Designing a distributed system for data sharing between competing actors: building a dependency-free path between transparency and opacity

Conference Paper · February 2022

CITATIONS
0

READS
43

1 author:

Léa Stiefel
University of Lausanne
10 PUBLICATIONS **1** CITATION

SEE PROFILE

Designing a distributed system for data sharing between competing actors: building a dependency-free path between transparency and opacity

I. Introduction

I am going to talk about a particular object, in the sense that one could think at first sight that it concerns only computer engineers and technicians. The object is systems architecture, and my talk will be devoted to showing you the interest and the importance of this object for social sciences.

I'm not the first to find an interest in systems architecture. Social scientists have in fact devoted three bodies of literature on it: Systems and Organization Studies; Platform Studies; and Internet Studies.

Systems architecture is viewed differently by these bodies of literature. Systems and Organization Studies view architecture as a design process. Platform Studies and Internet Studies view architecture as the result of a design process. This difference in perspective on architecture leads to different questions.

Systems and Organization Studies are concerned with the strategies and political choices of the architects that drive the design process. Platform Studies and Internet Studies are concerned with the political effects of the result on the behaviors and interactions among the users of the system. Despite this difference, both groups consider systems architecture as an object with political stakes.

I share this view, more precisely I share it because I've experienced it as a social scientist. My presentation will be devoted to bringing this experience to you. I will place myself in the first group of literature because what I've experienced and investigated in the field was a systems design and development project; and therefore a design process during which strategies and political choices were defined.

The argument that systems architecture is interesting and important for social sciences because there are political issues at stake is not new. So why is what I am proposing original, enough to deserve your attention? The reason is the following: the literature, especially Systems and Organization Studies, has argued that architecture as a design process is about political strategies and choices, but this has not been demonstrated empirically.

"(…) More research is needed to support the view of architecture as decision"

[Smolander et al. 2008]

"[Let's] leave the battlefield of defining architecture and instead look at how architects actually carry out their business in an everyday manner"

[Scheil. 2008]

"Despite their importance, there are just a few studies that consider the work of IT architects (…). Their work has not been sufficiently explored in empirical studies either (…)"

[Figueiredo et al. 2012]

We find the same conclusion in the engineering literature. And here, I'll give you just one quote:

"Unfortunately, really good engineering and architecting case studies are notoriously hard to obtain. The stories and details are rarely published. Books published on major systems are more likely to focus on the people involved than on the (technical) decision making"

[Maier 2009]

As you may have noticed, my quotes date back to 2008, 2009, 2012... In ten years, hasn't there any progress been made? As far as I have found, not really. Except by a small group of Anglo-Saxon, or more precisely Nordic, researchers. These colleagues have indeed documented cases of architecture as a process. However, the systems they studied had already reached an advanced infrastructural stage, that of large-scale established systems. Therefore, the strategies and political decisions they documented were aimed at integration. That's much further downstream from what I'm going to talk about.

I will discuss the political strategies and decisions made and reflected in the *initial* design of the system. Their goal was to interest and enroll heterogeneous, autonomous, sometimes competing and mutually distrustful actors in a system intended to make them collaborate.

It is now time to present my case study.

II. Case Study

A. Background

I'd like to first give you some background on the context in which my case study emerged. Some of you may be familiar with this context, so, I'll keep it brief.

In 2015, a large, centralized data warehouse project emerged in the Swiss agricultural sector. Its goal was to collect all the data from all the farmers into one database. A bold, one might say, unrealistic goal. The sponsors' argument was that with only one database for the whole sector, the farmers' administrative work would be simplified compared to their current situation where they had to feed a panoply of databases separately. The project's sponsors were a national agricultural extension center (experts working with farmers) and an IT company close to the regulatory authority (the Government).

Two years later, in 2017, they decided to build a centralized smart farming platform on top of the embryonic database. Their argument was that smart farming would enhance the competitiveness of farms. Other players quickly joined the platform's shareholders. Among them was the largest agricultural cooperative in Switzerland, which is both the main supplier to farmers and a major buyer of their products.

The vision of a centralized warehouse and then of a centralized smart-farming platform did by far not convince all players in the agricultural sector.

First, not all farmers were convinced. With all their data in a unique database, in the hands of a cooperative that would supply them with what they needed for their production and then would buy what they had produced, the risk of falling under the total control of that actor was too big. The latter could then be able to dictate what to produce, when, how and to what price. For the farmers who still considered themselves as free entrepreneurs, becoming de facto employees of a big cooperative was out of the question, even the more that they would continue to bear all the risks.

So, farmers, let's say many mostly small farmers, opposed the project. But not only them. The organizations in the agricultural sector, such as public organizations, for example the cantonal administrations, or private organizations, for example certification bodies, also feared to depend on a conglomerate of private actors associated with a central platform to manage their business. These organizations had their own databases, fed directly by farmers, which

they consider appropriate. For vague promises of "simplification" of the data handling, these organizations would have to overthrow all their business processes? And then what? Who will assure that the platform would provide them with the data they needed when they needed it? And who will assure that they wouldn't be charged to access their own data?

These are some of the reasons why some farmers and organizations did not want to endorse that the "all data on one platform" part of the project goes ahead. BUT there was a problem. The federal government was publicly supporting the initiative and several organizations were pushed to join the project, albeit a small number, but that number could grow. For those who were the most concerned, a way to stop the project had to be found.

Two farmers' associations, representing almost 50% of the farmers in Switzerland, decided to launch a counter-project in the form of a fully distributed platform for the transmission of data between their databases, under the authorization of the data-owners, *i.e.*, the farmers. This project had similar objectives: to simplify the administrative work of farmers by avoiding double entries (here by transmitting data already entered) and to promote the competitive development of farms (here by enhancing the creation of innovative services based on the exchange of data). However, the proposition to achieve these goals was diametrically opposed to the centralized platform: the platform would be fully distributed among stakeholders.

B. A distributed system

This brings me to the system I have been studying, I'm talking about the distributed platform. Not the centralized platform, which has been the subject of much controversy, and of which I have only presented a few aspects.

But before, I will briefly outline my methodology.

In September 2018, I went behind the scenes of the distributed platform project. I followed the day-to-day design and development of the platform until it went into production as a first operational prototype in July 2019.

During this year of ethnographic fieldwork, 1) I collected all the documents and emails produced within the framework of the project; 2) I attended, recorded, and transcribed all the working sessions of the team associated with the project; and 3) I conducted, recorded, and transcribed about ten interviews with the members of this team. These elements compose my materials. They form the basis of my presentation today.

I also conducted, recorded, and transcribed about 40 interviews with Swiss farmers and leaders of agricultural organizations. This material allowed me to provide the context for the emergence of the distributed platform. With that said, I can now begin to present my analysis.

So, let's go back to the distributed platform.

What is a distributed platform, or a distributed information system? I think it's useful to tell you in general terms, that is, without going into detail, what it is, from a technical point of view. I don't think everyone here knows what technical distribution means, and without some introductory premises, my argument might be hard to follow.

A distributed computer system is a system composed of nodes interconnected by a network. The nodes are hardware components on which software runs. Somewhat like your computers, but bigger. Your computers are also interconnected by networks, either locally through a Wifi or more globally as by the Internet. However, a distributed information system delivers a global functionality to the users of its nodes (not just local functionality replicated at each node). Moreover, if these nodes are functionally identical, that is, if the software that delivers the

global functionality is the same on each node, then the system can be said to be peer-to-peer. Each node is an equal peer in the network. It can do exactly what the others can do and nothing more.

Now, this network of nodes can be public, which means that anyone can access it. Or it can be private, which means, let's put it this way, you must get permission to enter it.

The system I studied is a private or permissionned peer-to-peer network. This is not the type of system we usually encounter. The systems we usually encounter are centralized systems. Facebook is a centralized system. The smart farming platform was a centralized system. Gmail is a centralized system. All these systems are centralized: they each are made of singular nodes that concentrate functions, software and data and distribute the service to slave-clients. As far as the service is concerned, there is no locally productive software on your home computer or smartphone. You are not a peer of Facebook; you are only a client of the Facebook server.

So much for the "technical" basics of distributed systems. I now propose to enter the distributed system I studied, the distributed data sharing platform.

III. Case Study Analysis

A. Rationale for distribution

As we have seen, it was decided to block the centralized smart farming platform project by launching a counter-project for a fully distributed platform. This was an initial strategic and political decision, made by its architect, accepted by the sponsoring farmer organizations, and embedded in the very design of the platform. The platform would be fully distributed. But why exactly? What was the architect's rationale, beyond his conviction that only radically different methods from their powerful opponents could eventually succeed in defeating them?

Let us listen to the architect. I might say it right away, since the verbatim mentions it: ADA is the name of the system we are analyzing here. ADA stands for Agrar Daten Austausch, or Agricultural Data Exchange.

The main and foremost constraint imposed by the context of ADA is that ADA is a fully distributed system. The reason is the following: the information systems that interact with ADA are independent, autonomous, heterogeneous, operated by different legally responsible organizations, that are possibly competing on some market.

[ADA system's architecture document, February 18, 2019]

The network of actors that the system addressed was composed of organizations from the agricultural sector and of farmers. They were, as the architect says, heterogeneous, autonomous, independent, and potentially competing actors. In this context, and for the architect, it was unthinkable to propose a system whose components, or even a single component, had to be operated by some specific actor or subgroup of actors. Which actor or group of actors could have played this role of an intermediary, in all neutrality? Which trusted actor could have been by unanimity designated to play this role of a middleman? None in the long run. So, a system without any central component was required. The system had to be fully distributed.

Let's continue the verbatim sequence:

The design choice that is made in ADA as a consequence of this situation, is that every operation that is executed in ADA is executed under the responsibility of exactly one actor directly concerned by data transmission

[ADA system's architecture document, February 18, 2019].

It was also necessary for the architect that each action performed in the system could be attributed to the actor responsible for it.

The system was designed to allow the transmission of authorized data. So, what were the actions that could be performed in the system? Several, but for now, let's say essentially three. First, for a farmer, the owner of the data, the only possible actions were to give (or withdraw) his authorization for one organization to transmit his data to a second organization. Second, for an organization, the only possible actions were either to send data to a peer, or, third, to receive data from a peer, provided the transfer had been authorized by the farmer.

Each of these actions (authorize, send, receive) had to be traceably assigned to the actor who was responsible for it. Why? Because otherwise, for the architect, these actors could have done anything they wished with data. An organization could have transferred the farmers' data that it managed to another organization without worrying about the owners' consent, since it would not be held accountable for the transfer. Or a farmer could have sued an organization for transferring his or her data, claiming that he or she had not given consent, since he or she could not be held responsible for the authorization. For the architect, such scenarios were not acceptable to either the organizations or the farmers. Furthermore, such scenarios would have prevented the implementation of the system, as they would not have complied with the legal framework on data protection.

For every action in the system to be attributable to its owner, the architect argued, all components of the system used by an actor had to be under his control. In particular, each node of the peer-to-peer network, from which data transfers could be made, had to be under the legal responsibility of only one organization. So much for the "raison d'être" of the distributed nature of the system.

I would now like to explore the how. How could the system make sure that each actor could be held accountable for their actions? To understand this, we must now enter the composition of each node in the peer-to-peer network.

B. A logger component

Each data transfer involved three actors: a farmer having previously authorized the transfer of his data, an organization sending the data, and a second organization receiving the data. How could these three actions be assigned to their respective actors? For the architect, the solution to this problem lay in the integration of a "logger" component in each node of the peer-to-peer network.

One of the constraints of ADA– and this is also a contractual constraint that ADA nodes will have to fulfil– is that whatever information they transmit, it has to be persistently logged in the system. But locally. So, in particular, it would be possible for an information provider, owner of information, to trace what has been done with his information in ADA.

[Transcript Workshop specification and development teams, September 28, 2018]

What is a logger? It is a file system, in which a series of records can be dropped and stored. For the architect, each data transfer, in which an organization, a node in the peer-to-peer network, would participate, as sender or as receiver, was to be recorded in the logger component of that organization's node.

The authorization given by a farmer, to organization A to send and to organization B to receive respectively, would be recorded in the logger components of A and of B. Likewise, the transfer of the authorized data would be recorded in the logger components of A and of B.

To add one more layer, each data transmission would have been signed, signed by the sender when sending and signed by the receiver when receiving, a signature that the sender would have received through an acknowledgement sent by the receiver. And as long as this signed acknowledgement had not reached the sender, the latter would not initiate any further transfer with that recipient.

Of course, since each logger component would have been under the control of the organization operating its node, the latter, say organization A, could have decided to delete the contents of its logger concerned by the transfer in question. And thus, claim not to have participated in the transfer. But organization B could have exposed its logger and shown that, yes, this farmer, say John, had indeed given him permission to receive this data that he had indeed received it from A, as evidenced by his signature. And vice versa. Let's hear the architect speak:

If, on the other hand, both organizations had decided to remove the relevant content from their loggers, things would have been more complicated... But it was at least possible to make the occurrence of a hack apparent on the loggers. How could this be done? By programming the logger components, as decided by the architect, to number each record. In this way, a deleted record would have been made visible in the number sequence, exposing a missing number.

To assure that each "correct" actor could prove his bona fides and that each "scheming" actor could arouse suspicion was the only way to proceed in a distributed system, according to the architect. But it was already a good guarantee, better than what could be observed in the sector, with controversial cases of data leakage of unknown origin and without a designated person in charge.

The architect saw this as a key asset for the interest and enrolment of stakeholders in the peer-to-peer system. They could say: at least in ADA, each peer-organization can be held accountable for its actions, I want my data to flow only through them.

C. A "ledger" component

I told you earlier that essentially three actions could be performed in the ADA system. At the same time, I informed you that other actions could be performed. I would now like to discuss some of these other possible actions, which are also very interesting for outlining the strategies and political decisions that are made during a design process and then become part of the design of the system itself.

These actions were: as a sending organization, being able to publish the type of data I was willing to send, as a receiving organization, being able to publish my interest in receiving that type of data. The architect wanted these actions to be mandatory before any data transfer was to take place, and to be made public, *i.e.*, visible to all farmers and partner organizations of the network.

But what is the interest, you may ask? First, from a purely practical point of view, it could be useful for network member organizations to know what types of data were being made available. Second, and less obviously, it could be useful for a farmer to know that an organization to which he or she wished to send their data, for example to avoid duplication of data entry, had expressed an interest in receiving it from a third party.

Indeed, under the ADA system, the sending organization decided de facto to whom it would or would not send the data that was in its care, as long as the transfer had been authorized by the farmer. It could, for example, have decided not to send the data to an organization that it considered as a competitor. Without visibility into the interest of the receiving organization, the farmer would not have known whether the organization he wished to receive his data was not receiving anything because it had not requested it or because it had been "boycotted" by the sender. With the publication of this information, he could know this. And with that knowledge, he could pressure the sending organization to send his data.

In the architect's worldview moreover, these types of data that organizations were willing to transmit had to be "ordered". The first to propose this or that data set had to be able to be recognized as the first to have proposed it. Why? So that this good idea would not be stolen. Innovative ideas of a small player should not be taken over by other more powerful players, published in their name, or simply put aside.

As a reminder, the objective of the data exchange was not only to simplify the administrative work of the farmer by avoiding double entries. It was also to support the development of innovative services based on the data exchanged. A group of "innovative" actors could thus organize themselves around the data present on each other's systems and propose value-added services based on them. For the architect, no actor should have been able to hinder this kind of initiative by making the data "catalog" disappear or by claiming to be the author of the catalog; and then decide not to transmit the data.

These were some of the architect's strategic visions. And to make the data types in the ADA distributed system 1) visible to all, 2) persistent and 3) ordered, one technology proved very useful: blockchain.

A blockchain is functionally the same as a logger: a file system in which a series of records can be dropped and stored. Except that, unlike the logger, where a single node unilaterally requests the deposit of records, in the blockchain, all nodes in the network participate in this action and the deposit is made consensually. These records then become visible to all, they persist (namely they cannot be destroyed) and they are delivered in the same order to all, request after request.

One may ask: Why not use the blockchain to record every action performed in the ADA system? Why, for example, not use the blockchain to record *all* authorized data transfers made in ADA?

For the architect, there was no reason for these authorized data transfers, which are private information shared only between two actors under the farmer's seal, to be made public.

Blockchain, as interesting as this technology is, is also costly and risky. In order to deposit any record, all nodes of the network must be involved. And failures (network breakdowns) are very likely. To the architect, using this technology to manage such sensitive information could have been fatal for the sustainability of the system and the collective action of the actors.

To quote the architect one last time:

Global persistent state values are difficult to establish and maintain, in fact impossible unless very specific conditions are met (consensus problem, byzantine general's problem, etc.). The technologies and projects are under constant evolution, it's expensive to deploy and operate. Its usage must then be carefully tuned to needs.

[email, Mai 2-5, 2019]

There would be much more to say (about how and why the ADA system was built as it was). But time is short, and I think I made my point: Architecture is a design process in which strategies are pursued and decisions are made that are political in nature. They are political in the sense that they determine the *rules of the game*, the conditions under which the actors will carry out their collective action.

Conclusions

Nevertheless. I still have to say something about the title of my talk. A distributed system for data sharing between competing actors. Building *a dependency-free path* between transparency and opacity. I will use the conclusion to do that.

Everything I have described to you, the architecture of the ADA system, is just code. Lines and lines of software code.

The nice thing about digital architectures, which are ultimately just code, is that - unlike the architectures of the physical infrastructures we know, think of our highways - they can be transformed, or even remade from scratch, if desired.

ADA was a free and open system in the sense that anyone, including organizations in the agricultural sector, could have used its code to do whatever they wanted with it. So, if the organizations were not happy with the way the architect had designed things, they could have transformed it.

Depending on how the system is initially designed however, it may be more or less easy to redesign. I would have liked to explain how ADA was designed in such a way as to be easy to redesign, technically and more broadly socio-technically. I would have liked to report which were the architect's visions and arguments about this. But I'll have to save that for a future talk.

For now, I hope you'll leave this presentation with some new knowledge or some new ideas. For me, the goal was to show what a system architecture means and why this object is interesting, and important for us as social scientists. System architecture is a design process in which strategies and political decisions are made. These strategies and political decisions determine the rules of the game, the conditions under which the actors can carry out their collective action. And while these rules-conditions are enacted by the architects, they can be renegotiated over time. As strategies and political decisions should always be debatable, they should even be negotiated from the start, collectively. For this to happen, we, field actors and/or social researchers, need to be aware of what is going on in the design of systems.