Role of Control Flow in Interoperable Services

Emmanuel ORAIN
Airial Conseil
3, rue Bellini
92806 Puteaux
France
Tel. +33 (0)1.41.02.89.00
Fax. +33 (0)1.41.02.89.25
emmanuel.orain@airial.com

Abstract : The present paper explores a specific aspect of eGovernment interoperability, the **control of eProcedures** involving multiple government agencies. It is based on work performed in the frame of the IST Project Terregov – "Impact of eGovernment interoperability on territorial governments". This project had to face two seemingly mutually exclusive constraints : the necessity of a centralized control of the eProcedures, and the impossibility of using a centralized flow of data. This paper explores some of the reasons behind these constraints, the architecture we built in order to integrate them, as well as some additional benefits.

1 Introduction

***e*Government interoperability** is a key technological instrument for offering public services that are cross-organisational, cross-level, transparent, integrated, secure, decentralised and available anywhere at anytime. It finds its roots in the **citizen/service re-orientation** taking place in public administrations all over Europe and is driving the development of *e*Government services.
In fact *e*Government is not just the electronic version of traditional administration but represents an opportunity to deliver better services to citizens and businesses, effective and more efficient. The electronic implementation of such re-designed processes, called **eProcedures** in this paper, thus implies major **organisational changes** in government processes and in inter-operations between administrations. This means in particular the redefinition of the value chain for each specific service, with the clarification of the roles and responsibilities of each actor (decision power, legal responsibility, interface to the citizens, …).

The present paper explores a specific aspect of eGovernment interoperability, the **control of eProcedures** involving multiple government agencies. It is based on work performed in the frame of the IST Project Terregov – "Impact of eGovernment interoperability on territorial governments".

This project, Terregov, aims at providing a framework for supporting the implementation of *eProcedures* that typically require the involvement of *several* administrative agencies or organizations; interoperability is at the heart of Terregov : it focuses on services provided by local and regional-level administrative agencies and organizations, and on the interactions that must happen between them. Terregov supports eProcedures that are targeted both to agencies and to citizen (although it is likely that most citizen-oriented eProcedures will actually be started by civil servants with whom the citizen will be interacting in an agency or an organization)

2 Background : a Service Oriented Architecture

Even with a regional or local scope only, Terregov needs to interact with *many agencies*, each one having its own peculiarities, its own legacy systems, its own specific structure, etc. It is unrealistic to expect all agencies to eliminate their current system, in order to use a new global cross-agency system, imposed from "the outside". The aim of Terregov is certainly not to replace the existing systems within agencies, but to allow them to interoperate. Locally, Terregov will interact with each agency's legacy system. In order to communicate *between* agencies, the approach we have chosen is to base Terregov on a *Service Oriented Architecture*, and to present the agencies' services as *Web Services* : each service an agency wants to allow access to within the Terregov framework will be exposed as a Web Service by this agency. In the course of eProcedures, these agencies' Web Services will be called in a specific order depending on each specific eProcedure definition.
Within an agency, some of the Web Services will sit on top of existing services, provided by the legacy system (*"legacy"* from Terregov's point of view, of course) and will essentially *wrap* them, providing the same service to the outside, but through a Web Service interface; some other Web Services will offer Terregov-dedicated services, which make sense only in the frame of Terregov (we will describe one such service later in this paper : the *Data Access Module*).

3 Centralized flow control, Monitoring and Ownership

To make the following discussion more clear, let's take a simple example of a procedure :

A citizen goes to the city hall, meets with a civil servant, and makes a request, which involves the civil servant to fill in a specific form. The form describing the citizen request must then be processed by another agency, which we will call Agency B. Agency B checks the request, performs some task related to this request, updates its own records, etc. Then, another agency, which we will call Agency C, is required to process the request, update its records, etc., and finally send some notification back to the originating city hall, which can in turn inform the citizen about the request's outcome.

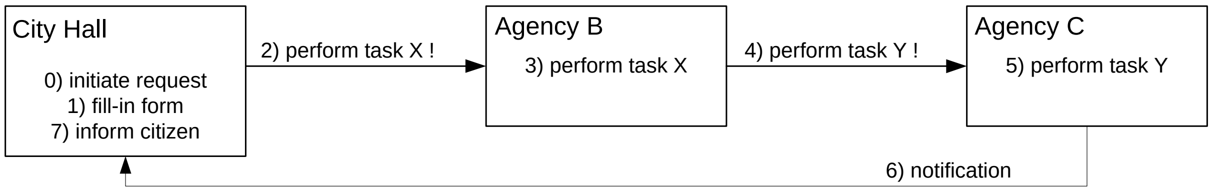
Even such a simple and mostly linear scenario raises an essential issue, when we consider how it can be supported by an interoperability framework like Terregov[1] : the issue of distributed vs. centralized flow control of the eProcedures.

[1] Note that this paper does not address the specific issue of *electronic* data exchange. Procedures that involve multiple agencies require a certain amount of data to be exchanged between these agencies. Today, these data are typically exchanged *manually* : a paper form is filled in an agency and sent to another one, either by mail or by

After the city hall civil servant is finished filling in the form for the citizen, this form must be sent to Agency B. Today, this form sending is typically handled by the city hall itself. Later, when Agency B is finished with processing the request, the request must be forwarded to Agency C, and this forwarding is typically handled within Agency B. Finally, when Agency C is finished with the request, the city hall must be notified. Again this notification is sent by Agency C itself. The important point here is that each *agency transition* is determined and performed *locally*, i.e. within the agencies themselves. This actually makes the *global procedure* itself *virtual*.
The procedure can usually be *described* at once, i.e. globally (for instance, we just described a procedure earlier in this section); however, it is not *implemented* globally. Note that we are not speaking here of the *internal* steps that an agency must perform in order to process a request – such steps are obviously local to each agency – but we are focusing on the *inter-agency transition* steps, which are described at the procedure-level (i.e. they are the steps described in the procedure itself : the procedure description does not usually include the smaller steps that are internal to each agency; typically, the sequence of steps that happen inside an agency, such as the sequence of operations that occur within Agency B when it has to process the citizen's request, appear as one atomic step in the global procedure description).
Of course, these local transitions are deriving from the global procedure definition, and each civil servant in each agency *can* probably refer to this definition in order to know what to do next, but there is usually no global entity that controls the actual unrolling of the whole procedure. We can make a parallel with a typical Web-based application, usually comprised of many pages, in which the (server-side) code included *within each page* knows which other page it should send its data to. Even though a global procedure is "followed", this global procedure *as an entity* exists only on paper and in the documentation of this application, and its implementation is distributed all across the code included in the pages : here again each decision on what to do next is local.
One important question is : when we move from manual procedures to eProcedures, do we want to keep this local advancement scheme, or do we want to offer an explicit support for globally/centrally controlled procedures ?

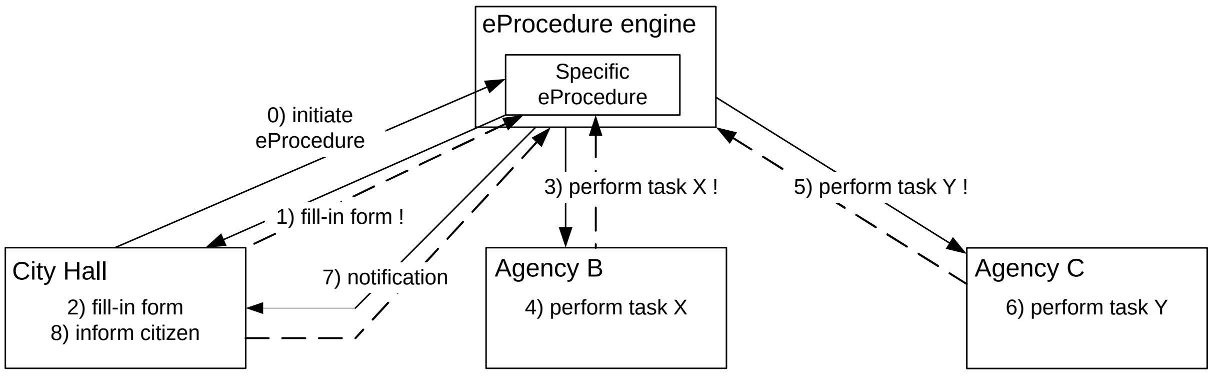
Let's describe an alternative : in a centrally controlled eProcedure scheme, an agency, once finished with its local processing of a request, instead of calling the "next" agency, would simply *return* the control to a central component, responsible for calling the next agency. Such a component would act as an "eProcedure engine", running the eProcedures from start to finish.
The following diagram shows the difference in the control flow between the two schemes :

Distributed flow control scheme :



fax, and the data obtained from this form is read and manually re-typed in the second agency. An interoperability framework that aims at providing electronic data exchange to replace these manual exchanges must address numerous issues (such as the interconnection with legacy systems, guaranteeing the *meaning* of the exchanged data across agencies, etc). An important part of the Terregov project is devoted to these issues, but they are out of the scope of this paper.

Centralized flow control scheme :



Note that in the distributed flow control scheme diagram, all the transition steps (2, 4 and 6) are not really correctly named : for instance, step 2 should be called "perform task X, *and make the procedure progress after that !*" (each transition steps in the diagram correspond to a call to a Web Service in the target agency)
However, in the centralized control flow scheme diagram, the calls are correctly named : they perform what their name says, and nothing else. Once the task is finished, they return control to the eProcedure engine (see the dashed arrows accompanying the transition steps)

If the experience gained during the recent years of Web-based application development can be of any help for our eProcedures environment, the answer to the previous question should be a resounding "*Yes, we do want a centralized flow control scheme !*"

Here are some of the most compelling reasons for using a centralized control flow for the eProcedures :

- **Understanding** : when the flow control is distributed across all the pages, it is almost *invisible* from the code only. One has usually to mentally rebuild the flow of the eProcedure from all the services' source to understand the entire application, in order to be able to debug it. This issue is much worse with eProcedures than with Web-based applications, since with eProcedures, the code is scattered over several agencies services, whereas with a Web-based application, most of the pages are physically stored on one server only. On the other hand, with a centralized scheme, the procedure is implemented as it is described globally.
- **Level clarity** : with a distributed control flow scheme, eProcedure-level steps (which services of which agency must be called in which order) and intra-agency steps (with which operations an agency actually implements a service) are mixed behind a unique Web-Service interface. Even if, internally, the separation is made explicit, this is not visible from the outside of the Web Services (hence the remark about Web Service naming in the diagram above)
- **Reuse** : with a distributed control flow scheme, the eProcedure's logic lies within the agencies, and at least part of the agencies' services code is dedicated to the eProcedure logic (in the worst case, the code implementing the eProcedure logic and the code implementing the agency-service can be hopelessly intertwined). In order to reuse an agency service in another eProcedure, the eProcedure-logic part of the service's code would have to be modified, to take into account the possibility of being part of two

different eProcedures. This type of modification becomes more and more complex with each additional eProcedure the service has to be a part of.

- **Updating and debugging** : with a distributed control flow scheme, updating is difficult, because when the eProcedure control flow has to be modified, the modification is often spread across several services, themselves spread across several agencies. Debugging and testing are also made difficult, because each service *must* be tested in the context of each eProcedure it belongs to. Unit tests are insufficient.

In addition, the potential impossibility of accessing the code of a service within a given agency, thus making a change in a eProcedure impossible, prevents us from relying on a distributed control flow scheme for the eProcedures.

As a result, an eProcedure, in Terregov, is similar to a Business Process, defining explicitly the sequence of which services must be called in which agencies, where each service returns control after having completed its task. Note that the example presented earlier is very simple, and involves only a linear sequence. Real life eProcedures include more complex control flow, with loops, branches, choices, etc. In addition, note that most services in agencies may take a very long time to answer, and that most the interactions between the eProcedure engine and the agency services will happen in an asynchronous way[2].

Using a centralized control flow scheme for running the eProcedures brings two very positive points, especially important in the area of eGovernment : eProcedure *monitoring* and *ownership*.

3.1.1 Monitoring

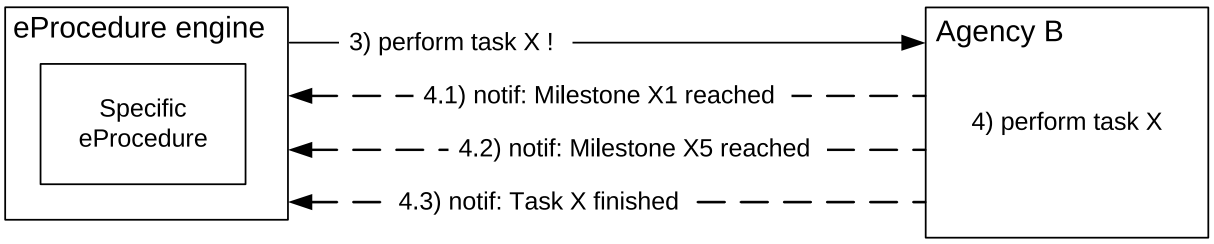
In order to actually run an eProcedure, the eProcedure engine, which is at the heart of our centralized control flow scheme, must keeping track of which calls to which services of which agencies to perform, must actually perform these calls, and must handle the return of these services' calls. As obvious as is may seem, this implies that the eProcedure engine *has to know* where in a running eProcedure we currently are, and which services we are currently waiting for. This information is very important, because it can serve as the basis for eProcedure *monitoring*, which consists precisely in giving information to a user about which step the eProcedure has reached, and which agencies we are currently waiting for.
Both the explicit description of an eProcedure, and the ability to monitor its current advancement state bring very valuable *visibility* and *transparency* to the entire system. This is especially important for the citizen at the origin of the procedure (such information will mostly be relayed to the citizen by the civil servant that helped the citizen in the first place). The citizen can obtain information about the advancement of the procedure relevant to him, can know whether it is stalled or not, can "see" it advance. Such transparency is very important in bringing confidence about the entire system.

Most steps of an eProcedure consist in a call to a Web Service, within an agency, and from the eProcedure engine's point of view, these calls are considered *atomic*. This means that the eProcedure does not care whether the service called is itself implemented as a single or as many steps. The engine only knows the service interface, and calls it as a single step.

[2] Note that here, "returning control" means "send an asynchronous message, informing that the processing is finished".

This implies that from a monitoring point of view, the eProcedure engine cannot give *finer-grained* information about the state of advancement of an eProcedure. However, such finer-grained information could be very useful, for several reasons :

- in Terregov, eProcedures are themselves are exposed as Web Services (by the eProcedure engine). This means that a step in an eProcedure can be another entire eProcedure, potentially very complex and involving many agencies, or even involving others nested eProcedures. The steps inside this nested eProcedure could be easily monitored as well, provided that the eProcedure engine is able to detect this nesting and take it into account.
- when an agency's service takes a very long time to answer, or stalls, it is very interesting to know if why, and if there is something that can be done about it. Even without detailing all the internal steps necessary to the agency's service completion, some *major milestone* steps could be described by the agency service itself. For instance, the service could inform that the request has been taken into account, or that it is waiting for someone to handle it, or that it is waiting for some other agency to complete some task, etc. Such information can be consolidated with the monitoring information that the eProcedure engine is able to determine itself, and give to the citizen a rather precise information about the advancement of his request. Even though this seems contradictory with the atomicity of a Web Service call, this makes sense in the context of monitoring. Of course, nothing prevents an agency to implement their service in a way that they remain absolutely "*opaque*" to the caller, not reporting any internal milestone completion.


eProcedure engine
Specific eProcedure
3) perform task X !
4.1) notif: Milestone X1 reached
4.2) notif: Milestone X5 reached
4.3) notif: Task X finished
Agency B
4) perform task X

3.1.2 Ownership

One issue of high importance in the field of eGovernment is procedure *ownership* and *responsibility*. At any time, the person at the origin of an eProcedure (a civil servant, on behalf of a citizen) should be able to know what is the status of the eProcedure, in other words, who is responsible for its next step(s).

The eProcedure engine is responsible for performing the calls to the agencies, synchronizing these calls, and keeping track of the global completion of the eProcedure. However, the eProcedure engine cannot be held *responsible* for each step's completion : the *agency* answering each call is responsible for it. In other words, the eProcedure engine knows who is responsible (and keeps track of it), but each agency is responsible for each step's completion. At the time an agency is responsible for a step's completion, it can be considered as the *owner* of the eProcedure (even though at the higher level, the eProcedure itself is running on an engine which is not located in the agency)

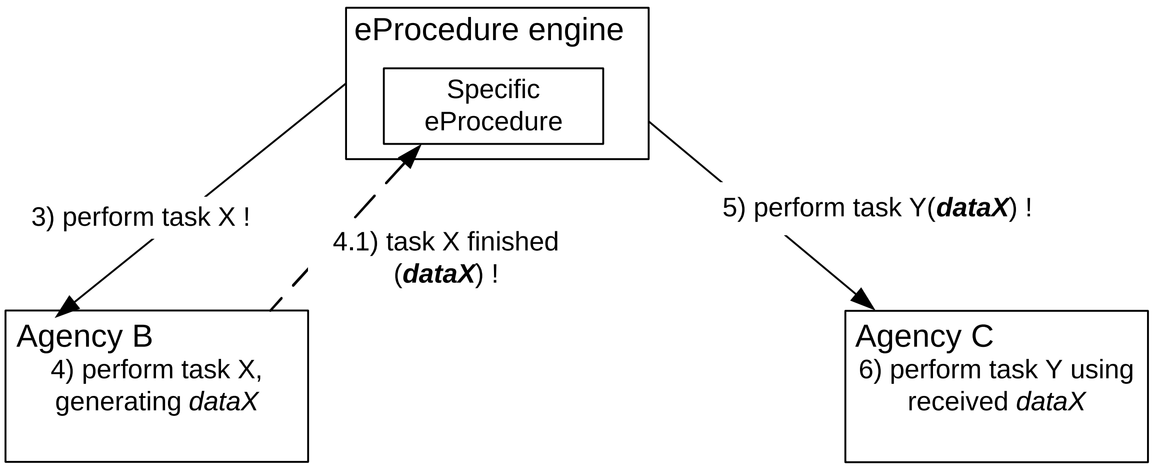
In the area of eGovernment, the notion of ownership should be made explicit. This enforces responsibility, and by using the monitoring information the eProcedure engine exposes, a civil servant or a citizen is able to know explicitly who is currently responsible for the case.

The notion of ownership we describe here is easy to understand in the case where one and only one call to an agency service is currently pending : *this* agency is the owner of the eProcedure. However, in the case of an eProcedure making “parallel” calls to several agencies, ownership is less clear : can we have *several* owners for a single eProcedure ? This issues needs to be explored further. The case of nested eProcedures deserves careful attention as well : can we have *nested* owners for a single eProcedure ?

4 The impossibility of centralized data flow

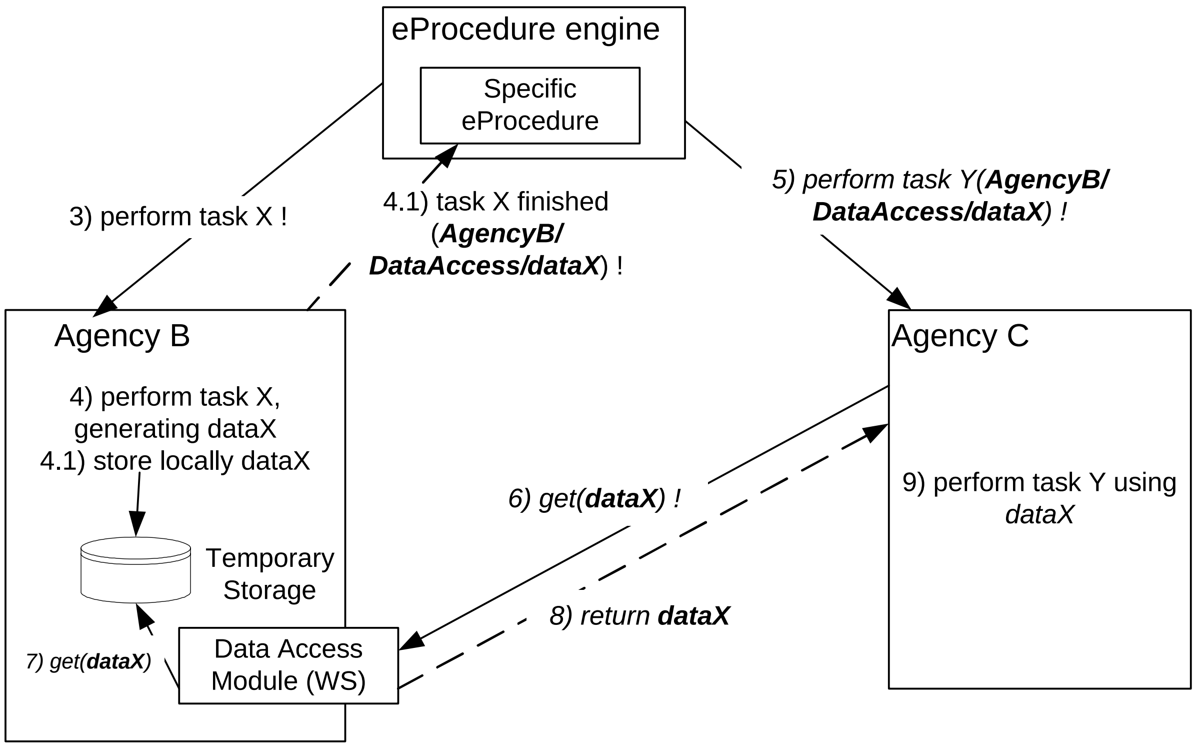
The previous discussion introduced the importance of running the eProcedures in a centralized fashion, and introduced the eProcedure engine, the component responsible for running these procedures. However, the very existence of such a centralized component, as useful as it is from a flow control point of view, raises a serious issue in the area of eGovernment, regarding *data* flow : the flow of data, especially citizen private data, is very constrained. In particular, for privacy reasons, for security reasons, and sometimes for legal reasons, data cannot be allowed to flow through *any* third party agency or organization. In Terregov, the eProcedure engine *must* be considered third party : even though Terregov does not impose a physical location for the eProcedure engine, it is a component that has a given physical location, and we must consider that in some cases (or even most cases), it will resides outside agencies that must exchange data. This implies that *data cannot flow through the eProcedure engine.*

Coming back to our example, let’s assume that Agency C needs some data generated by Agency B. In the decentralized current “manual” situation, Agency B simply sends the data to Agency C, by mail or by fax. However, with the centrally controlled eProcedure approach, Agency B does not contact directly Agency C : Agency B returns control to the eProcedure engine, which then calls Agency C. However, Agency B simply *cannot* give its data to the eProcedure engine, relying on the latter to forward them to Agency C. This is simply forbidden :


eProcedure engine
Specific eProcedure
3) perform task X !
4.1) task X finished (***dataX***) !
5) perform task Y(***dataX***) !
Agency B
4) perform task X, generating *dataX*
Agency C
6) perform task Y using received *dataX*

FORBIDDEN SCENARIO : dataX is not allowed to go through the eProcedure engine.

Since data is *only* allowed to flow *directly* from an agency to another one, we must provide a support for this direct flow, even though we want to maintain a centralized eProcedure engine. Here is Terregov’s solution to this issue : once Agency B is finished with its processing of the request, and before returning control to the eProcedure engine, it *stores locally* the data it would have otherwise sent back. Later, when Agency C is given control for performing its task, Agency C requests the data from Agency B and obtains it *directly*. In order for Agency C to be able to request and obtain the data it needs, Agency B gives to the eProcedure engine a “pointer” to a specific Web Service – exposed by Agency B – which Agency C can call to actually get the data[3]. This Web Service, specific to Terregov, is called the *Data Access Module* :



This way, the control flow remains centralized, but actual data flow is directly from agency to agency (step 8). In other words, instead of having data flowing through the eProcedure engine, only a reference to this data is passed, effectively maintaining privacy. The security and privacy is enforced by the fact that when it receives a request to from Agency C to get *dataX*, the Data Access Module in Agency B checks the identity of Agency C (through a PKI-based electronic signature), and, based on a locally defined policy, it can filter out from the data it returns all the data Agency C is not allowed to obtain.

An interesting possibility offered by this two-step data passing mechanism is the possibility of performing “lazy data processing” : in some cases, instead of actually performing task X at step 4, Agency B could simply return control immediately to the eProcedure engine, and delay the performance of task X until Agency C (or any other agency) actually calls to get the data. Of course, the “pointer” to the Data Access Module given back by agency B to the eProcedure engine would be slightly different from the first case, and it would indeed mean something different : instead of returning a reference to data, Agency B would return a reference to a *promise* of data. However, the caller could not detect the difference. With this lazy processing scheme, the processing in Agency B would happen *only* in the case where it is *actually* required. This could save some processing time, in the case where we can’t know in advance this processing will be needed or not; this brings a simple and direct support to *on-demand* processing. The important point to keep in mind here is that the eProcedure needs not

[3] Technically, the end point of the service in Agency B is returned, along with the parameter values that allow to find the correct specific data.

to be changed, as the order of the steps in the eProcedure is the same in both bases : request Agency B to perform task X, and *then* request Agency C to perform task Y. The fact that task X is actually performed in time *after* task Y is a function of the definition of task X itself as an on-demand task, not of the eProcedure description. Such a flexibility could not be supported with such simplicity and elegance by a distributed eProcedure control flow.

5 Conclusion

The separation between a centralized control flow and the peer-to-peer data flow implemented in the Terregov project brings the benefits of a high control of the eProcedure advancements and their monitoring, while providing support for maintaining privacy, and permitting fine-grained security checks at the agency-level. The apparent added complexity in the number and details of the interactions between agencies is largely compensated by the ease of creation and deployment of eProcedures, by the new possibilities of easily implementing on-demand processing within the agencies, and by the transparency the centralized control flow allows to brings to the citizen.