Process Support for Tools Interoperability

Anh-Tuyet LE
Laboratoire Logiciels, Systèmes Réseaux - IMAG
220 Rue de la Chimie, Domaine Universitaire - B.P. 53
38041 Grenoble Cedex 9
Anh-Tuyet. Le@imag.fr

Abstract

Commercial tools are heavily used, relatively cheap, well maintained, and provide powerful functionalities. However, composing these tools in order to build larger applications raises a lot of difficulties not found in component based system. Our work seeks at building a platform that makes entities of various types (component, COTS, tools, etc.) interoperate in order to build a new application. We call this new kind of application a federation.

Our federations use workflow as a support for applications integration and interoperability. In this approach, the process is not defined in term of tools and their parameters; instead, the process is high level and describes only abstract steps without knowledge on how these steps will be carried out. Therefore, the federation offers a mean to describe and control the synchronization between the abstract and executable process, and a set of concrete tools. The federation ensures that the execution of the abstract level involves a compatible real execution at the concrete level. Moreover, the real execution requires the collaboration of several tools. The description on how the abstract level is refined into the real execution satisfies consistency rules and interoperability paradigms.

We think our work contributes by providing a high level view in which the application can be described, independently from the real tools specificities, and by providing the means to describe the application behavior and the tools can be used and modified flexibly and dynamically.

***Keywords**: Federation, tools, interoperability, Software Engineering Environment, component, Process support*

1 Introduction

Commercial tools (called simply **tool** in the following) are heavily used, relatively cheap, well maintained and provide powerful functionalities. However, composing these tools in order to build larger applications raises a lot of difficulties not found in component based system. Our work seeks at building a platform that makes entities of various types (component, COTS, tools, etc.) interoperate in order to build a new application. We call this new kind of application a federation.

In this context, three major issues needs to be overcame:

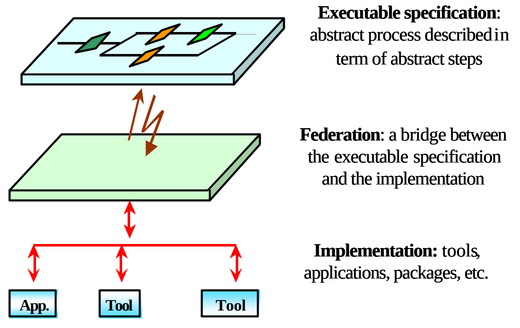
1. Overlapping of tools functionality and knowledge.
2. Mismatch between the concepts definition and imple mentation of the different tools
3. Non-determinism apparent behavior of tools and the federation.

Indeed, tools have not been designed to be used in a specific federation; it is common they provide similar functionality, and thus propose similar but not identical concepts. Furthermore, they provide user interface that allows the user to interact freely giving the impression that tool acts in a non-deterministic way [4].

Recently the idea to use a workflow as a support for applications integration and interoperability became more common [3][8]. In fact, the main objective of workflows is to automate the information flow in a working group composed by a number of tools and persons. The workflow drives the behavior of the integrated tools [9].

Using workflow as a support for integration is developed by the last generation of EAI tools (Enterprise Application Integration) [2][3]. In their vision, an elementary activity corresponds to the execution of an application; the workflow is then charged to connect these activities and to transfer information (files generally) between the participating applications. Let us note that the EAI objective is to integrate applications, i.e., to offer a means to exchange information between applications. The workflow is thus used as a means to express and control the informa tion flows between participating applications. This approach, called data-centric integration, is inflexible and complex. Changing the scenario of integration or one participating application is expensive because the process is defined directly in term of the applications and their parameters.

Contrary to EAI, our federation approach is a top-down approach. It means that the process is not defined in term of tools and their parameters. Indeed, the process is abstract and describes only abstract steps without knowledge on how these steps will be carried out[1][4].


Executable specification: abstract process described in term of abstract steps
Federation: a bridge between the executable specification and the implementation
Implementation: tools, applications, packages, etc.
App.
Tool
Tool

Therefore, the federation should offer a means to describe and control the synchronization between the abstract and executable process, and the set of concrete tools. Thus, from the federation designer point of view, the abstract process can be seen as an executable specification. The set of concrete tools can be considered as an implementation of this specification.

2 Architecture of federation

This part uses the following example in order to make easier the presentation about architecture of federation.

We want to construct a federation of document management which is a complete environment aimed to manage the life cycle of complex documents, i.e. documents which may contain a set of directories and files. Once a document is created; it will be then edited, and modified simultaneously by some developers. After modifications, it will be validated to be communicated with external persons. However, at any time, a validated document can be subject of modifications to create a new version.

To create this federation, we need a lot of components: CVS tool (Concurrent Version Control) to manage multiple physic versions of documents; a Workspace Manager tool to manage workspace of developers, i.e. to let developers to select their workspace, to make available documents that they need, etc.; a Product Manager to manage logic document which contains some properties of document; APEL which is a process tool, to define and manage the steps that developers work on a document [1][5] and some agenda tools by which the developers does a unit of work in the federation.

2.1 Common Universe

Let us note that when tools pertain to one federation, they should all recognize roughly the same concepts (the document, etc.) and they are likely to share a large body of knowledge.

The solution we propose consists in defining the main model of the application, the Common Universe (CU), which abstracts and subsumes the most important concepts used by participating tools. At execution time, the CU contains entities, instances of concepts of this model. Each tool can have a partial view of this model which may overlap with some other.

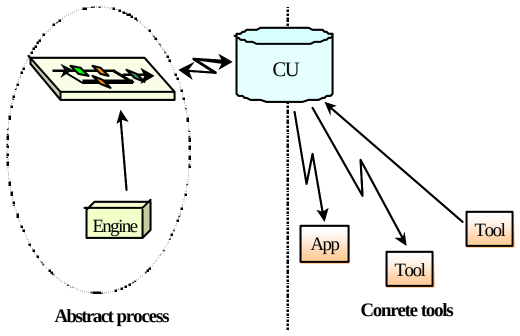
In general, the information in the CU represents the basic semantic of the application. This semantic should be supplemented and extended by semantics used by participating tools, through their information and operations. Thus, the federation must ensure the synchronization between the basic semantic and the extended semantic realized by tools in order to give the complete semantic of the application.

The simplest strategy is based on the concept of notifications. Each tool asks to be notified when a piece of information it (also) handles is changed in the CU. Conversely, when a common piece of information is changed in a tool (remember that most tools are interactive), the CU changes accordingly. In this case, the CU is main way tools synchronize. It provides a place where the common information is stored in a format, which allows each tool to synchronize (both ways) its local knowledge with the common one.

For example, suppose a developer creates a document by help of his agenda. Then, a product object is created in the CU. All others tools will be notified and will prepare for the work of developer. The Workspace Manager will ask the developer for his prefer workspace, the Product Manager will let him specify some property on his product like the description, who will be validator for the document, what is deadline for finish the document, etc. Then, the CVS tool will create a new physic document in its repository and send a copy of the document to the workspace of the developer created by Workspace Manager.

The CU is the central part of a federation which allows tools to synchronize. The evolution of CU represents the application behaviour. So, in our system, a process model will be used to describe the application behaviour in term of the CU evolution.

There are two possible places for the process tool. The first solution consists in regarding the process tool as other tools. Thus, the process model is considered as the local information of the process tool. The synchronization between the process, which is enacted be the process tool, and the CU make the CU evolutes, which then guides the set of concrete tools to coordinate in order to achieve the goal of the application. In this case, there is a large amount of information in the CU related to the process. The figure bellow shows the case where the process tool is used like other tools.



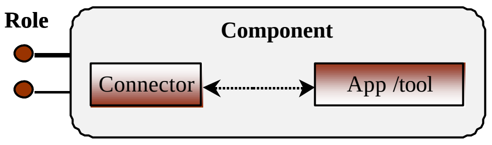
The second solution consists in putting the process directly in the CU. The others tools synchronize around the CU, which is enacted by the process tool. This solution is more reasonable than the first solution because it avoids the loss of performance to synchronize between the process tool, which is the central tool of the applications guided by a process model, and the CU.

In the document management federation, these two solutions have been tested. In this particular case, the second solution seem better because it did eliminate many duplication and thus many consistency issues. On the other hand, in the first solution any process tool can be used, while in the second solution the process environment is a federation; the CU contains the process model and instances; various tools likes agenda, process engine, monitoring are seen as the tools making up this federation. The federation of application (i.e. document Manager) is one extension of the process federation.

2.2 Managing heterogeneity

In a federation, the tools can use similar concepts. However, these concepts can be represented in a different way according to each tool and differently with those in the CU. The adaptation between the common concepts, abstracted into the CU, and those in the concrete tools must be solved.

The adaptation uses the concepts of adapter or connector. Thus, each tool uses an adapter for the semantic and syntactic translation between the vocabulary, including the concepts, the methods, and the events, of the CU and these that it uses. We call the whole of the tool and its adapter a component. Each component offers services to a federation through roles that it implements.



A federation sees tools or applications only through their roles. All controls towards a tool, which will be presented in the following part, will be pressed on its roles. The role thus represents an abstract component while the whole of the tool and its adapter is the concrete one. In execution, there are only concrete components, which satisfy all the properties of the corresponding abstract components. Separation between concrete and abstract components helps us to use tools offering the same services.

2.3 Control in federations

The interoperability strategy by notifications above may work fine but only in specific cases. It lacks any kind of control; tools are free to react and change anything in the CU at anytime. Therefore, there is nothing to ensure that the additional semantics realized by tools correspond with the basic semantic in the CU. Moreover, in certain cases, this additional semantics should be the result of the coordination, with transactional or temporal constraints, of some tools.

We add a layer of control. This layer uses the Control Common Universe (CCU), which stores information concerning the rights of the tools and coordination between them. We distinguish three parts of control: observation control, initiative control, and coordination control.

Observation control

This part of control prevents the components to subscribe to the not-allowed notifications. Thus, a component can receive only allowed notifications from the CU.

Initiative control

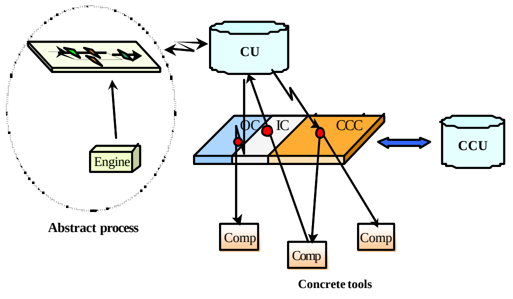
This part of control allows only components having the right to change the CU. Indeed, it captures all the requests of change on the CU to check if this request is legal or not. The request will be rejected if the request is not legal.

Coordination and consistence control (CCC)

Each method of an object in the CU is potentially linked with a protocol which extend its semantics by the semantics of concrete tools. Thus, this part controls explicitly the extension protocol of each CU method.

A extension protocol consists in set of *aspects*. Each aspect is in charge of defining an extended semantics, realized by some concrete tool, to the correspondent operation of the CU that it extends.

The CCC receives the CU change requests, computes the extension protocol involved (if any), defines the consistency required (transaction control, temporal constraints, and so on); and then executes the extension protocol, enforcing the required consistency constraints. To do so, the CCC interprets a "coordination model" stored in a CCU. The CCC has also the knowledge of the components forming the federated application (a component model), rights and duties of each component and enforces the coordination.



It is interesting to note we use actually AOP technique (Aspect Oriented Programming) to insert, modify and remove dynamically aspect in our system. More interesting, we can use a coordination language to implement each aspect. Our current implementation uses a very simple language within which the body of aspect is a method that calls the different components.

3 Experimentation

Our example of Document Management federation is now in phase of industrial exploitation.

The federation is fully distributed; we use RMI or Socket as the communication protocol between wrappers and proxy. The foundation also contains a "launcher" which, based on the component model stored in the CCU, is in charge of instantiating and initialising automatically and remotely tools, wrappers and proxies, subscribing a new tool to what it is allowed to observe (this is how observation control is performed). A component is launched either on user request or on predefined conditions (often a notification or an aspect request), as indicated in the component model for that tool.

The special effort has been done for the complete environment support for definition, development and administration of federations. We have defined an environment with graphical interfaces allowing designing quickly and easily a federation. The federation engine will execute the package generated by the conception and deployment tool. It is the charge to ensure that the federation will be executed as defined.

Actually, the federation handles different classes of tools: local tools, i.e. the unique instance of tool execute in the server of federation; remote tools that can be instantiated a number of times and on different machines like Workspace Manager, tools that work in a client server mode (e.g. CVS), ad hoc tools (i.e. developed on purpose like Product Manager) and so on.

4 Conclusions

The majority of the research groups tend to use process in order to construct co-operative systems with automatic guidance. With the federation architecture, the process becomes much more important in integrated systems.

We think our work contributes by providing a high level view where the application can be described independently from the real tools specificities, and where the application behavior and tools can be inserted and modified flexibly and dynamically.

Once the objectives are defined at high level, the process one, the work consists in implementing that new application by using the existing component or existing tools. The CU containing concepts utilized by these tools will be defined (in an abstract representation or not). The adaptor of each tool will be created if necessary.

This decoupling allows us to consider a federation at different levels of abstraction; which simplifies vastly the definition control and change of the application behaviours and properties; and which gives a large flexibility in the choice of the tools to use and the way to use them.

The federation technology gives also a solution to the objectives of the EAI, while proposing a process guidance of the heterogeneous tools constituting an enterprise application. However, contrary to the EAI, the process is not a workflow expressed in term of tool and their parameters.

5 References

[1] J. Estublier, P.Y. Cunin, N. Belkhatir, *Architectures for Process Support System Interoperability*. In Proceedings of the 5th International Conference on The Software Process, The International Software Process Association Press, Lisle, IL, June 1998.

[2] Jeffrey C. Lutz. *EAI Architecture Patterns*. EAI Journal, March 2000, P64-73.

[3] John E. Mann. *Workflow and EAI*. EAI Journal, September/October 1999, P49-53.

[8] David S. Linthicum, "*Process Automation and EAI*", EAI journal, March 2000, P12-18.

[4] J. Estublier, and T. Le Anh. "*Early experience in Modelling and Managing Software Federations*". ICSSEA Paris-France 2001.

[5] J. Estublier, S. Dami, and M. Amiour. "*APEL: A graphical yet Executable Formalism for Process Modelling*". Automated Software Engineering, ASE journal. Vol. 5, Issue 1, 1998.

[7] Jeffrey T.Pollock. "*The Big issue : Interoperability vs. integration*". EAI Journal, Octobre 2001, P48-52.

[8] WFDL, "*WDPL*". WFMC TC-0020, Workflow Management Coalition, 1996.