ABSTRACT
Building large software applications from Commercial Off The Shelf tools (COTS) is not an industrial reality so far. This work present a new approach to solve the different problems found when building a federation of COTS.

Keywords
Interoperability, federation, COTS. Architecture,

1. INTRODUCTION
Despite its promises, building large software applications from Commercial Off The Shelf tools (COTS) is not an industrial reality so far. It is because building large applications using COTS raises a number of hard issues : COTS qualification, adaptation, testing, vendor/customer relationships, new development process, cost and difficulty of integration and so on. Our work does not address all these issues; it focusses on the different aspects of federation support: the new methods, tools and paradigms required for actually building an application based on previously selected COTS.

In contrast to «classic» components, COTS source code is not available, they do not require the presence of other COTS, they often directly interact with users and so on; which raises major difficulties not found in “classic” component based approaches.

Overlapping of tools functionality and knowledge.
Non-deterministic apparent behavior of tools.
Mismatch between the concepts definition and implementation of the different tools.

We claim that this explains why, to a large extend, current support for COTS integration is inadequate: integration of new components is too hard to manage; evolution support requires too much effort and investment and is domain or application specific.

The system we present proposes original concepts, techniques and tools allowing applications to be built assembling many components of all nature, with strong and sometimes conflicting requirements about flexibility, evolution and control.

2. MANAGING THE OVERLAP
When the federated tools pertain to the same application domain, they all recognize roughly the same concepts and they are likely to share a large body of knowledge; the consistency of shared knowledges is a critical issue.

The solution we propose consists in defining a set of high level common concepts that abstract and subsume most of the individual tools concepts. We define the Common Universe (CU) as the set of entities, instances of the common (abstract) concepts, shared by the federation tools. Each tool has a specific view of these concepts which may overlap with some (or all) others. Tools actions and reactions corresponding to an abstract CU concept constitute the “real” concept i.e. what is really done (outside the CU) by the tools during execution.

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.

The CU is the main way tools synchronize. It provides a place were the common information is stored in a format which allows each tool to synchronize (both ways) its local knowledge with the common one. Notifications are not only used to synchronize the tool local information with respect to the global one (in the CU); they are also used to manage the global federation control. In this strategy each tool observes the CU and is free to react to changes it finds relevant, perhaps changing other parts of the CU; which in turn produce notifications and so on. Nevertheless, in this approach, tools have the complete initiative to what, when and how to resynchronize; the knowledge overlap is managed only if each tool owning a shared information agrees in the resynchronization strategy.

3. MANAGING NON DETERMINISM
Tools are deterministic, but humans, using their interactive interfaces, are not. Tools thus perform concurrently, have some initiative and behave in unpredictable ways; further, tools are free to react and change anything in the ACU at anytime. We can compare this strategy with a society (the federation) where each
individual (a tool) observes the state of the world (ACU) and, based on its own understanding of what happens, its own goals and capabilities, decides to act, thus changing the state of the world. In this strategy there is no society explicit goal, no rule, no control. This is why we call this strategy “the anarchy” [2]. But in many cases, tools reactions need to be coordinated, in order to react consistently.

**Process support : handling goals**

Process support is a domain where issues related with non determinism have been addressed. Indeed in process technology, a (process) model describes the intended evolution of the real world; a process engine executes the process model changing the values within an universe in which are represented (as objects) the entities of the real world. Process technology takes provision for handling the discrepancies between the universe state and the real world state, to handle non-deterministic world evolution and so on [3].

In our system, a process model describes the application behavior in term of the ACU evolution. This model can be seen as a formal representation of the application goal.

4. MANAGING META MODEL MISMATCH

In the previous approach, tools are abstract tools, compliant with our protocols. The issue is to make the link between the abstract tools and the real tools; and to prove that the adaptation does not break the properties defined in the abstract federation model.

We introduce the concept of **Proxy, Proxies** are in charge of translating the concepts present in the ACU from/to the concepts handled by the tools (files for example). A proxy, which runs on the foundation machine, calls remotely the tool wrapper. The **Wrapper** is in charge of calling the tool but also, more challenging, to observe the tool and to inform the federation when the activity of the tool has an impact on the federation [4].

5. CONCLUSION

The complete foundation, as sketched above, has been implemented. We use the Apel process support system for the process common universe management, for process modeling and for process execution [1]. We use the ObjectStore database for the Application and Federation CU’s. We have developed tools and graphical interfaces for federation design and development, including tools for debug and for dynamic federation evolution (new tool, migration, replication ..).

We think this work contributes in the following ways:

**High-level view.** A major contribution of this work is to provide the federation designer with a high-level federation view, ignoring the many details of the real tools. This high level view is nevertheless a representation of the reality accurate enough for reasoning, defining goals and rules, applying process technology and so on. Our framework provides the federation designer with a graphical interface for the design of federation behavior and characteristics. We think the approach has the merit to relate the high-level view with the actual tools, in a very flexible way.

**Interoperability paradigms.** We believe that the evolution from completely rigid control paradigms (i.e. the current technology), to the availability of a large pallet of interoperability paradigms is of major importance at least for tools with a high degree of autonomy and complexity, as illustrated by our society analogy[2]. It is our belief that relying (reasonably) on initiative and loosely coupled systems is promising as illustrated by the increasing introduction of the message-based technology.

**Federation architecture.** The architecture we propose decouples a federation “horizontally” and “vertically”. Horizontally, we have identified three domains: the application, process and control. We believe this separation of concern greatly helps in designing and building a federation.

Vertically, we have identified different levels of abstraction each one dealing with a precise topic and with minimum dependencies with the other layers. The top level (the CU) provides an abstract view of the application behavior; the next level links the abstract level with abstract tools, emphasizing the needed paradigm and coordination constraints. The third level (proxy and wrappers) deals with the actual tools, their interfaces, actual localization, communication constraints and so on. It is striking to see that most work deals with that last level only.

**Openness, flexibility, evolution.** This separation of concern along two axes allows changes in one area with few or no impact to the others. The same application can be implemented with different tools, using different interoperability paradigms, and with different properties. A federation can be easily modified and tailored to fit company needs. Notably, the flexibility is the explicit topic of the interoperability paradigms. It is the explicit duty of the federation control foundation to set these properties.

We believe this work is a significant step towards the realization of future Software Engineering environments.


