two existing Farm Software Ecosystems. These mappings are based on semi-structured interview with the CTO's of AgroSense and Crop-R.

4. Requirements analysis

4.1. Object system

To define the scope for Farm Software Ecosystems, the object system of this study should be defined. For that purpose Farm Enterprises are considered as the Business Systems with basic inputs and outputs for which agricultural software is developed that virtualize the objects (Fig. 2). Virtualization allows to decouple

[5] http://dictionary.isobus.net/isobus/.
[6] www.agroconnect.nl.
[7] www.agroxml.de.
[8] See www.fi-ppp.eu.
[9] See www.fiware.org.
[10] https://bitbucket.org/fi-space/doc/wiki/Home.

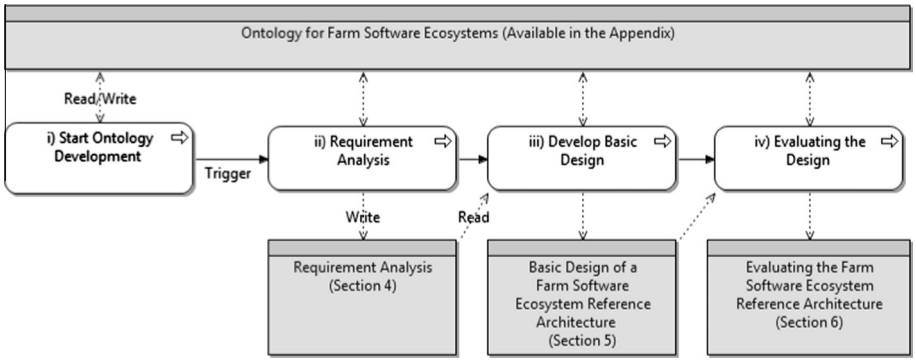

Fig. 1. Research approach.

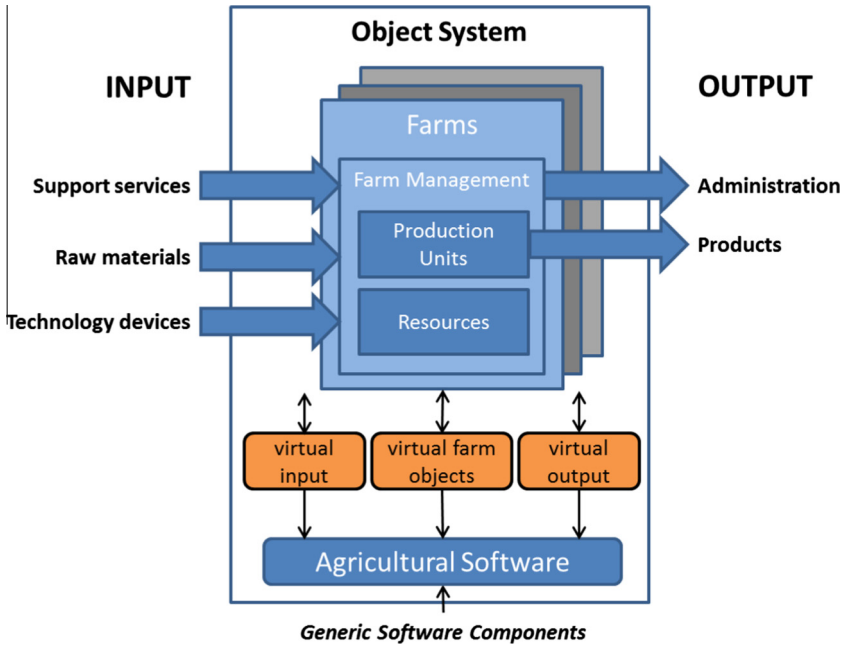

Fig. 2. The object system that is identified for Farm Software Ecosystems. Input, output and generic software components are considered to come from outside the system.

physical flows from information aspects of operations (Clarke, 1998; Verdouw et al., 2013). The core of the object system is formed by a number of Farm Enterprises that consist of production units (e.g. fields and its crops), resources (e.g. humans, Devices, buildings, etc.) and its management. To manage the production units all kind of resources are used for the production that take place in these fields ranging from big tractors, combine-harvesters to small sensors. Farm management aims to control crop production using resources that should operate as an integrated system. Raw materials (seeds, fertilizers, pesticides, etc.) are primary inputs for the production processes. Support services can be all kind of advice that helps the farmer in managing the whole Farm Enterprise (e.g. weather information, crop status). At the output side products come from the fields that can be temporarily stored in buildings. Farm management results into administration about the products produced and the usage of resources needed in supply chains (e.g. certificates), but also more general administration at farm level that is needed for public administration and other purposes.

It can be observed that virtualization plays an important role to represent input, output production units and resources. Providers of inputs such as raw materials, technology devices and support services are trying to add value to their products by selling software and information in combination with their products. Processors of products and retailers try to get more control on the outputs from Farm Enterprises (e.g. certification information) by introducing their software at the farm. Similarly, public administrations are introducing software for farmers to streamline the information they need (e.g. for subsidies). At this moment the software for farm management is usually produced by companies to whom the farming domain is their core business. The software that is involved at the input- and output side is usually provided by third parties ranging from big established software companies to small, innovative start-ups. All this software is specifically focussing on the Farm Enterprise and thus included in the farm object system. Software producing companies are generally using generic software components that according to Fig. 2 are considered to come from outside the farm object system (e.g. FIWARE Generic Enablers, Operating Systems etc.). Companies that are involved at the input/output side (e.g. chemical companies, processors) also have software to support their own production processes. These software applications can have Application Interfaces to enable data exchange with software applications used at the farm.

4.2. Functional requirements for a Farm Software Ecosystem

A Software Ecosystem was previously defined as the interaction between Actors on top of a common technological Platform result-

ing in a coherent set of ICT Products or Services. The previous section described the object system that a Farm Software Ecosystems has to focus on. The main question is how to operationalize and organize a Farm Software Ecosystem around a common technological Platform in order to meet and overcome the challenges intrinsic to the farm object system and mentioned in the introduction. To answer this question functional requirements are derived within the SmartAgriFood project, the FIspace project and an additional literature analysis. In SmartAgriFood the users' needs and expectations were derived and mapped onto the FIWARE GEs. This resulted in a number of use case scenarios in particular for smart farming that are previously published by Kaloxylos et al. (2012). Based on these results, literature both on Software Ecosystems and smart- and precision farming and the development of the FIspace platform the following main functional requirement categories for Farm Software Ecosystem can be identified, which are:

- smooth data handling and seamless data exchange between ICT Components (Kaloxylos et al., 2012; Kruize et al., 2013);
- a configuration approach to link ICT Components to each other in a meaningful and coherent way (Kaloxylos et al., 2012; Verdouw et al., 2014);
- interoperability of different ICT Components (Kaloxylos et al., 2012; Kruize et al., 2013);
- an Open Software Enterprise that smoothly facilitates the previous points (Jansen et al., 2012).

The next subsections will describe these functional requirement categories in more detail.

4.2.1. Data handling and seamless data exchange supported by Standards

In general, but also in agriculture, the amount of available data is exploding due to the introduction of all kind of sensing and monitoring Devices. One aspect of this increasing amount of data in agriculture is its storage and the transport capacity of the network. Especially this transportation of data in agriculture is challenging due to lack of connectivity and bandwidth in the remote areas and the mobile ICT Components in farm environments. The storage of farm data is mostly done in data repositories located at multiple sites (e.g. at the farm or in different cloud repositories). Locating all these data to one central place to add intelligence is difficult because this data is often acquired by different ICT Components (e.g. sensors, monitoring Devices or Application Components). Still, farmers need integrated solutions in which access to data located in distributed repositories with multiple Application Components is required. This requires a Farm Software Ecosystem to support the development of Application Components that are able to exchange messages that contain data between distributed data repositories. For seamless data exchange data lifecycle considerations have to be taken into account (e.g. data collection, processing, sharing).

To exchange data between different applications the Application Programming Interfaces (API's) must enable sending and receiving messages. These messages must be based on technologies and semantics that are known by the other API's. Technology standards including the syntax are currently well standardized (e.g. web-service technology using XML or Json as a syntax). Furthermore, semantic standards are available although in agriculture semantic standards or its implementation are often missing, making data exchange between ICT Components cumbersome (Kruize et al., 2013). The result is that farmers are not able to exchange data between different ICT Components (e.g. data exchange between FMIS and sensors) and are affected by a vendor lock-in hindering farmers changing ICT Components (e.g. to change their FMIS) (Kruize et al., 2013). In some cases, farmers would need advanced data handling skills to make data exchange between components work, which is a not desirable situation. Hence, Application Components developed within a Farm Software Ecosystem should be able to share data in an automated manner (Kruize et al., 2013; Sørensen et al., 2010b). The sharing of data between different Application Components should be organized in a robust manner because new Application Components emerge over time, needing data from existing ones. Therefore transfer of data between Application Components should be facilitated based on a shared Implementation of a communication protocol. Such a communication protocol is based on both technical and data semantic agreements. To enable the re-use of data from a semantic perspective Application Interfaces require data that is tagged with metadata or require certain data attributes. These metadata or data attributes enable that data, saved in a data repository (e.g. database) can be reused by multiple Application Components to support multiple Business Processes. This metadata should additionally describe for what kind of purposes the data can be used to resolve the fit for use aspect of data. From a technical perspective the integration of Application Components and their distributed data repositories requires a aligned technical architecture (e.g. a Service Oriented Architecture) (Wolfert et al., 2010).

Currently, there are multiple standardisation organizations that organize data exchange between Application Components by providing technical and semantic standards. Examples are ISO Standards, ISA Standards[11], OGC Standards[12] and farm specific Standards such as EDI-Teelt or AgroXML. The use of these standards should be stimulated by Farm Software Ecosystems to enable data exchange between Application Components of multiple vendors. When existing standards are not proficient ad-hoc standards should be implemented to enable data exchange between the Application Components. The Farm Software Ecosystem should provide documentation about these ad-hoc standards. The documentation of these ad-hoc standards can be provided to software developers within the Farm Software Ecosystem to enable data exchange. Furthermore, the documentation can be used to adapt existing standards.

The exchange of data between different Application Components is related to the next main requirement for Farm Software Ecosystems: a configuration approach to link ICT Components to each other that fit to the needs of the supported business process.

4.2.2. Flexible configuration of ICT Components

Every Farm Enterprise is unique in its specific Business Processes and in the connections with other Actors at the input and output side (see Fig. 2). Moreover, farm business can be very dynamic because of changing situations (e.g. fluctuating markets, weather changes, resources, etc.). These factors are influenced by regional differences and different farming practices. At the same time, at higher abstraction levels there are many similarities between Farm Enterprises and their Business Processes. The challenge for Farm Software Ecosystems is to deliver customized ICT Components based on both generic and specific Application Components. For Farm Enterprises the configuration of an aligned and integrated system, consisting of components of multiple vendors, is required (Aubert et al., 2012; Fountas et al., 2005; Pedersen et al., 2004; Pierce and Nowak, 1999). Farm Software Ecosystems should therefore provide Artefacts that support software developers to develop interoperable ICT Components that can be configured by farmers – or their service providers – into an aligned Composite Application Component that dynamically supports cer-

[11] www.isa.org.
[12] http://www.opengeospatial.org/standards.

tain Business Processes and can be customized for their specific circumstances. Such a configured ICT Component should have a coherent look and feel regarding the User Interface to enable farmers to participate in dynamic business networks and support a variety of farm Business Processes and management goals.

A known configuration approach in software development that fits to this requirement is called ICT mass customization. ICT mass customization combines efficient standard software and flexible customized software allowing the configuration of standardized ICT Components into a customer-specific assembly (Verdouw et al., 2010). To enable such a mass customization approach the Farm Software Ecosystem must fulfil multiple functional requirements (Verdouw et al., 2014):

- *Software Modularity* – ICT products consist of loosely coupled modules for which policy, input–output data, and Application Interfaces are well defined and that can be easily substituted by other modules;
- *Information Integration Platform* – deployed environment that enacts the execution of modules and enables/manages the exchange of information between them;
- *Component Availability* – all required components should be readily available to configure the right Products that are required by the customer;
- *Configuration Support* – adequate tools that guide users interactively through the Product specification process at different abstraction levels accounting for the fact that different configuration steps can take place at different moments in time by different people;
- *Reference Information Models* – standardized taxonomies that represent all possible configuration options of Product Instances and the interdependencies that exist between Components or features, including rules for permitted combinations.

In the current situation some ICT Components can exchange data. However, services that support farmers in configuring ICT Components into an integrated Farm Information System are hardly available (Kruize et al., 2013). In a Farm Software Ecosystem a common platform should enable integration of ICT Components into an integrated system by dynamic orchestration of Application Services, i.e. organizing, maintaining and managing. This requires substantial knowledge about the customer that will use it. An overview of required Application Services -grouped as Functions – are for example automated advisory, task plan analyser, crop availability, etc. (Kaloxylos et al., 2012). To operationalize ICT mass customization, ICT Components need to be interoperable such that the components collaborate as they are one aligned and integrated system.

4.2.3. Interoperability between ICT Components

Interoperability of ICT components is understood as components that have a shared Implementation of a communication protocol (e.g. a transfer protocol, an Application Interface) to communicate properly[13]. Interoperability enables that components are able to share data and can collaborate as if they were components of one aligned and integrated system. Such an integrated system should be configured in which the actual farm Business Processes are the foundation of the configuration process and the selected ICT Components (Wolfert et al., 2010).

To create an aligned and integrated system that can cover and support Business Processes of multiple Farm Enterprises a large variety of Application Services is required. However (i) currently, not all required Application Services are available and can be realized by the ICT Components and (ii) ICT Components currently used in a single arable Farm Enterprise have partly overlapping and partly unique Application Services and Application Interfaces (Kruize et al., 2013). Therefore, Farm Software Ecosystems should enable that multiple vendors can develop interoperable Application Components offering Application Services. These Application Components can offer similar Application Services to stimulate competition within a Farm Software Ecosystem or different Application Services to offer enough functionality. The best fitting Application Components can be selected in a configuration process. To enable configuration each component should have a detailed supplementary description in what kind of configuration the Application Component can be used, what kind of input data is required and what output data it provides. Additionally, the description of the Application Component should describe the performance (e.g. idle time) to ensure that the configuration works as a coherent system. Such descriptions should show which Application Components are interoperable and for what kind of configurations they can be used. The Farm Software Ecosystem should provide guidance to enable interoperability between ICT Components and provide a format for the description of each Application Component.

An overview of general requirements regarding seamless interoperability in collaborative-competitive economic networked environments can be found in Chituc et al. (2009). These requirements focus on establishing collaborations with external Actors regarding data exchange (e.g. weather data, input data).

Besides these aspects the Farm Software Ecosystem should provide an organizational structure that support the development of interoperable ICT Components.

4.2.4. Organization

To develop interoperable ICT Components that can be used by Farm Enterprises different Actors, with aligned incentives, performing different roles are required to collaborate. To facilitate collaboration Farm Software Ecosystems should provide an organization (Open Software Enterprise) (Jansen et al., 2012). This Open Software Enterprise should enable the development of interoperable Application Components, the configuration process and the operation of the configured Application Components in runtime.

For this run-time environment a technological (cloud) infrastructure should be available to host the platform and which is able to connect to all Application Components. Furthermore, it should provide a revenue and cost sharing model as software developers, infrastructure providers and configuration service providers are using each other's components and services. To facilitate this, basic support for e.g. Contracts, payments, etc. is required. Furthermore, when possible disputes arise the governance structure should provide a resolving mechanism.

For the development of interoperable Application Components in a distributed environment a basic level of governance is required and the collaborating Actors should agree on the use of both technical standards as semantic standards that are publicly available or specific for a Software Ecosystem.

Overall, it is important that the Open Software Enterprise of a Farm Software Ecosystem enables that costs are reduced so that End-Users (e.g. farmers, contractors) can buy software functionalities that make their enterprises more advanced. Especially as farms are relatively small enterprises and are not able to invest large amounts of money in software. Therefore it will be required that all kind of Actors, having aligned incentives, are able to become part of a Farm Software Ecosystem to stimulate competition within the platform. Hence the Open Software Enterprise of a Farm Software Ecosystem should be open and avoid domination by large players that could cause vendor lock-ins, which ultimately will hamper innovation (Gawer and Cusumano, 2002). According

[13] Based on https://wiki.oasis-open.org/tab/InteropGuide.

to Eisenmann et al. (2008) a Software Ecosystem is 'open' when (i) no restrictions are placed on participation in its development, commercialization or use and (ii) any restrictions are applied uniformly to all potential platform participants (e.g. requirements to conform to technical standards or pay licensing fees are reasonable and non-discriminatory). Still, other forms of governance of Open Software Enterprises, for example with a more dominant player, can as well result in successful Software Ecosystems.

4.2.5. Technical/non-functional requirements

Beside the functional requirements that are mentioned so far, there are several functional requirements to be addressed, such as user management, data security, routing of information or machine-to-machine communication. Although these requirements are very important in the end, they are considered to go beyond the scope of this paper because they are not very specific for Farm Software Ecosystems.

4.3. Specifications for Farm Software Ecosystems

The requirements from the previous section are summarized in Table 2 which can be found in the Appendix. Based on these requirements specifications for Farm Software Ecosystems are derived that can improve farm enterprise integration. These specifications supported designing the reference architecture for Farm Software Ecosystems and are::

- The ecosystem should provide functional **ICT Components** for farming that
  - can be based on Application Components developed by various Actors independently;
  - allow for distributed data exchange in an automated, seamless manner;
  - use existing standards or be able to exchange data with ICT Components using other standards to enable data exchange.
  - support a configuration approach of the aforementioned interoperable Application Components that can be deployed in a distributed manner
- The ecosystem should enable Actors to perform different roles to enable that different **Business Services** for farming are offered that support:
  - software configuration
  - software development
  - software hosting
- The main categories of **Actor roles** are providers of ICT Components (Software Vendors), Agricultural Service Providers, Providers of the Infrastructure and users of these Services and the ICT Components. These Actor should be able to:
  - join the ecosystem;
  - influence the ecosystem and enable innovation;
  - form a critical mass of users and providers to make the ecosystem efficient and effective;
- A common and open **Platform** is needed:
  - to facilitate collaboration between various Actors using the ICT Components and its Application Services;
  - that provides consistent standards for this collaboration that in the future will not create restrictions for exploitation (backwards- and forwards compatibility) so that multiple ecosystems around the same platform are possible;
  - that supports:
    - development of ICT Components according to the platform standards;
    - development of cohesive User-Interfaces to improve the user experience;
    - configuration of ICT Components into integrated systems in an easy but consistent manner, using reference information models.
- An **Open Software Enterprise** should:
  - provide the actual (cloud) infrastructure to make the ecosystem possible;
  - orchestrate the whole collaboration process and resolve possible disputes;
  - be neutral toward the other Actors in the ecosystem or at least transparent in case they also participate as provider of services and/or ICT Components;
  - not be dominated by a single organization;
  - manage the platform in such a way that it is affordable for SMEs to participate;
  - ensure that Application Components are developed according to the Platform Architecture (e.g. a service-oriented architecture) to facilitate configuration and collaboration;
  - ensure that Application Components contain an description to enable the use of it in various configurations;
  - ensure that the data shared between Application Components are tagged.

5. Basic design of a Farm Software Ecosystem reference architecture

A reference architecture for a Farm Software Ecosystem describes the generic structure (concepts and relations) of specific Farm Software Ecosystems. In this section a Farm Software Ecosystem reference architecture is described according to the requirements and specifications that were defined in Section 4. At the end an illustrative example is provided to show how the different components fit together in practice. A table containing all the components and sub-components that are part of the Farm Software Ecosystem reference architecture can be found in Section 6.2 Table 1. This table is used to map the existing Farm Software Ecosystems.

5.1. High-level description of the reference architecture

Fig. 3 provides a high-level view of the Farm Software Ecosystem reference architecture design. The architecture comprises five main components: (i) Actors, (ii) Platform, (iii) Open Software Enterprise, (iv) Business Services and (v) ICT Components. Actors provide or use ICT Components and Business Services. The Platform includes ICT Components for End-Users. The relation between the Actors and the Platform is managed by the Open Software Enterprise (organization). The following subsections will describe various components of the design in more detail.

5.2. Platform

A platform is a set of stable components that supports diversity and evolution in a system by constraining the linkages among the other components (Baldwin and Woodard, 2008). These components are integrated and work as an integrated system. These components include software and service modules, along with an architecture that specifies how they fit together (Eisenmann et al., 2008). A Platform used within Farm Software Ecosystems must be able to support four Actor roles which are; end-users (e.g. farmers, contractors), software vendors/developer (e.g. app developer), agricultural services providers (e.g. a configurator of different systems) and the Platform orchestrator that among others runs the Platform. With such a Platform a configurator should be

Table 1
Mapping of the Farm Software Ecosystem reference architecture components and sub-components on the Farm Software Ecosystems Crop-R and AgroSense.

Components of the Farm Software Ecosystem reference architecture	Sub-Components	Part of AgroSense	Part of Crop-R
Open Software Enterprise	Open Software Enterprise structure	Yes	Yes
	A partnership model/Actor model	No	Yes
	IP Strategy Documentation	Yes	No
	Technology Vision	Yes	Yes
	Technology research vision	Yes	Yes
	Technical Architecture Documentation	Yes	Yes
	Farm Information Model	-	-
	Actor Model	Yes	No
	Business Control Models	No	Yes
	Business Process Model	No	No
	Data Model	Yes	Yes
	Application Programming Interface	Yes	Yes
	Collaborative tools	Yes	Yes
	Configuration Support documentation	No	No
Actors	Orchestrator Role	Yes	Yes
	Niche Player Role	Yes	Yes
	External Actor Role	Yes	Yes
	Vendor/Value Added Reseller Role	Yes	Yes
	End-User/Customer Role	Yes	Yes
	Software Vendor Role	Yes	Yes
	Agriculture Service Provider Role	Yes	Yes
	Infrastructure Provider Role	Yes	Yes
	Customer/End-User	Yes	Yes
Business Services	Business Services Offered	Yes	Yes
Platform	Operating System	Yes	Yes
	Orchestration Module	Partly	No
	System and Data integration module	Yes	Yes
	Security Privacy Trust Framework	Yes	Yes
	Development Kit	Yes	No
ICT Component	Atomic Application Components	Yes	Yes
	Composite Application Components	Yes	Yes

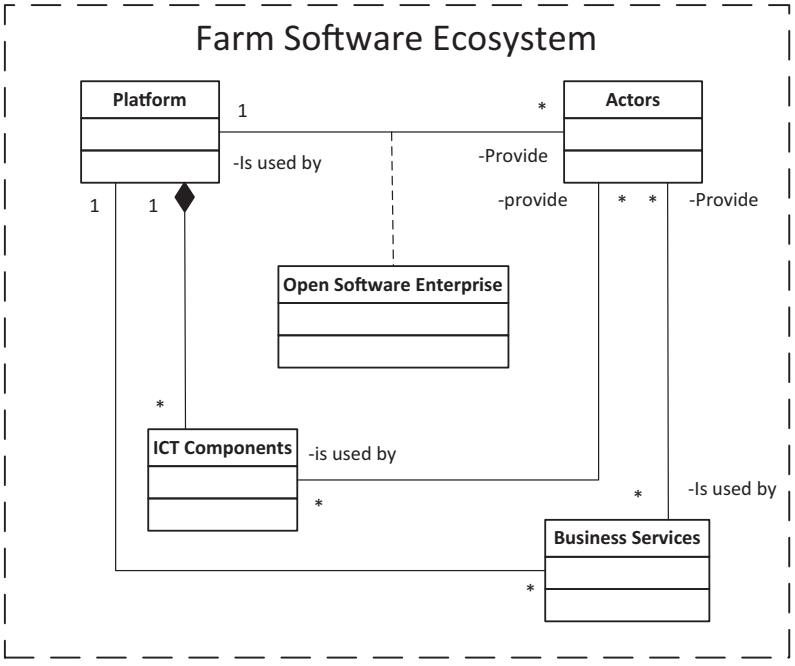

Fig. 3. High-level view of the Farm Software Ecosystem reference architecture.

enabled to configure an ICT Component for a farmer using Application Components of multiple vendors. More details regarding these roles can be found in Section 5.3.

According to the requirements and specifications that were defined in Section 4 platform modules are defined. Each of these modules has independently value for the actor roles using the Platform. These modules can be found in Fig. 4 that provides the basic architecture. In this platform architecture five modules are defined; Operating System, Development Kit, Orchestration, Security, Privacy & Trust framework and System & Data Integration.

The system & data integration module must provide API's to enable smooth data exchange between Application Components and enables access to distributed data repositories. To enable smooth data exchange it should contain mechanisms for data

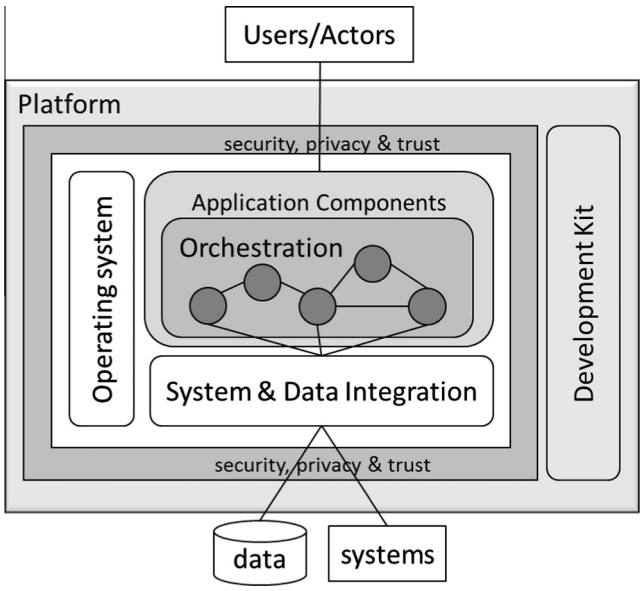

Fig. 4. Basic architecture of a platform for a Farm Software Ecosystem. Adapted from the FIspace platfor.

mediation to be able to handle heterogeneous data from various sources. Additionally, Payment of Application Services should be handled within this module.

The security privacy & trust framework manages all the connections. These connections can be with external data and systems or with human users. To manage these connections secure authentication and authorization methods that meet required levels of security assurance are used.

An orchestration module enables configuration. In this module linkages between different Application Services of Atomic Application Components (or Platforms) can be defined[14]. After configuring Application Services of Atomic Application Components the module intermediates between status and events of each individual Application Component. The technical specifications of this configuration are not provided in this paper and will be presented in future research. In this paper there is a focus on the organizational aspect of configuration which will be described in more detail in Section 5.4.

A software development kit is available to enable software developers to develop Application Component that can become part of the Farm Software Ecosystem. It supports the development of Application Components that can be connected to the orchestration module and should stimulate that the Application Components have a coherent look and feel regarding the User Interface.

The operating system module is needed to ensure the technical interoperability and communication between the different Platform components so that it operates as a consistent whole. One aspect is the execution of the configured ICT Components.

Regarding the deployment of the Platform and the configured ICT Components we do not provide a detailed description. Some Platforms are instantiated on a cloud Node, others on a server that is part of an enterprise and others are instantiated on a client such as a personal computer at a farm. The deployment of such a Platform and the associated ICT Components can differ for each Farm Software Ecosystem. A detailed description of a specific Platform for farming can be found in Kaloxylos et al. (2012) and Kaloxylos et al. (2014). A description of how the platform in the FIspace project was developed can be found in Verdouw et al. (2014).

5.3. Actors and their relationships

In a Farm Software Ecosystem multiple Actors collaborate having different organizational and operational roles. The five organizational roles are (Manikas and Hansen, 2013):

- *Orchestrator* – manages the software ecosystem and is responsible for its overall functioning and performance; runs the Platform, creating and applying rules, processes, business procedures, setting and monitoring quality standards and/or orchestrating the Actor relationships determining the openness.
- *Niche Player* – develops or adds ICT Components to the technical Platform, producing functionality that customers or End-Users require.
- *External Actor* – makes use of the possibilities of the Farm Software Ecosystem and providing indirect value to the ecosystem (e.g. by testing the platform and Application Components or by providing business services to End-Users).
- *Vendor/Value Added Reseller* – makes profit from selling the ICT Components to End-Users or other vendors/value added resellers.
- *End-User/Customer* – purchases or obtains a complete or partial Product which is a service and/or a configuration of ICT Components (e.g. farmers, agronomists).

From an operational point of view Actors part of Farm Software Ecosystems can be classified into four main roles (Handoyo et al., 2013):

- *Software Vendor (Software Developer)* – is developing the technical platform, the Application Components and the Devices/Nodes.
- *Agricultural Service Provider* – provides organizational Business Services including selling, customization, deployment and maintenance of ICT Components and operational services including End-User oriented functions and data. Examples of Business Services are requirements engineering, configuration and orchestration support of ICT Components.
- *Infrastructure Provider* – provides a technical infrastructure (e.g. servers, networks).
- *Customer/End-User* – uses ICT Components or Services that are offered by the Software Ecosystem (e.g. farmers, agronomists).

In practice, organizational and operational roles can be played by one and the same company.

5.4. ICT Components, configuration and orchestration

The end-user roles (e.g. farmers, contractors, agronomists) are supported in their Business Processes by ICT Components. These ICT Components will usually be a configuration of several subcomponents that should collaborate seamlessly and support dynamic business collaboration processes. Therefore it is necessary to define these components in a more detailed way. The relationship between several components is presented in Fig. 5.

An ICT Component is an Application Component that is deployed on a Node (e.g. local computer, internet, etc.) which supports one or more Business Processes of a company. An example of an ICT Components is the hard- and software of a terminal or PC that supports a spraying process. An Application Component is a modular, deployable, and replaceable piece of software system that

[14] Within a Farm Software Ecosystem one or more Platforms can be present. In the case of configuration, one Platform will be in charge of the orchestration by connecting Application Services offered by Atomic Application Components and/or Platforms. In the case of competition between Farm Software Ecosystems and its Platforms, data from one Platform should be exported to another. In such case a data broker could be useful to exchange (and maybe store) data used by both platforms.

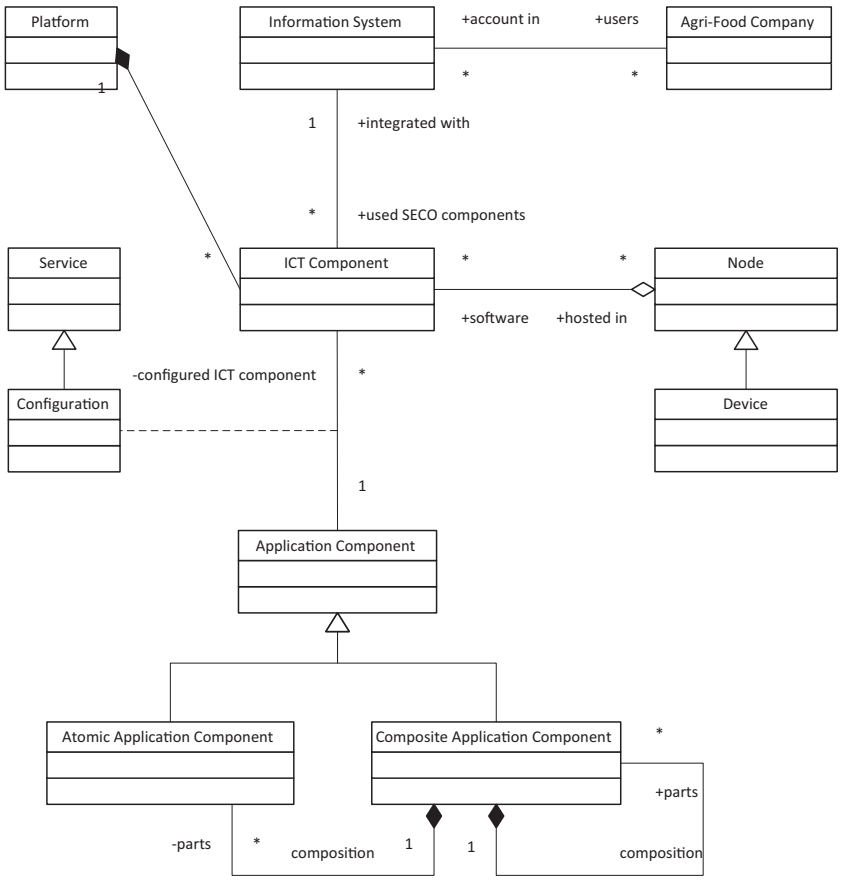

Fig. 5. A UML Class diagram describing the relationships between various components in the Farm Software Ecosystem. Further explanation can be found in the text.

encapsulates its behavior and data and exposes these through a set of Application Interfaces (Wiederhold, 1992), for example a software module for spraying that needs to be deployed on a Node. ICT Components can be either based on Atomic Application Components or Composite Application Components. An Atomic Application Component is a piece of software, not able to share data automatically with other Application Components, e.g. a loose App on a smartphone that provides weather information. A Composite Application Component is a configuration of Atomic Application Components that performs collective behavior and has a coherent user-interface, in which data is automatically exchanged between (Atomic) Application Components. The orchestration module of the Platform makes them work together seamlessly as if it was one system. Therefore, the Composite Application Components require that each individual component keeps to be maintained and stays operational. When one component fails the whole composite component might fail. This requires the Farm Software Ecosystem organization to provide functionalities that covers these issues.

In a configuration process Application Services of different Application Components are selected and configured that can support Business Processes of a specific farm. The ICT Components can be offered by actors within the Farm Software Ecosystem roles software vendor or agricultural service provider and are based on various (Atomic or Composite) Application Components. A configured ICT Component can be instantiated multiple times and, in this way, support similar Business Processes. For example, an actor having the agricultural service provider role configures a Composite Application Component that can support crop protection processes. This Composite Application Component can then be instantiated on different Devices (e.g. a computer or the hardware of a Terminal) so that the resulting ICT Component can support similar Business Processes at different Farm Enterprises. Additionally, an instantiation of the Application Component part of an ICT Component can take place several times at the same Farm Enterprise. In this case, a Farm Enterprise can be supported in the crop protection Business Process, basically returning every season, with the same Application Component that is instantiated and customized to the actual situation (e.g. field, crop, etc.) of each season.

Maintenance support – a role that can be played by an agricultural service provider - is needed because (Atomic) Application Components will usually be offered by different vendors or Agricultural Service Providers. If one (Atomic) Application Component stops working, a whole (Composite) Application Component and ultimately the ICT Component and Service might malfunction as well. Therefore, a monitoring service is required that checks if each component is working correctly and if not repair or replace it with another one. These situations should also be carefully described in agreements between the users and providers of components. That will be discussed in the next section.

5.5. Business Services and Contracts

Farmers require a variety of Business Services for support. An example of an Business Service that should be offered by an agricultural service provider in a Farm Software Ecosystem is the support of a configuration process (Kruize et al., 2013). Configuration processes are knowledge intensive because the functionality of multiple Application Components and possible configurations need to be known. Therefore, a farmer might not be able to configure a Composite Application Component himself, requiring an agricultural service provider to help. In Fig. 6 the relationship between

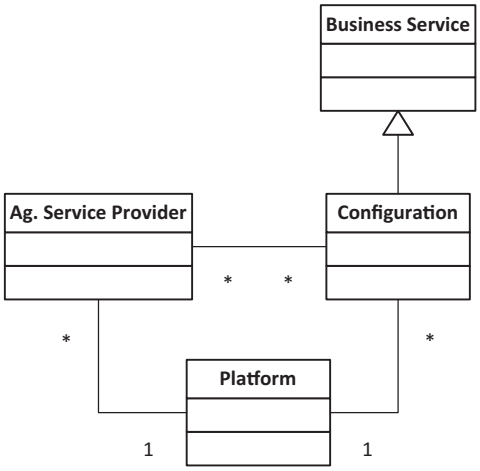

Fig. 6. Relationship between the role Agricultural Service Provider, and its Business Service (Configuration). Further explanation is in the text.

the agricultural service provider role, the Business Service 'Configuration' and the use of the Platform to enable a configuration process is described.

To enable collaboration between Actors that perform different roles contractual arrangements are required to support collaboration using a Contract. Contracts can be formal when two roles collaborate that are part of different legal entities or less formal when two roles collaborate that are part of the same legal entity. The relationships between Actors in a certain role in which a Contract forms the formal connection are described in Fig. 7.

Fig. 7 describes three possible contractual agreements; in reality there can be more. The first one is between a software vendor that provides a licence to an agricultural service provider to use an Application Component in a configuration Service. The Contract describes e.g. how much is paid on what basis, use and ownership of the data that is involved, etc. The second one is about collaboration between an agricultural service provider and an agri-food company. The agricultural service provider configures an Application Component which is bought by an agri-food company. This Contract contains agreements about the costs, service level, data use, etc. In the third example an agri-food company collaborates with an infrastructure provider. The agri-food company pays an infrastructure provider to host an ICT Component and eventually to provide data storage. Beside collaboration of Actors with different operational roles Actors can collaborate with Actors having the same operational role. For example, an agricultural service provider, offering a data services, can collaborate with another agricultural service provider that aggregates data into new data.

5.6. Open Software Enterprise

In a Farm Software Ecosystem Actors, which are part of various legal entities at different geographical locations, need to collaborate and develop software across organizational boundaries. The Open Software Enterprise that has to facilitate and orchestrate the ecosystem should fulfil this role. Different Farm Software Ecosystems can implement such an Open Software Enterprise in different ways. For example, an Open Software Enterprise can be orchestrated by a single company or a joint venture of companies that facilitates the ecosystem and possibly also the platform infrastructure. Because of these variation in implementation of Open Software Enterprises we cannot provide an exact blueprint of how they should be organized, but at least they should enhance innovation and collaboration by covering the following aspects, based on Jansen et al. (2012): (i) Governance, (ii) Research and Development, (iii) Software Product Management, (iv) Marketing and Sales and (v) Consulting and Support Services. The following subsections will describe this in more detail.

5.6.1. Governance of a Farm Software Ecosystem

An Open Software Enterprise governs a Farm Software Ecosystem that involves the assignment of roles and decisions rights, as well as the measures and policies that enable continuation of the ecosystem. The processes that are part of the governance process group will be performed by Actors having an orchestrator role. The ability of other Actors to participate in the ecosystem and

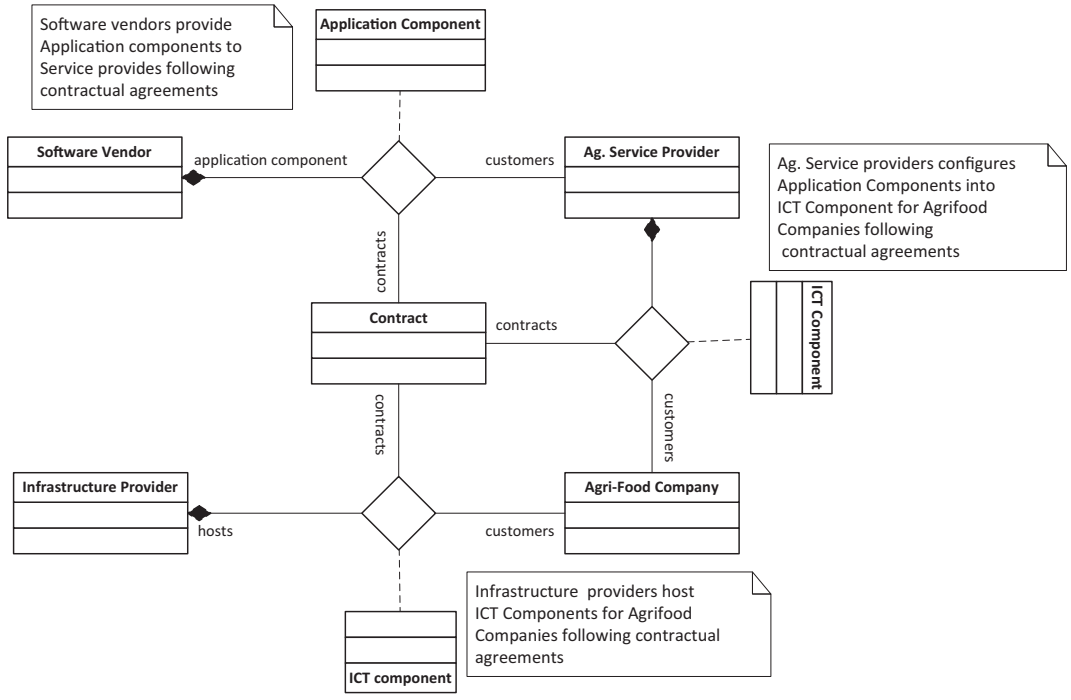

Fig. 7. Relationships between several Actors of a Farm Software Ecosystem in which a Contract plays a central role.

the decision making aspects of the governance determines the openness of each individual Farm Software Ecosystem. Independent of the openness of each ecosystem, an orchestrator must allow multiple Actors to develop and build ICT Components and Services using the Farm Software Ecosystem Platform. Two important aspects should be taken into account:

- *Partnership Model* – describes the organizational model of the Farm Software Ecosystem and makes governance policies explicit.
- *IP Strategy Documentation* – describes how to use and reuse source code, data libraries, etc. which is an important basis for the contracts between Actors.

5.6.2. Research and development

Farm Software Ecosystems will develop continuously and should follow the state of the art in technology developments. A research and development strategy is therefore important in which the direction of the Platform, the Application Components and the ICT Components is determined and should address:

- *Technology Vision* – identifies the upcoming challenges that are relevant to the domain on which the software business focusses;
- *Technology Research Vision* – defines the research priorities that the Actors of the Farm Software Ecosystem will focus on for the next two to three years;
- *Documentation of the architecture* – describes the platform architecture as the fundamental organization of a system embodied in its components, their relationships to each other and to the environment, and the principles guiding its design and evolution (IEEE 1471:2000);
- *Farm information model* – describes the organization of Farm Enterprises providing a systematic representation from different viewpoints and at various abstraction levels (Verdouw et al., 2010). The most important information model levels are actor- business control-, business process- and data models that describe the semantics (Wolfert et al., 2010);
- *Application programming interfaces documentation* – defines and describes how a particular Application Component could and should be used by other components, including semantic specifications;
- *Development of Collaborative Tools* – required for collaborative software development (e.g. communication, project management, issue tracking, bug tracking, globally accessible backlog, burn down chart tools etc.) (Hossain et al., 2009).

5.6.3. Software Product Management and configuration support documentation

Software Product Management is a process of managing Products (ICT Components), taking lifecycle considerations into account. Software Product Management focusses on the details of the Products such as the requirements, quality, development and marketing (Jansen et al., 2012) and will be mostly performed by Actors that have an agricultural service provider role. Actors collaborating in software product management processes should ensure that a variety of modular Application Components is available that can be configured into farm specific ICT Components, which contain Composite Application Components.

Because configuration of different Application Components is a key asset of Farm Software Ecosystems, it should be carefully documented how to do this. The available Application Components and their additional descriptions should be documented. This documentation can be supported by the farm information model that describes – or possibly *pre*scribes – certain processes or workflows in farming and the type of Application Components that could – or should – be used. Examples and tutorials will help software developers to adopt these principles quicker.

5.6.4. Marketing and sales

Processes of the marketing and sales process group focus on the marketing of the Products. Actors with a software vendor role or agricultural service provider role that sell Composite Application Components will perform most of these Business Processes. Marketing and sales of a Farm Software Ecosystem is essential to attract a large amount of end-users. A large amount of end-users makes the ecosystem more viable and attracts vendors to offer new Application Components.

5.6.5. Consulting and Support Services

Consulting and support service focuses on supporting end-users with implementing and using their ICT Components. These processes will be mostly performed by Actors having an agricultural service provider- or infrastructure provider role. They should provide implementation services, in which they support other Actors in configuring a Service or ICT Component e.g. by providing documentation, reference information models, tutorials, example configurations, etc.

5.7. Example of a Farm Software Ecosystem

In the example Farm Software Ecosystem 'Alfa' different fictive actors collaborate. This collaboration results into an illustrational implementation of which the layout is presented in Fig. 8. In this example a farm enterprise is supported in crop protection by a FMIS that is configured, using a platform, from different components that are provided by different Actors.

The farm Information System contains three Atomic Application Components, each offering specific Application Services. Atomic Application Component 1 offers a sensor-based field monitoring service that provides the climate conditions of a specific field. Atomic Application Component 3 can receive these sensor data and is able to analyse this data to provide insight into the risk of some diseases. Atomic Application Component 3 requires basic crop data (e.g. crop, planting time, soil type of field) that is provided by Atomic Application Component 2. The disease analysis is presented to the farmer using a coherent user interface in a Composite Application Component that is configured from Atomic Application components 2 and 3. The components are deployed through the Internet by a cloud Node into ICT Component I that can be used by the farmer as an FMIS to support a crop protection process. As indicated in Fig. 8, ICT Component I is offered to the farmer by an agricultural service provider. This agricultural service provider has configured Atomic Application Components from Software Vendor A and B into a Composite Application Component. It should be noted that only ICT Component I is part of a platform that is hosted by an infrastructure provider, but through an Application Interface it can use the sensor data from ICT Component II. As indicated, the Sensor (ICT Component II) can also be used standalone by the farmer if necessary or in similar ways to support other processes (e.g. fertilization) that requires the same sensor data.

If one of the three Atomic Application Components is not working according to the requirements of the farmer it can be replaced by a similar one that is offered within Farm Software Ecosystem Alfa. This process is supported by an independent agricultural service provider. In such case a reconfiguration is needed. This can

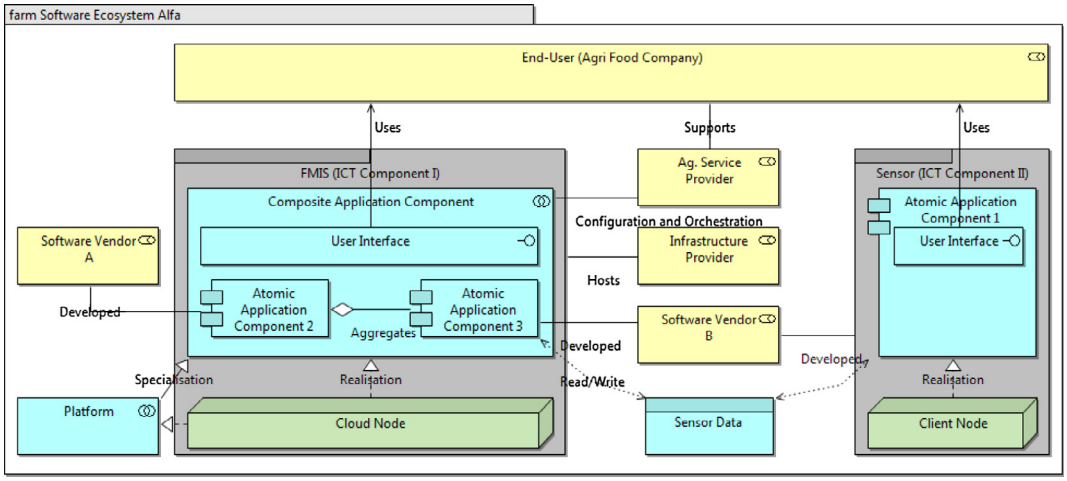

Fig. 8. Layout of the farm software ecosystem 'Alfa' showing how different components and Actors roles are related to each other. Further explanation is in the text (The model can be downloaded using the following link: http://tinyurl.com/gwbpdco).

prevent a vendor lock-in as an Application Component from a vendor can be replaced by an Application Component from another vendor. Moreover, the farmer can not only change the software vendors that offer Atomic Application Components, he can change the infrastructure provider or agricultural service providers as well. This enables the Farm Enterprise to configure an Information System that is flexible and can fulfil specific requirements over time.

The Farm Software Ecosystem concept can prevent vendor lock-in as multiple vendors can compete within the same Ecosystem and specific Application Components can be exchanged. Still, the farmers could be locked into a specific Farm Software Ecosystem. To reduce the risk on a Farm Software Ecosystem lock-in the use of data standards within Farm Software Ecosystems is recommended. This would enable transferring data from one Farm Software Ecosystem to another.

6. Evaluation

This section describes the evaluation of the Farm Software Ecosystems reference architecture. First, the reference architecture is verified by checking the requirements with the design. Second, a conceptual validation is presented in which the reference architecture is used for mapping the existing Farm Software Ecosystems AgroSense and Crop-R. The evaluation misses a verification regarding the usefulness of the reference architecture to asses, design and implement Farm Software Ecosystems.

6.1. Requirements verification

In Section 4.2 requirements regarding Farm Software Ecosystems are presented. Software Ecosystems fulfilling these requirements should resolve current integration challenges in farming and enable Farm Enterprise integration. To contribute to Farm Enterprise Integration this research has designed a Reference Architecture for Farm Software Ecosystems. This architecture can be used to map, asses, design and implement real worlds Farm Software Ecosystems that fulfil the requirements presented in Section 4.2. As a verification these requirements and our design are compared, see Table 2 that can be found in the Appendix. For each requirement we describe what part of the reference architecture addresses the requirement. Based on this verification we conclude that all requirements are addressed by the reference architecture for Farm Software Ecosystems.

6.2. Validation of the reference architecture by mapping existing Farm Software Ecosystems

As a conceptual validation, the Farm Software Ecosystem reference architecture was used to map the existing Farm Software Ecosystems AgroSense and Crop-R. Based on such a mapping similarities and differences between real-world Farm Software Ecosystems should become visible. Additionally it provides a comparison between the reference architecture and real-world Farm Software Ecosystems. Too create a mapping the components (Open Software Enterprise, Platform, ICT Components, Actors and Business Services) and sub-components provided in the reference architecture are used to describe the real world Farm Software Ecosystem. The results of the mapping can be found in Table 1. A more detailed description of both Farm Software Ecosystems can be found online[15].

The mapping of the real-world farm Software Ecosystems on the reference architecture shows that many components and sub-components of the reference architecture are part of real-world Farm Software Ecosystems. Both Farm Software Ecosystems together provide a more encompassing mapping. This shows that the reference architecture can be used to map the real-world Farm Software Ecosystems.

The interviews with the CTO's showed that this mapping provides insight into real-world Farm Software Ecosystems as similarities and differences can be found that depend on the scope and objectives. Crop-R focusses more on combining data from multiple sources to provide information to farmers. Their focus is less on configuration and system integration and the orchestration of services is coded in the platform. AgroSense has more focus on integration aspects and provides a software development kit and modules can be added by other software developer to the Platform. These modules provide specific functionalities and include the coded orchestration of services. Both Platforms do not yet provide a flexible configuration approach. Consequently, both existing ecosystems AgroSense and Crop-R do not provide documentation for configuration support. This indicates that configuration, and how this should be done, is not yet well developed although it is identified as a major requirement for Farm Software Ecosystems (see Section 4.2). An extensive configuration approach is currently missing in both Farm Software Ecosystems. However, it should be noted that these ecosystems are still developing and that new parts are becoming available in the near future.

[15] The description of each Farm Software Ecosystem can be found on: http://tinyurl.com/gwbpdco.

7. Discussion and conclusions

This paper proposed a reference architecture for Farm Software Ecosystems that contributes to software development to enable smart farming. Seamless data exchange and dynamic interoperability of different Application Components to enable configuration were identified as the most important challenges. A suitable configuration approach is necessary to link Application Components to each other in a meaningful and coherent way. The resulting Composite Application Component needs to be deployed at Farm Enterprises and be supported by a agricultural service provider. An Open Software Enterprise governs and facilitates the ecosystem and the collaboration between Actors. The reference architecture can be used to map current real-world Farm Software Ecosystems and it is expected to be able to design and implement future Farm Software Ecosystems.

It was demonstrated how Farm Software Ecosystems can align software development with multiple Actors in a distributed environment. In such an ecosystem small players can focus on development of small Application Components while relying on larger ICT players that provide general infrastructural components (data storage, servers, etc.) or more complex analyses (e.g. super computers, big data) by generic software components (see Fig. 2). In this way innovation can be stimulated giving small start-up companies a fair chance to accelerate their innovative product and gain market share. This approach is currently stimulated in the FIWARE accelerator program[16], which is partly connected with the FIspace Platform, and in which agriculture is an important focus area. For end-users in the ecosystem it is expected that the alignment of software development will lead to better and more affordable solutions because innovation costs are shared. Additionally, the solution can become more flexible because end-users can change the different sub-components of an integrated solution, which will stimulate competition at that level.

For integration of different sub-components it is important that the configuration process is supported in an appropriate and well-defined manner. It was concluded that both evaluated Farm Software Ecosystems, AgroSense and Crop-R, are lacking such a support. The FIspace Ecosystem and platform provides a business collaboration core for that purpose (Kruize et al., 2014; Verdouw et al., 2014), although this platform and its ecosystem is still being established. Reference information models - especially Business Process models – could play an important role in a configuration process by describing the farm Business Processes at various levels and from different viewpoints (Verdouw et al., 2010). Such reference information models are also still poorly defined in the current FIspace architecture. They should also refer to common standards as much as possible to ensure interoperability at higher integration levels and acceptation by many users at a global level. The reference models and standards could be offered by an (agricultural) service provider to one or more Farm Software Ecosystems. Future research should focus on developing these information models for Farm Software Ecosystems to provide knowledge about software configuration.

The reference architecture presented in this paper is supposed to be a common basis for various Instances of Farm Software Ecosystems that could compete with each other. However, the Open Software Enterprises should prevent ecosystem lock-ins, *i.e.* that End-Users are hindered in substituting components from one ecosystem to the other. To that end, a kind of federated structure between the different Farm Software Ecosystems is needed. This can be reached by using common standards mainly focussing on platform and semantic interoperability.

Although the general market principles and business models for Farm Software Ecosystems will not be fundamentally different in comparison to the past, there are definitely new modes of collaboration emerging that make it necessary for companies to reconsider their strategies (Porter and Heppelmann, 2014). Developers of application and ICT Components should realize that their solution will be part of larger integrated solutions at a higher level. As a consequence, they should not focus at the final end-user so much but establish collaboration with agricultural service providers supporting configuration and other Application Component developers. It also becomes more important to comply with common standards as much as possible to increase the potential usage of a component at a global level. Agricultural Service providers could focus on specific end-users groups (e.g. farmers, input suppliers) combining personal advice and communication with the best customized configuration of ICT- and Application Components based on flexible Contracts. These are just a few examples of developments in new business models that can be expected from Farm Software Ecosystem development. Examples from other sectors show that successful business models emerge gradually as the ecosystem develops (Van 't Spijker, 2014). An essential issue in these developments is about data ownership, security, privacy and trust. Although the importance was emphasized in this paper, further research is needed on how to deal with this topic in the context of open dynamic Farm Software Ecosystems.

From this discussion it can be concluded that there are still technical challenges to be met, especially in the configuration of different components into integrated solutions. It is also clear that there are still many organizational developments needed to successfully develop Farm Software Ecosystems. The reference architecture in this paper can help to guide these developments, which can enrich the architecture.

Acknowledgements

This project was partly funded by the Dutch Program on Precision Agriculture (PPL) and the FIspace project, which is a FI-PPP phase 2 project, granted by the EC in FP7 under grant agreement n° 604123. Furthermore we want to thank all the persons who contributed to the case studies, especially Ivor Bosloper, Nicole Bartelds and Timon Veenstra. Thanks are due to Dr. John McBreen for valuable editorial contributions.

Appendix A. Requirements table

See Table 2.

Appendix B. Ontology

In this research a first version of an ontology has been developed to describe our research object: Farm Software Ecosystems. This ontology can be found in Table 3, and is based on literature (See http://tinyurl.com/gwbpdco). It is developed iteratively during this research and open for improvements in future. The ontology provides descriptions of concepts and Artefacts and should describe the research object Farm Software Ecosystems from a farm perspective and from a software development perspective. In these descriptions italic words denote to other concepts or Artefacts defined within the ontology. This ontology is used to describe the requirements, basic design and case studies. In the basic design relations between some of these concepts and Artefacts are presented.

[16] www.fiware.org/fiware-accelerator-programme.

Table 2
Requirements and design verifications.

Category	#	Requirement	Design verification
Data handling and seamless data exchange	1.1	Enable message (data) exchange between distributed systems	Message exchange is address in the architectures part: • Open Software Enterprise, specifically R&D o Stimulates to use standards and the development of an Information model o Defining API's • Platform o Stimulates linking Application Components (configuration) o Proposing System and Data integration module • ICT Components o Describe configuration
	1.2	Provide technical and semantic standards to facilitate data exchange	Providing technical and semantic standards is done in the architecture part: • Open Software Enterprise, R&D; o Stimulating the use of existing standards o Defining API's
	1.3	Enable the re-use of data	The re-use of data is addressed in the architecture part: • Open Software Enterprise, specifically R&D and Governance o Stimulates to use standards and the development of an Information model o Defining API's o Provide IP Strategy Documentation • Platform o Stimulates linking Application Components (configuration) o Proposing System and Data integration module • ICT Components o Describe configuration
	1.4	Stimulate the use of existing standards	Stimulating the use of standards is addressed in the architecture parts: • Open Software Enterprise, specifically R&D: o Stimulates to use standards and the development of an Information model
Configuration of ICT Components	2.1	Deliver customized ICT Component configurations	Customization of ICT Components is addressed in all the architecture parts: • Open Software Enterprise • ICT Components • Platform • Actor descriptions • Business Services
	2.2	Provide artefacts supporting a multi-vendor software development	This is addressed in the part about the Open Software Enterprise; the artefacts to stimulate collaboration are listed
	2.3	Develop integrated systems with a coherent User Interface regarding the look and feel	The development of integrated systems in addressed in the parts about: • the Open Software Enterprise • Platform • ICT Components
	2.4	Enable software modularity	Software modularity is addressed in the parts about: • the Open Software Enterprise • Platform • ICT Components
	2.5	Provide an information integration platform	This is addressed in the part about the Platform
	2.6	Component availability	This is addressed in the parts: • Open Software Enterprise, Marketing and sales where is described how this can stimulated actors to join a Software Ecosystem
	2.7	Configuration support	This is addressed in the part • Business Services • Actors • Open Software Enterprise, Consulting and Support Services
	2.8	Reference Information model	This is addressed in the part: • Open Software Enterprise, R&D
Interoperability of ICT Components	3.1	Provide meta data about Application Components (e.g. performance etc.)	This is addressed in the part: • ICT Components • Open Software Enterprise, Product Management
Open Software Enterprise's	4.1	Provide an Open Software Enterprise	This is addressed in the part: • Open Software Enterprise
	4.2	Provide hosting for the platform	This is addressed in the part: • Actors, by describing the different roles actors can take in a Farm Software Ecosystem
	4.3	Enable collaboration between multiple vendors	This is addressed in all parts of the reference architecture
	4.4	Enable competition between actors within Farm Software Ecosystems	This is addressed in the part: • Open Software Enterprise, o Governance, proving a Partnership Model
	4.5	Stimulate innovation	This is addressed in the part: • Open Software Enterprise, o Governance, by proving a Partnership Model and IP strategy Information o Research and development • Software Product Management

Table 3
First version of an ontology for Farm Software Ecosystems. The references of these definitions can be found online (http://tinyurl.com/gwbpdco).

Concepts	Definition
Active Structure element	An entity that is capable of performing behavior
Actor	An organizational entity that is capable of performing behavior
Application Interface	An *Interface* where an *Application Service,* as part of an *Application Component,* is made available to an *End-User* or another *Application Component*
Application *Service*	A *Service* that exposes automated behavior
Artefact	An object made by a human being to support a software engineering process
behavioral element	A unit of activity performed by one or more *active structure elements*
Business Process	A behavior*al Element* that groups behavior based on an ordering of activities. It is intended to produce a defined set of *Products* or *Business Services*
Business System	*Information System*, *Business Processes*, Policy Statements, Activities, *Standards*, and People which together implement a *Function*
Component	*Component* is a part or element of a larger whole
Contract	A formal or informal specification of agreement between actors that are part of different legal entities that specifies the rights and obligations associated with a *Product*
End-User	Actors who ultimately uses *ICT Products* or *Services* to support their *Business Processes*
Farm Enterprise	A *Farm Enterprise* is a *Business Actor* which has agricultural production as its main activity
Farm *Information System*	An *Information System* that is part of a *Business System* that is part of a *Farm Enterprise*
Farm Management *Information System*	A *Farm Management Information System* (FMIS) is an *ICT Component,* consisting out of one or more *Application Components*, for collecting, processing, storing and disseminating of data in the form of information needed to carry out the operational functions of the farm
Farm Software Ecosystem	A *Software Ecosystem* that has *Farm Enterprises* and its collaborating *Actors* as *End-Users*
Function	An intentional activity of a *System*
Implementation	The process to create an *Instance*
Information *System*	A computer based system to support organizational processes of an Actor
Instance	A realization of an *Artefact* in its environment
Product	A coherent collection of *Services* or *Artefacts*, accompanied by a *Contract* and usage documentation which is offered as a whole to customers
Service	A unit of functionality that a system exposes to its environment, while hiding internal operations, which provides a certain value (monetary or otherwise)
Software Ecosystem	The interaction of a set of Actors on top of a common technological Platform that result in a coherent set of *ICT Products* or *Services*
Software Producing Organization	A *Business Actor* developing *ICT Components*
System	A collection of *Components* organized to accomplish a specific Function or set of functions
Artefacts	Definition
Application *Component*	A *modular* (Wiederhold, 1992), deployable, and replaceable part of an *Information System* that encapsulates its behavior and data and exposes these through a set of *Interfaces*. Its *instance* that is deployed on a *Device* is named an *ICT Component*
Device	A hardware resource upon which *Application Components* may be stored or deployed for execution
ICT *Component*	An *ICT Component* is an *Application Component (Composite Application Component* or *Atomic Application Component)* that is deployed on a *Node* and which supports one or more *Business Processes* of a Business Actor
Atomic Application *Component*	An Atomic *Application Component,* deployed on a *Device*, is part of an *Information System* however not able to share data automatically with other *Application Components*
Composite Application *Component*	A composition of *Atomic Application Components*, which is created by ICT Customization using a configuration process (Verdouw et al., 2010), that performs collective behavior by exchanging data automatically between the Atomic Application Components and which can be deployed on a *Device*
ICT *Product*	A Product including a *Contract*, *ICT Component,* Configuration support (Verdouw et al., 2010) and usage support that is offered by an *Actor*
Node	A computational resource upon which *Application Components* may be stored or deployed for execution
Platform	A set of stable *Components* that supports variety and evolution in a system by constraining the linkages among the other components. These components include hardware, software and service modules, along with an architecture that specifies how they fit together
Software Ecosystem Instance	A real-world and operational *Software Ecosystem*
Standard	A *Standard* is something used as a measure, norm, or *Model* in comparative evaluations. There are four standards classifications in ICT Development: Business Standards, Data Standards, Application Standards and Technology Standards

References

Aubert, B.A., Schroeder, A., Grimaudo, J., 2012. IT as enabler of sustainable farming: an empirical analysis of farmers' adoption decision of precision agriculture technology. Decis. Supp. Syst. 54, 510–520.

Baldwin, C.Y., Woodard, C.J., 2008. The architecture of platforms: a unified view. Harvard Business School Finance Working Paper.

Bosch, J., 2009. From software product lines to software ecosystems. In: Proceedings of the 13th International Software Product Line Conference. Carnegie Mellon University, pp. 111–119.

Chituc, C.-M., Azevedo, A., Toscano, C., 2009. A framework proposal for seamless interoperability in a collaborative networked environment. Comput. Ind. 60, 317–338.

Clarke, M.P., 1998. Virtual logistics: an introduction and overview of the concepts. Int. J. Phys. Distrib. Logist. Manage. 28, 486–507.

Cox, S., 2002. Information technology: the global key to precision agriculture and sustainability. Comput. Electron. Agric. 36, 93–111.

Eigenmann, R., Vucic, N., Dillinger, M., Viola, K., Meyer, F., Quesada Pimentel, D., Verhoosel, J., Kaloxylos, A., Lampropoulou, I., Gábor, I., Perea Escribano, C., Sundmaeker, H., 2012. Inventory of future capabilities of Internet to meet future long and short term needs of the food sector (D700.2), In: Wolfert, J. (Ed.), SmartAgriFood reports, Munich, p. 84.

Eisenmann, T.R., Parker, G., Van Alstyne, M.W., 2008. Opening platforms: how, when and why? This paper has been published under the same title as Chapter 6 in Platforms, Markets & Innovation (ed. Gawer, 2009) pp 131–162; Harvard Business School Entrepreneurial Management Working Paper No. 09–030, pp. 131–162.

Fountas, S., Pedersen, S.M., Blackmore, S., 2005. ICT in Precision Agriculture–diffusion of technology. In: Gelb, E., Offer, A. (Eds.), ICT in agriculture: perspective of technological innovation, <http://departments.agri.huji.ac.il/economics/gelb-main.html>.

Gawer, A., Cusumano, M.A., 2002. Platform Leadership: How Intel, Microsoft, and Cisco drive Industry Innovation. Harvard Business School Press, Boston.

Handoyo, E., Jansen, S., Brinkkemper, S., 2013. Software ecosystem roles classification, Software Business. From Physical Products to Software Services and Solutions. Springer, pp. 212–216.

Hevner, A.R., March, S.T., Park, J., Ram, S., 2004. Design science in information systems research. MIS Quart. 28, 75–105.

Hossain, E., Babar, M.A., Paik, H.-Y., 2009. Using scrum in global software development: a systematic literature review. In: Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on. IEEE, pp. 175–184.

Iftikhar, N., Pedersen, T.B., 2011. Flexible exchange of farming device data. Comput. Electron. Agric. 75, 52–63.

Jansen, S., Brinkkemper, S., Souer, J., Luinenburg, L., 2012. Shades of gray: opening up a software producing organization with the open software enterprise model. J. Syst. Softw. 85, 1495–1510.
Kaloxylos, A., Eigenmann, R., Teye, F., Politopoulou, Z., Wolfert, S., Shrank, C., Dillinger, M., Lampropoulou, I., Antoniou, E., Pesonen, L., 2012. Farm management systems and the Future Internet era. Comput. Electron. Agric. 89, 130–144.
Kaloxylos, A., Groumas, A., Sarris, V., Katsikas, L., Magdalinos, P., Antoniou, E., Politopoulou, Z., Wolfert, S., Brewster, C., Eigenmann, R., 2014. A cloud-based Farm Management System: architecture and implementation. Comput. Electron. Agric. 100, 168–179.
Kruize, J.W., Robbemond, R.M., Scholten, H., Wolfert, J., Beulens, A.J.M., 2013. Improving arable farm enterprise integration – review of existing technologies and practices from a farmer's perspective. Comput. Electron. Agric. 96, 75–89.
Kruize, J.W., Wolfert, J., Goense, D., Scholten, H., Beulens, A.J.M., Veenstra, T., 2014. Integrating ICT applications for farm business collaboration processes using FIspace. In: Global Conference (SRII), 2014 Annual SRII. IEEE, San Jose, CA, USA, pp. 232–240.
Lamb, D.W., Frazier, P., Adams, P., 2008. Improving pathways to adoption: putting the right P's in precision agriculture. Comput. Electron. Agric. 61, 4–9.
Light, B., Holland, C.P., Wills, K., 2001. ERP and best of breed: a comparative analysis. Bus. Process Manage. J. 7, 216–224.
Manikas, K., Hansen, K.M., 2013. Software ecosystems–a systematic literature review. J. Syst. Softw. 86, 1294–1306.
March, S.T., Smith, G.F., 1995. Design and natural science research on information technology. Decis. Supp. Syst. 15, 251–266.
Messerschmitt, D.G., Szyperski, C., 2005. Software Ecosystem: Understanding An Indispensable Technology and Industry, vol. 1. MIT Press Books.
Nash, E., Dreger, F., Schwarz, J., Bill, R., Werner, A., 2009. Development of a model of data-flows for precision agriculture based on a collaborative research project. Comput. Electron. Agric. 66, 25–37.
Nikkilä, R., Seilonen, I., Koskinen, K., 2010. Software architecture for farm management information systems in precision agriculture. Comput. Electron. Agric. 70, 328–336.
Pedersen, S.M., Fountas, S., Blackmore, B.S., Gylling, M., Pedersen, J.L., 2004. Adoption and perspectives of precision farming in Denmark. Acta Agric. Scandinavica, Sect. B – Plant Soil Sci. 54, 2–8.
Pierce, F.J., Nowak, P., 1999. Aspects of precision agriculture. Adv. Agronomy 67, 1–85.
Porter, M.E., Heppelmann, J.E., 2014. How smart, connected products are transforming competition. Harvard Bus. Rev., 65–88, November 2014
Scholten, H., Kassahun, A., Refsgaard, J.C., Kargas, T., Gavardinas, C., Beulens, A.J., 2007. A methodology to support multidisciplinary model-based water management. Environ. Modell. Softw. 22, 743–759.
Sebök, A., Viola, K., Gábor, I., Homolka, F., Hegyi, A., 2012. Inventory of long and short term future needs of food chain users for future functions of internet (D700.1), In: Wolfert, J. (Ed.), SmartAgriFood reports, Budapest, p. 80.
Seichter, D., Dhungana, D., Pleuss, A., Hauptmann, B., 2010. Knowledge management in software ecosystems: software artefacts as first-class citizens. In: Proceedings of the Fourth European Conference on Software Architecture: Companion Volume. ACM, pp. 119–126.
Sørensen, C.G., Fountas, S., Nash, E., Pesonen, L., Bochtis, D., Pedersen, S.M., Basso, B., Blackmore, S.B., 2010a. Conceptual model of a future farm management information system. Comput. Electron. Agric. 72, 37–47.
Sørensen, C.G., Pesonen, L., Fountas, S., Suomi, P., Bochtis, D., Bildsøe, P., Pedersen, S. M., 2010b. A user-centric approach for information modelling in arable farming. Comput. Electron. Agric. 73, 44–55.
Steinberger, G., Rothmund, M., Auernhammer, H., 2009. Mobile farm equipment as a data source in an agricultural service architecture. Comput. Electron. Agric. 65, 238–246.
te Molder, J., van Lier, B., Jansen, S., 2011. Clopenness of systems: the interwoven nature of ecosystems. In: Third International Workshop on Software Ecosystems (IWSECO-2011), pp. 52–64.
TheOpenGroup, 2011. TOGAF Version 9.1, first ed. Van Haren Publishing.
Turner, M., Budgen, D., Brereton, P., 2003. Turning software into a service. Computer 36, 38–44.
Van 't Spijker, A., 2014. The New Oil – using Innovative Business Models to Turn data into Profit. Technics Publications, Basking Ridge.
Verdouw, C., Beulens, A., Van Der Vorst, J., 2013. Virtualisation of floricultural supply chains: a review from an Internet of Things perspective. Comput. Electron. Agric. 99, 160–175.
Verdouw, C.N., Beulens, A.J.M., Trienekens, J.H., Verwaart, T., 2010. Towards dynamic reference information models: readiness for ICT mass customisation. Comput. Ind. 61, 833–844.
Verdouw, C.N., Beulens, A.J.M., Wolfert, J., 2014. Towards software mass customization for business collaboration. In: Global Conference (SRII), 2014 Annual SRII. IEEE, pp. 106–115.
Wiederhold, G., 1992. Mediators in the architecture of future information systems. Computer 25, 38–49.
Wolfert, J., Sørensen, C.G., Goense, D., 2014. A future internet collaboration platform for safe and healthy food from farm to fork. In: Global Conference (SRII), 2014 Annual SRII. IEEE, San Jose, CA, USA, pp. 266–273.
Wolfert, J., Verdouw, C.N., Verloop, C.M., Beulens, A.J.M., 2010. Organizing information integration in agri-food–A method based on a service-oriented architecture and living lab approach. Comput. Electron. Agric. 70, 389–405.