Knowledge in Interoperable and Evolutionary Systems

Nacer Boudjlida
CRIN - Bat. LORIA - Université Henri Poincaré Nancy 1, B.P. 239
54506 - Vandœuvre Lès Nancy Cedex (France)
E-mail: nacer@loria.fr

Databases, Logic Programming and Artificial Intelligence fields successfully cooperated in the area of deductive databases. Substantial results were gained in querying, albeit results on *updates and revision* are less impressive, especially from a computational perspective. Cross-fertilisation among the fields seems also very promising in the domain of dynamic and reactive systems that behavelike systems that supervise on-going activities: they must *execute* actions, *reason* about these, *gather* information about ongoing activities, *predict* possible illfunctioning, *control and coordinate* the activities, etc. New database application domains, like databases for CAD/CAM or software engineering, require this kind of functionalities. In particular, Software Engineering Environments (SEE) that support Software Process Models (SPM) falls in this category of systems: these are also called Process Centred SEE (PCSEE). SEEs usually concentrate on the support for software products development. PCSEEs provide additional supports for the activities and the agents that are implicated in to the development and the management of software projects. In this framework, SPMs is an abstract specification of how the software related activities *should* be carried out. The specification at least encompasses descriptions of the object types that are produced by the activities together with descriptions of the activities themselves and policies to be obeyed to. A PCSEE includes a *knowledge* base that contains SPMs, an object base that contains SPMs instances and software products. The PCSEE's *Software Process Engine* interprets (*enacts*) a SPM to drive the development of a software project in conformance with an *instantiated SPM*. The *Process Engine* is a set of mechanisms that controls the ongoing activities and provides a set of assistance facilities like predicting future states of the objects, explaining how a given state has been reached or can be reached, and so on. The mechanisms that constitute the *Process Engine* share and inter-operate on the knowledge base that contains the specifications of the SPMs and the object base that contains the products being developed and the gathered information about the activities that have been performed.

The environment is viewed as a collection of *tools* that cooperate in the support of the activities, that communicate and exchange objects, messages and events, and inter-operate on the ob jects in the environment's bases. Considering the variety of tools in a PCSEE, their ability to *inter-operate* on a same set of objects is crucial for the evolution of the environment. Inter-operability may be achieved through a common representation of the knowledge and the ob ject bases or through specific mechanisms that restructure objects, i.e that adapt their representation to the in ter-operable tools. Syntactic-based approaches to object interchange for inter-operability, like those based on an Interface Definition Language, must be extended by knowledge on the objects contents. We experimented a knowledge-based implementation of object restructuring and we are currently investigating the potential mutual contributions of the works on data interchange (like Common Data Interchange Format) and knowledge interchange (like KIF and KQML)to incorporate more knowledge in to object descriptions and to exploit it in the object restructuring process (this process can be viewed as a dynamic knowledge-based mechanism to achieve *ad hoc polymorphism with coercion*).

Objects in PCSEEs are no more "classical" database objects as are "flat" relations in relational databases. Objects, like design documents or source code, are complex objects with possible nesting (*is_part_of relationship*) and specialized/generalized objects (*is_a relationship*). Moreover, the associated Data Base Management System, called Object Management System, must be extendible with new object types. It must also support cooperative work, active rules, long- term activities and object versioning. Indeed, experimental activities like software engineering often require going back to previous steps or previous states of objects: versioning is then "a must" as it is to enable various evolutions in PCSEEs. *Evolution* can take place at different levels: the environment's hosting platform, the SPM level, as well as the SPMs' instances and the object base levels. Existing knowledge and objects *must be adapted* to the changes, i.e. multiple versions of the knowledge and the object bases may be maintained, every version corresponding to a version of the knowledge base and the object base specification, or alternatively, the existing knowledge base and object base may *migrate* to meet their respective new specifications. This appeals for mechanisms to *manage* knowledge and object schema *evolution and versioning*, mechanisms to *re-use* existing SPMs and objects, and means to *analyse the impacts of* *a change*, etc. Change impact-analysis and change side-effects propagation meet the *frame* and the *ramification problems* in knowledge bases revision.

In this position paper, we argue that management and reasoning on structurally complex objects in the framework of dynamic systems, like PCSEEs, require knowledge concerning the knowledge itself, the objects and the actions that may be performed on the objects. It also requires a kind of "*reflexivity*" to enact (i.e. execute) the knowledge provided by the Process Models and to manipulate it, notably to ensure its evolution. Reflexivity is the fact that Software Process Models are considered as objects: so they can be updated and revised as any other object. At any level it occurs, evolution requires impact analysis similar to the resolution of the frame and ramification problems. Further, similarly to multi-agents systems like blackboard systems, interoperability of the tools must be ensured not only to enable them cooperate in carrying out the activities, but also to adapt existing knowledge and objects to possible evolutions. This feature favours knowledge and object re-use and must be founded on objects' structure and content.