SocietyByte Wissenschaftsmagazin des BFH-Zentrums Digital Society

September 2018 issue: Trust and Confidentiality

ADA-EDA: A distributed system supporting the development of agriculture 4.0 under the constraints of trust and confidentiality.

Alain Sandoz, ADA Project Manager
Ass. prof., University of Neuchâtel

A secret is some information that is only told to one person at a time.
(Popular saying)

How should confidential information be transmitted between information users in the economy and more generally in society? The paper examines a set of conditions under which this question can be consistently answered.

The need to manage and to regulate the transmission of confidential information arises in the presence of two types of actors: on one side, information providers (*e.g.* farmers) and on the other, information consumers (*e.g.* public or private database operators that collect and manage information on farms).

We assume an information provider *trusts* two or more information consumers with some identical piece of *confidential information*. The information is confidential in the sense that it is sensitive and the information provider wants to control its dissemination. Under what conditions can the first consumer to receive that information be allowed to transmit it to others that request it?

If this is possible, the information provider will not have to repeatedly send the information to different consumers (sending information can be a burden). Moreover, globally the information in the system might end up being more consistent. In some complex situations the information can be composite, each piece being trusted to some consumer, and the whole to some other.

Situations of this type occur traditionally in medicine, in education, in industry, and in agriculture for example. The paper focuses on the latter, and more specifically Swiss agriculture where this is currently a hot topic.

1. Introduction

Swiss farmers produce goods and services. In general terms, goods are edibles or other products grown and processed under well-defined conditions (respectful of animal welfare, organic, etc.). Services can be the contribution of farmers to the environment (bio-diversity), to the landscape, or to the preservation of cultural heritage, for example.

A farm is a business and a farmer is an entrepreneur. A farmer can increase potential revenues by registering to labels (which bring a premium on products), by making agreements with buyers, and by receiving payments from the Federal government and/or cantons in exchange for services.

Some of these measures are defined in the law, others in contracts, others still by actors in the agro-food market. In order to manage and to remunerate the flow of goods and services, information has to be provided along the associated logistic and value chains. This information mostly concerns the farmers and their property and is sensitive: indications on quality, prices, quantities, results of controls by the contractual partner on the truthfulness of provided information, etc. are confidential. In this case, information is mostly collected, processed, and stored in the form of structured digital data, so that at any time several datasets coexist in different databases that partially and sometimes redundantly describe different aspects of a farmer's property and business.

As an example, consider a Swiss farmer who produces wheat under some specific label, say organic (Bio Suisse) or integrated production (IP-SUISSE)[1]. The farmer supplies the label organisation with data in order for the latter to plan production, distribution, and controls, and to charge the farmer for its services (such as marketing campaigns related to the label, that support the product's price on the market). If the label organisation orders the farm to be independently controlled, some of this data will be collected again together with other attributes by the controlling organisation. The farmer also supplies the canton with similar data, in order for the canton to decide on the amount of direct payments this culture entitles the farmer to receive. Again an on-farm control is possible (collecting more of the same data) and in any case the canton will transmit the dataset to the Confederation, which will then manage its copy on its own. The data might additionally be supplied to the wheat producers' professional organisation (which defends wheat producers political interests). The farmer might also produce maize, vegetables, forage, different breeds of animals, milk, etc. For each type of production, different sets of data will be required from the farmer by different actors.

We call the farmer an *information provider* (noted IP below). The different actors or organisations that require data from the farmer are called *information consumers* (noted IC). So information providers must often repeatedly make the same information available to different ICs, possibly in different forms and at different times of the year.

Information consumers store information in databases in the form of structured digital data. Unless they consent to a collective effort (like the cantons and the Confederation in the example above), there is no reason nor any means for different consumers of the same information to use identical data structures.

To summarize, each IC requests information from the IP in the form of structured data on some predefined authenticated electronic channel (front-end). For the IC, this has three positive effects: (1) the IP is identified and authenticated on the IC's system; (2) the information is delivered in a usable digital format and values can be validated (or invalidated) using application

[1] see http://www.bio-suisse.ch and http://www.ip-suisse.ch, respectively.

logic; (3) the procedure provides a technical means for the IC to register from the IPs their commitment to the information they have delivered. Information transmission and delivery is fully traceable.

In this situation, ICs have no interest to change their information provisioning procedure, even though new demands on new information related to the farmer increase over time. New digitally supported production tools also supply the farmer with an increasing quantity of data, in turn generating new information demands from ICs.

With the increase in the number of ICs and of digital sources over the past 20 years, the problem of data transmission from farms to all types of ICs has become acute in Swiss agriculture.

This situation has led to the creation of the ADA-EDA project. ADA stands for "Agrar Daten Austausch". EDA (*Echange de Données Agricoles*) stands for the French translation of the German acronym ADA.

The paper describes the solution that ADA proposes to the problem of data delivery to multiple ICs. Not surprisingly, the approach can be generalized to other application domains under specific conditions and requirements.

2. Structure of the paper

In section 3, we sketch several problems related to the topic, that we however do not develop further in the paper.

In section 4 we mention some design principles underlying our approach.

In section 5 we state the problem in a formal manner and go through a minimal set of necessary conditions for a solution.

In section 6 we describe the distributed system architecture. The distributed system will be called ADA and we will define what it means for an IC to operate in ADA and to operate out of ADA.

In section 7 we examine how this system scales (with the number of information providers, of information consumers, and of information objects managed and exchanged between ICs). In this section we also argue why the project's aim is to distribute ADA openly and freely.

We conclude in section 8 by drawing parallels to other application domains like healthcare.

Sections 5 and 6 are relatively technical. We hope that the formal presentation will help the reader to understand some of the issues and possible solutions of transmitting confidential information in a possibly not so trustworthy environment.

3. Problems not developed in this paper

The first problem we do not develop is what happens if some information consumer deliberately sends false information to another. Because data transmission in ADA is fully traced (see below), sending false information in ADA would be fully traceable backwards to the sender. This behaviour would be like publicly lying about an IP and might incur prosecution if discovered. In ADA, ICs can choose not to send information to others, and not to ask others for information, but they shouldn't lie.

Another problem we do not develop is what an IC does with information it receives. In agriculture for example, if a government agency pays a farmer some amount of money based on some information, then the fact that this IC received this information from a third party could prevent the agency from prosecuting the farmer if it later appeared that the information was inconsistent. Conversely, an unhappy farmer could sue the government for basing its decision on third party information. In such cases, the IC must set up some verification procedure to establish the fact that the information received is really what the IP intended the agency to receive.

Finally, we do not treat the problem of the truthfulness of information, *i.e.* how and how far can an IC appreciate the veracity of the information received either by an IP or through another IC.

4. Design principles

There are no constraints on the legacy applications of ICs in ADA. Therefore the design of ADA must be that of a truly distributed heterogeneous information system. In general, heterogeneous systems are connected to a logically centralized system (possibly with peripheral slave-subsystems and satellite end-users). For example, some central resource like a database imposes strong coupling between participating nodes. This approach has limits determined by the complexity of the application domain and the liberty left to the participants of the system. Designing a sustainable centralized system for public regulation and private/business usage, for an entire economic sector, for an entire nation, and correctly solving asynchrony of heterogeneous participants, would indeed be a challenge.

As an alternative to a centralized approach, we design ADA as a distributed system, imposing homogeneity at the lowest possible level common to all ICs. ADA is a service platform that enables participating information systems to exchange data under conditions that implement a common understanding of confidentiality and trust between IPs and ICs.

For the farm as a production unit and for its owner, the platform enables vertical integration along the logistic and value chains of production, including public funding and production support using all sorts of technical, IT and communication means.

For the farmer as a businessman and for all the actors along the value chains of transformation and distribution leading to the consumer, the platform enables horizontal integration of information, where needed. Along these chains, information should flow in any direction (respecting needs and data protection constraints) from the provider of any service all the way down to the consumer.

The design principles we apply to ADA are:

- to define problems clearly and unequivocally
- for each problem, to identify a minimal set of necessary conditions that must be met in order for the problem to be solved

- to design the system architecture for each necessary condition to be realized by an individual system component
- wherever possible, to relax constraints (constraints are expensive to implement and bad for generality)

5. Problem statement and necessary conditions

In this section, we examine what conditions must be met in a distributed computer system in order to allow for some information consumer IC_1 to send some information I_{IP} to another information consumer IC_2 where:

- *IP* is an information provider that owns some confidential information I_{IP} that *IP* has trusted IC_1 with and that *IP* is willing to trust IC_2 with
- IC_1 and IC_2 know *IP* and share the meaning of I_{IP}
- *IP* trusts IC_1 and IC_2, but IC_1 and IC_2 do not necessarily trust one another
- IC_1 and IC_2 could be made accountable in case they violate confidentiality of I_{IP} as defined by *IP*

Information *ownership* (*IP* owns some information I_{IP}) is defined *in ADA* in the following sense: any information I_{IP} that specifically describes *IP*, *IP*'s belongings, or *IP*'s business is considered to be in *IP*'s ownership.

Confidentiality conditions are defined as follows:

- what an IC_i is allowed by *IP* to do with I_{IP} is written in a contract, including the action of forwarding I_{IP} to any other IC
- IC_1 is authorized by *IP* to send I_{IP} to IC_2, and IC_2 is authorized by *IP* to receive I_{IP} from IC_1

Both IC_1 and IC_2 are legal entities that operate all the necessary technical components, in particular a networked IT infrastructure connected to the Internet, that make the problem statement above technically meaningful.

To illustrate the problem statement, *IP* could be a farmer that raises sheep, I_{IP} the number of sheep in *IP*'s herd at the end of the past year; IC_1 could be *IP*'s label organisation and IC_2 could be *IP*'s cantonal authority. The farmer must provide both IC's with I_{IP}, each to its own system (as well as to other actors). How can multiple inputs be avoided for the farmer?

5.1. Identity

Problem: who is who for whom?

a) *IP* must know both IC_1 and IC_2 and trust each one of them (sufficiently at least to send each one confidential information)

b) IC_1 must know *IP* and IC_1 must have the means both to identify and to authenticate *IP* (in particular if *IP* initially transmits I_{IP} to IC_1, but in any case if and when *IP* authorizes IC_1 to transmit I_{IP} to IC_2)

c) IC_2 must know *IP* and IC_2 must have the means both to identify and to authenticate *IP*

d) IC_1 must know that IC_2 operates data transmission services and must know how to address IC_2 and how to transmit I_{IP} to IC_2

e) conversely for IC_2 regarding IC_1, with the provision that IC_2 must know how to receive I_{IP} from IC_1

f) when IC_2 receives I_{IP}, IC_2 must be able to establish that I_{IP} concerns *IP*, based on some value function $id_2(IP, I_{IP})$ that IC_1 must have sent to IC_2 along with I_{IP}. This condition is delicate and important: the only way to implement this correctly is that some actor must have sent $id_2(IP, I_{IP})$ to IC_1 previously (a good candidate is IC_2). This doesn't imply that IC_1 and IC_2 share *IP*'s identity, *i.e.* $id_1(IP, I_{IP})$ is not necessarily equal to $id_2(IP, I_{IP})$, which would be too strong a constraint

5.2. Understanding of I_{IP}

Problem: what is the meaning of I_{IP} for each one of the participants?

g) *IP*, IC_1 and IC_2 must all share some common understanding for I_{IP}, at least pairwise

h) IC_1 must know that IC_2 shares its understanding of I_{IP}. Conversely for IC_2 regarding IC_1

i) when IC_2 receives I_{IP}, IC_2 must know how to use it

j) in particular, when IC_2 receives I_{IP}, IC_2 must know when I_{IP} was meaningful to IC_1

5.3. Authorization

Problem: what exactly does *IP* authorize IC_1 and IC_2 to do?

k) *IP* must know that both IC_1 and IC_2 use I_{IP} and that I_{IP} can be sent from IC_1 to IC_2 within some predefined conditions on confidentiality

l) *IP* must have the means to authorize IC_1 to send I_{IP} to IC_2 under some predefined conditions

m) *IP* must have the means to authorize IC_2 to receive I_{IP} from IC_1 under some predefined conditions

n) at the moment IC_1 sends I_{IP} to IC_2, IC_1 must have been authorized by *IP* to do so and this authorization must still be valid

o) at the moment IC_2 receives I_{IP} from IC_1, IC_2 must have been authorized by *IP* to do so and this authorization must still be valid

p) authorizations by *IP* can be revoked by *IP* at any time

5.4. Traceability

Problem: who can be made accountable when things go wrong?

q) any action on I_{IP} that concerns both IC_1 and IC_2 must be recorded thoroughly by both IC_1 and IC_2 (*i.e.* in case of later litigation relatively to I_{IP}, both IC_1 and IC_2 must have the means to prove correct behaviour)

6. <u>Proposed solution in ADA</u>

The ADA distributed environment is composed of ADA-*nodes* and the ADA architecture is composed of ADA-*services* (see Figure 1). All ADA-nodes are *peers* within ADA. There are no other entities in ADA than ADA-nodes. With regards to ADA, all ADA-nodes are identical (they provide the same ADA-services, are structured alike, and operate under the same conditions). ADA-nodes are interconnected through a peer-to-peer private network.

Each ADA-node is operated by a legal entity (***IC_i***) that is legally responsible for operating and maintaining the node ***ADA-Node_$_i$*** and for operating a legacy infrastructure that is embedded in the ADA-node. The legacy infrastructure and the services it provides are fully independent of ADA. In particular, the legacy services are connected to the Internet uncontrolled from ADA. Legacy services are connected to ADA-services using a tailored connector (dependent both on the legacy services and the ADA-services). The ADA-services are standard and identical on every ADA-node. Peer-to-peer communication between ADA-nodes is provided by ADA-communication services running on each node.

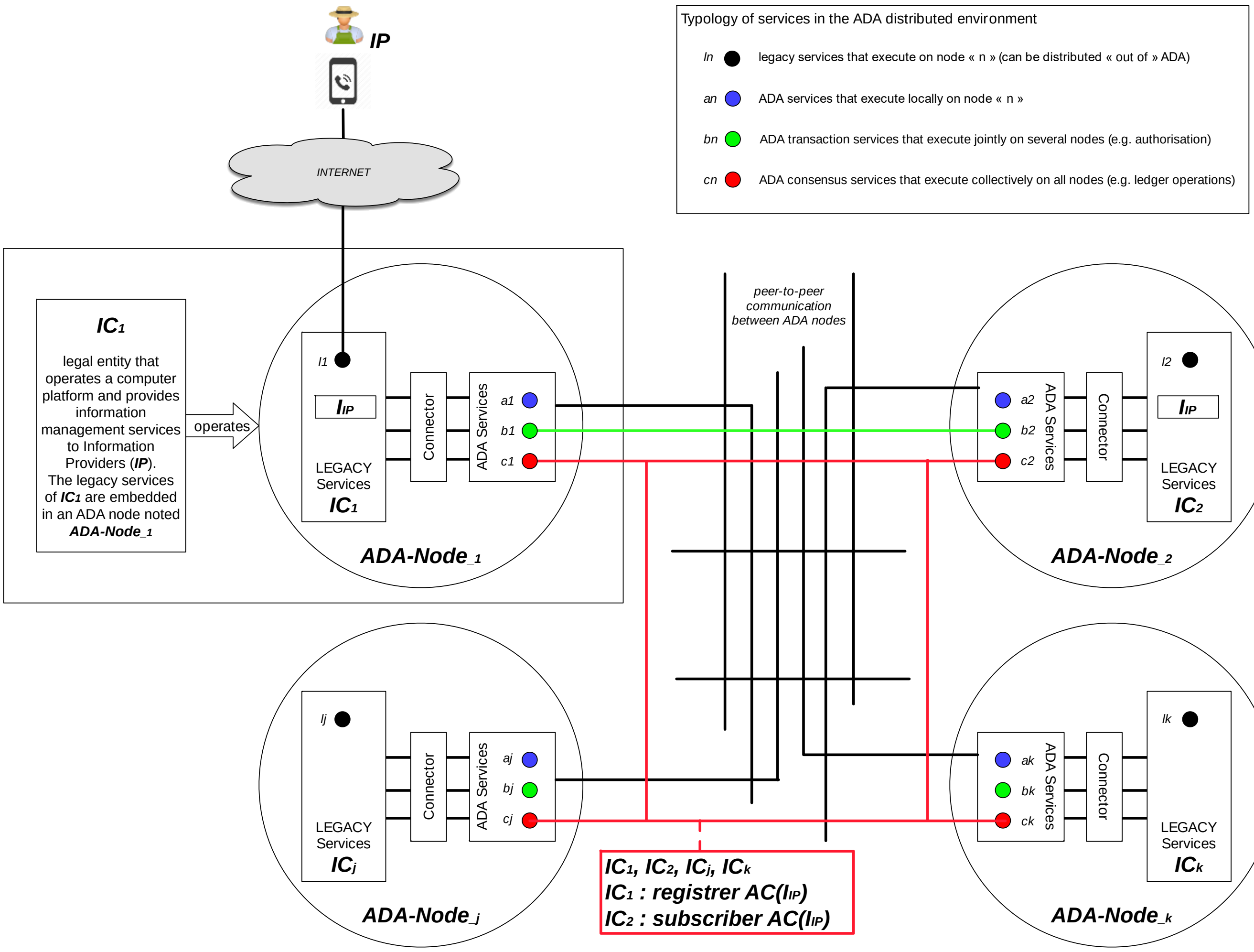

Figure 1: ADA distributed environment

The ADA-services act together in order to provide each IC_i with the necessary functionality to operate information exchange with other ICs in ADA. Each ADA service runs an instance at each *ADA-Node_$_i$* and every local ADA service instance (say AS_{ki} for the k^{th} ADA service's local instance in *ADA-Node_$_i$*) implements its functionality in a distributed transactional environment together with the subset $\boldsymbol{S_{ki}}$ of all the other AS_{nm} necessary to implement that functionality.

The subset $\boldsymbol{S_{ki}}$ might be composed of AS_{ki} alone (case of blue colored services in Figure 1) if the service is stand-alone (for example library services for the conversion units –e.g. hectares into ares).

The subset $\boldsymbol{S_{ki}}$ might be composed of AS_{ki} and one or several more AS_{kn} (green colored services) if the service requires a distributed transaction to run on several ADA-nodes.

For example, when *IP* defines the authorization to allow IC_1 to send I_{IP} to IC_2, the local authorization services AS_{Auth1} and AS_{Auth2} (green in Fig. 1) will be executing an atomic distributed transaction and its commit protocol in order to establish and trace this new authorization both in $ADA\text{-}Node\_{1}$ and in $ADA\text{-}Node\_{2}$.

Finally, the subset $\boldsymbol{S_{ki}}$ might be composed of all AS_{kn} (red colored services) if the service requires a distributed consensus to run on all ADA-nodes.

Any information consumer IC_i that wishes to integrate ADA will provide a cluster of computer systems where its own legacy services and the ADA service instances running in $ADA\text{-}Node\_{i}$ will be operated (under the legal and technical responsibility of IC_i).

Any IC_i that respects the conditions and uses the services of ADA when executing some action *A* will be said to execute *A* in ADA. If an IC_i executes some action *A* not using the services provided by this system or does not respect some condition imposed by ADA, then IC_i will be said to execute *A* out of ADA. Naturally IC_i will execute many actions out of ADA (like running its own business, internal computations or internal updates of data by its legacy services, etc.), and from time to time IC_i will execute some action in ADA.

By executing any action that has a remote effect related to some sensitive information I_{IP} in ADA, IC_i will be guaranteed not to violate any of the confidentiality conditions defined in ADA by *IP* on I_{IP}.

Below, we sketch how the necessary conditions defined in §5 are implemented in ADA.

***IP* must know and trust $\boldsymbol{IC_i}$** (condition a). If *IP* is to trust IC_i with some confidential information I_{IP}, then *IP* must be connected to IC_i using some technical means. To simplify the presentation, we consider the case where this technical means is a mobile application supplied by IC_i to *IP* over some standard mobile infrastructure (smartphone, app-store, etc.). Possibly some part or all of I_{IP} will be supplied to IC_i directly by sensors. However, the control of I_{IP} by *IP* is provided by some mobile app App_i with a front end (mobile device) and a back-end (application server and database infrastructure). Usually, this software App_i will be licensed by IC_i to *IP* under some explicit or implicit contractual conditions.

In ADA this is explicitly mandatory, and the usual contract between IC_i and *IP* must be extended by clauses that describe what information App_i might exchange with other applications if authorized by *IP* and what App_i does with this information.

$\boldsymbol{IC_i}$ must know *IP* and $\boldsymbol{IC_i}$ must have the means to identify and to authenticate *IP* (condition b, and also condition c). The extent to which the first part of this sentence (IC_i must know *IP*) is close to a formal identification of *IP*, depends on the application domain, as well as on legal, financial, and/or contractual conditions IC_i and *IP* are accountable for. If IC_i provides service against payment, knowledge of who *IP* is (what legal entity *IP* represents) by IC_i must be stronger than if App_i is a professional information service with identification of *IP* through an e-mail address and the establishment of a simple user profile for *IP*; and weaker than the case where IC_i is a government institution that pays money to *IP*.

Once again, if both IC_1 and IC_2 were to be integrated in ADA, if *IP* were to trust them both with I_{IP} and to authorize IC_1 to send I_{IP} to IC_2, this does not mean that IC_1 must send I_{IP} to IC_2, nor that IC_2 must request (or accept) I_{IP} from IC_1 in ADA.

However, if IC_i shares I_{IP} with some other IC_j in ADA, then IC_i must be able to identify and authenticate *IP* before proceeding to any operation on I_{IP} in ADA if I_{IP} is confidential, including establishing the authorization for IC_i to send I_{IP} to IC_j.

ADA supports federated Identity and Access Management for ICs under SAML 2.0, but it is neither mandatory nor necessary for IC_i to organize its identity provisioning in that way.

IC_i must know that IC_j operates in ADA, and IC_i must know how to address IC_j and how to transmit I_{IP} to IC_j (conditions d and e). The moment of integration of IC_i into ADA after certification is the time when 1) the local ADA communication component (see below) at IC_i is integrated in the ADA distributed configuration; and 2) the necessary information is provided to IC_i in order to communicate with other IC_j in ADA and conversely.

At the lower levels of service, communication between *IC*s is machine-to-machine, ciphered, fully traced. The functional connexion between IC_i and IC_j is established at the moment (during the transaction when) *IP* authorizes IC_i to send I_{IP} to IC_j. The same transaction establishes condition f (how does IC_j know that IC_i is speaking about *IP*?).

***IP*, IC_1 and IC_2 must all share some common understanding for I_{IP}** (condition g). For *IP* and IC_i (resp. *IP* and IC_j), this knowledge is provided by App_i (resp. App_j). For IC_i and IC_j, the situation is different: there has to be some technical means for IC_i and IC_j to establish that they share knowledge on objects of the type of I_{IP}, *i.e.* common knowledge about the *class* of I_{IP} in ADA (that we will note $AC(I_{IP})$ for the "ADA-class of I_{IP}").

ADA provides the services (accessible locally to IC_i but implemented in the distributed ADA environment) for IC_i to:

- register in ADA some classes $AC(I^{in})$, $n = 1,\ldots$
- subscribe in ADA to some classes $AC(I^{km})$, k=1,... ($k \neq i$), m = 1,... already registered by some other IC_ks

where I^{in} and I^{km} are some types of information managed by IC_i, resp. IC_k.

Registration of $AC(I^{in})$ by IC_i makes the class available for subscription to all the *IC*s that are in ADA. The ADA component that manages this feature is a public distributed ledger implemented and shared together by all the ADA *IC*s. Subscription by some IC_i to a class $AC(I^{km})$ in this manner then enables all the *IC*s that have subscribed to $AC(I^{km})$ (as well as IC_k) to know that IC_i shares their common knowledge of $AC(I^{km})$ (condition h).

When IC_j receives I_{IP}, IC_j must know how to use I_{IP} (condition i). Registration of and subscription to $AC(I_{IP})$ enable ICs to describe how they represent I_{IP}. Do IC_i and IC_j use hectares, or ares, or square meters to represent an area of winter wheat? ADA libraries provide conversion code between units and other software tools to establish the local representation relevant to IC_j of I_{IP} when IC_i has sent its own representation.

Who writes this code? This continuing development is devoted to the ADA software community: the community has an interest in spreading ADA and increasing the number of classes that can be used in multiple applications, without compelling editors to code strong coupling in API's or XML representations of information. Neither of the latter is sustainable at the level of an economic sector in a nation or on a continent. In this manner, ADA supports innovation in Swiss agriculture and will help Swiss agriculture contribute to the drive towards digital agriculture worldwide.

IC_j must know when I_{IP} was meaningful to IC_i (condition j). This type of condition (managing time consistently in ADA), and others like the meaning and the implementation of transitivity of authorised transmission are technical considerations that must be, and are, managed in ADA, but are of lesser interest in the context of this paper.

What exactly does *IP* authorize IC_i and IC_j to do? (conditions k – o).

ADA provides *IP* with a technical means to define authorizations. This technical means is a mobile application that associates on *IP*'s mobile device the applications App_i and App_j, and the objects these two applications share (i.e. some $AC(I^{in})$s).

Therefore, when *IP* changes some authorization (say *IP* authorizes IC_i to send new values of $AC(I_{IP})$ to IC_j when these new values arise in App_i) this change can be made to happen in one given spot (on *IP*'s mobile) at one given time (the moment IP commits the distributed change authorization transaction) and committed atomically to IC_i and IC_j within this locus in space-time.

Any action on I_{IP} that concerns both IC_i and IC_j must be recorded thoroughly by both IC_i and IC_j (condition p). We have just seen that changes in authorizations on $AC(I_{IP})$ that concern IC_i and IC_j can be recorded unambiguously on both of the ICs systems (*i.e.* in their local ADA-database service). The same is true for any service-request or request-response transmitted between IC_i and IC_j using the same ADA service, that provides the full traceability of all activities in ADA at the places where these activities have an effect. This in particular, is one of the reasons why the local ADA infrastructure must be certified before it can be switched into service locally at IC_i and IC_j.

As shown in Figure 1, ADA is built as a set $\boldsymbol{AS} = \{AS_k, k = 1,.., n\}$ of services that implement the conditions described above for all actions in ADA. ADA service AS_k is distributed and is implemented by a set of peers AS_{ki}, i=1,..,m that run each respectively on the node $ADA\text{-}Node\__i$ of IC_i (here m is the number of registered ADA nodes in the configuration).

The implementation of ADA-service AS_k depends in particular on the level of functional distribution required for that service. In figure 1, the *blue* colour is used to illustrate an ADA service AS_k implemented by instances AS_{ki} that execute the full functionality of AS_k locally at each $ADA\text{-}Node\__i$. The *green* colour is used to illustrate an ADA service AS_k implemented by instances AS_{ki} that execute the full functionality of AS_k jointly with other AS_{kj}s within a distributed transaction spanning $ADA\text{-}Node\__i$ and those $ADA\text{-}Node__j$s concerned by that action. Finally the *red* colour is used to illustrate an ADA service AS_k implemented by instances AS_{ki} that execute, if it is possible, the full functionality of AS_k jointly with *every* other AS_{kj}s, within a distributed consensus spanning all the active $ADA\text{-}Node__j$s.

- AS_{Comm}: the service provides all basic (peer-to-peer) communication services in ADA at the level of addressing of the IC_is (*blue* in ADA. The delivery semantics of messages is provided at lower levels of the OSI stack)
- $AS_{Control}$: the service provides the global transactional facility (*green*)
- AS_{Auth}: the service provides authorization management local at each IC_i, and distributed among IC_i and IC_j when *IP* defines a new authorization for IC_i to send an I_{IP} to IC_j (*green*)
- AS_{Class}: the class- definition, registration, subscription, and resolution facility (*red*). Class definition and management in ADA is specifically tailored to the problem of information transmission between loosely coupled information systems. At the opposite of API-based or XML-based strong coupling of applications, ADA relaxes all possible constraints on (shared) structures, and displaces resolution of structure-mismatch into libraries of shared, open, code (*blue*, but the latest release of that code is available on the ledger)
- AS_{Ledger}: the distributed ledger facility (*red*) that implements all shared permanent writes and distributed-consensus functionalities in ADA and furnishes read-access to ledger records over API's. The ledger is public. It contains no confidential information concerning IPs, ICs, or the information and authorisations they handle
- AS_{DB}: the ADA database facility (*blue*) local to each IC_i, that stores all state-changes in ADA that concern IC_i along with possible long term storage of received I_{IP}, for example for future use (the local database facility of IC_i can be used as a buffer to compensate for the asynchrony of other IC_ks)

ADA itself does not have a front-end: front-ends for farmers (or other types of information providers) are those mobile applications that make accessible the application App_i on *IP*'s mobile device. However ADA provides two services in relation to front-end management in ADA:

- ADA_{APP}: the mobile app used by *IP*'s to manage authorizations. The ADA_{APP} does not itself require identification and authentication of *IP*, but establishes a session with each one of IC_i and IC_j during the definition of a new authorization for IC_i to send an I_{IP} to IC_j. The latest release of ADA_{APP} executable code is always available on the distributed ledger

- AS_{Ident}: the federated identity management facility that enables ADA_{APP} and App_i to function in a single-sign-on mode using existing sessions running on behalf of *IP* on IC_i during the authorization procedure or, more simply, for daily execution of App_i

7. Openness and scalability

ADA is a distributed system infrastructure that provides services to the information consumers (IC_i) that have integrated ADA. In order for an IC_i to integrate ADA, it must fulfil some conditions and it can be later held accountable by information providers and other ICs in ADA if these conditions are not respected. ADA also provides the technical means to help the information consumer IC_i respect these conditions. Any IC_i that wishes to integrate ADA can use either these technical components supplied by ADA, or on the contrary use components implemented on its own. In both cases, the IC_i that wishes to integrate ADA must be certified before any activity in ADA can take place that involves IC_i. If the IC_i uses components supplied by ADA, then certification will be easier and less costly. If not, then the IC_i must support the additional costs of certification of its own components.

This is the reason why ADA software, respectively documentation, will be distributed under the Affero General Public license (GNU Affero General Public License v3), respectively the Creative Commons license (CC BY-NC-SA 4.0): the software (ADA components) and the documentation in ADA will be distributed openly and freely, and made available, to any actor (database operator, software editor, government agency, etc.) that wishes to study or later to integrate ADA.

This is meant in particular to facilitate innovation and drive digitalization of Swiss agriculture. ADA's perspective is that Industry 4.0, also in agriculture, is meaningful and sustainable only if data produced by sensors, devices, and software in the application field can be used multiple times with increased value. This in turn is only possible if new software, extended possibly with new sensors and new devices, can use this data; meaning in particular that this data must be transferable in good conditions from the former application to the latter. Therefore, ADA must be open, whereas the data not necessarily: the *IP* decides to what extent it wishes its data to be open, and only the *IP* has the right to make this decision in ADA.

The ADA concept is based on the idea that ICs (and their software) have specific needs and specific targets (in terms of users, of market, of functionality, etc.). Parts of the information these systems use (and for which they have internal structured representations) exist elsewhere in other systems (provided systems share not only the objects, but also the users) so the IC might be willing to send and receive information when new values arise. On the other hand, ICs have technical or commercial secrets they are not willing to give out.

So the exchange of information is not in first line a technical IT problem. It's a problem of finding the right information objects to exchange with the right ICs concerning the right information providers, and this at the right time.

In this sense, the numerical complexity of the data flow in the ADA system is by far not of the order of number($AC(I_{IP})$) x number (IC_j) x number (*IP*) (or worse of number(I_{IP}) x number (IC_j) x number (*IP*)) which it could be if the system were designed in a centralized way. Moreover, in agriculture, time runs at the speed of nature, of the seasons, of the growth of plants and animals, and on very limited scales in a country like Switzerland. New values that are worthwhile to exchange between systems do not arise at a high frequency on farms. As we described above, the real problems are 1) the cost of media breaks between systems that could share information, compelling the farmer to loose enormous resources just to manually breach these gaps from time to time; and 2) the cost of building and maintaining a strong coupling between all the ICs and a central control.

The number of animals raised, the areas cultivated, the yields, the number of people implicated, even the number of products and the amount of produce, at the scale of the nation, grow slowly, and sometimes tend to decrease over time.

Scale and scalability in ADA (for agriculture) are not considered to be a problem. Our early computations have shown that the additional bandwidth necessary to operate ADA would not be significantly measurable (in the case it was positive).

The main technical problem to scale ADA upwards will be the ability for ICs to integrate ADA. This is why ADA is built as a, mostly generic, service infrastructure, distributed freely and openly, and concentrating in the software distribution all but a very restricted number of operations that are linked to the specifics of each IC's legacy system.

This is a common approach in modern software development, especially in the Internet technology: every modern information system is connected to the Internet, meaning every system technically integrates the service infrastructure of the Internet, a large portion being run locally and replicated identically at each Internet node.

In the Internet, some low level services (like routing or domain name resolution services) do not run locally. This is not the case in ADA. ADA resides within layer 7 (application layer) of the ISO-OSI model and is designed in a way as not to exhibit any functionality that is not fully distributed.

8. Homomorphism with other application domains

The problem we've described in this paper is widespread in today's connected world: people's data (*IP*'s confidential information) are input, copied, transferred, cut into pieces, joined, and redistributed in any possible and often non-conform manner. Trust between some *IP* and several IC_k is possible, but what IC_i and IC_j do with *IP*'s information is neither controlled nor controllable. The information provider is doubly burdened with the tasks of multiple input of the same information in different forms on different systems, and managing what information was sent where for what purpose.

This situation arises, for example in healthcare: a patient (*IP*) visits his or her family doctor (IC_i) who orders some exams, studies results (I_{IP}), makes a provisional diagnostic (possibly incomplete), refers *IP* to a specialist IC_j and asks *IP* permission to send I_{IP} to IC_j. Then I_{IP} is copied, scanned, or faxed from IC_i to IC_j, and both copies live on, uncoordinated and uncontrolled by *IP*. Public authorities have imagined very complicated systems to manage this situation. To release constraints without impairing either trust or confidentiality is however possible, and allows for feasible solutions under specific conditions.

Exchanging information in ADA and in health-care are homomorphous to a point that is worthwhile studying and developing. The same seems to apply to other application domains that span the interface between the public sector and the private sector in relation to confidential data management.

9. Conclusion

ADA is currently under development. Two tracks run in parallel: the formal definition of requirements of ADA services by a team at the Bern University of Applied Sciences will lead to the development of a first free open-source software release of ADA during the year 2019. On the other hand, the development of pilot demonstrator instances within the application domain of agriculture will be used to experiment the ADA concepts (the distributed authorisation services and the sharing of loosely coupled data structures between ICs) already in 2018.

10. Acknowledgements

We wish to thank the ADA Requirements Definition team at the Bern University of Applied Sciences for their careful, patient, and much appreciated proofreading, asking sharp questions, and making constructive suggestions on earlier versions of the paper.

180820 V3.1 / ASA

Alain Sandoz, ADA Project Manager

Ass. prof., University of Neuchâtel