Software Asset Classification and Retrieval with Description Logics: the Intelligent Reuse Assistant

Enrico Franconi
Knowledge Representation and Reasoning group
IRST, I-38050 Povo TN, Italy
franconi@irst.itc.it

1 The Intelligent Reuse Assistant project

Reuse of software artifacts at all phases of the software engineering life-cycle is recognized as being one of the major enablers for productivity and quality improvements. However, a common inhibitor to company-wide reuse is the lack of visibility of reusable assets among the developers community. Repository systems partly address this problem by providing central mechanisms for classification and retrieval of software assets. However, these systems usually provide relatively straightforward mechanisms for classification and retrieval (e.g. faceted scheme). Moreover, retrieval of assets is driven by the interest of a software developer. This interest may be biased by pre-conceived ideas about the existence of reusable assets while ignoring the existence of others. This is a limitation of all focused searched. Instead, an approach in which reuse opportunities are suggested by the system (rather than by the user) would guarantee a wider coverage of the reuse repository and thus increase the number of reuse opportunities.

The long-term objective of the project we are presenting here is to conceive and study a tool for the storage and retrieval of software assets for reuse purposes. This tool will permit a semi-automated extraction of an asset representation from a series of artifacts adopting de-facto, object-oriented technology standards (C++ for code artifacts and OMT for models). Intelligent access capabilities will also be implemented (e.g. reasoning mechanisms, pattern-matching) through a user-friendly man-machine interface adapted for software developers. These functionalities will allow an organization to maximize its reuse opportunities by significantly reducing the *invisibility* of assets that are present in the repository. From a software engineering perspective, the selection of the object-oriented paradigm for asset representation, and of OMT and C++ in particular, will facilitate further exploitation in a broad industrial context having adopted these mature object-oriented technologies. From a technological perspective, innovation will come from the implementation of the concept of *Intelligent Reuse Assistant*.

Innovative techniques which should be carefully studied for the realization of the Intelligent Reuse Assistant include:

- Semi-automatical tools for homogeneous translations into a uniform representation – using a knowledge representation language of the family of *Description Logics* – from the OMT object-oriented notation and from the formal specification language which are used for the asset representation from the structured information about software artifacts – such as requirements and architecture of software assets and of its code specification.
- Use of a uniform language based on Description Logics for the representation of all artifacts that compose a software asset.
- Sophisticated querying capabilities of two types:
  - Reactive (or declarative) queries to fully exploit the rich reasoning capabilities of the language to retrieve software assets matching reusers' requests about assets.

  - Pro-active (or comparative) queries to suggest reuse opportunities to potential reusers by doing a sort of pattern-matching between (possibly incomplete) artifacts of assets under implementation and assets present in the reuse repository.

- Facilities for knowledge base maintenance (interfaces to assist the semi-automated extraction process, to update the schema, and to create the ontology).

Description logics can be considered as an unifying formalism, since they allow the logical reconstruction and the extension of representational tools such as frames, object-oriented data models, semantic data models, semantic networks, and formal specification languages. Formal specification languages were chosen just because they are suited to provide a *bridge* between informal requirement specifications and information expressed in Description Logics. A fundamental requirement for the formal language will be the feasibility to provide an automatic translation into Description Logics. For this reason, the formal language considered might be a subset of existing formal languages.

Thus, description logics play the role of uniform logical language for the conceptual modeling of software assets, in order to allow classification and flexible access.

The system uses description logics for the conceptual modeling of software assets, and for data aggregation and abstraction in the various dimensions and at different levels of granularity, with a particular emphasis to the notion of *dynamic* information. The *classification* of description logics based asset descriptions is completely automatic and it is based on complete inference engines. A description considers non only simple facets of an asset but also properties and their interrelations in a very expressive language, making classification much more effective. The terms used in a description refer to a large *ontology*, specifically developed for this project, which is focused on the general topic of software engineering and on the particular domain of telecommunications. In this way,also the work of a knowledge engineer to maintain and update the knowledge base is greatly simplified.

Our proposal considers description logics as a tool for representing not only the model but also the queries as well. Traditional information access tools and technologies are not adequate for the described application, since they are tailored for the optimization of search in huge archives large amounts of data, where the information lacks of structure in its description – such as in the Internet. On the contrary, description logics capture the deep semantics in the descriptions of artifacts having a complex structure, where the emphasis is given not to the quantity of softer assets items, but to the fine grained relationships among the assets themselves. The presence of complex structures – possibly with hundreds of attributes – leaves out from human capacities the ability of excerpting the most of significative interrelationships among data, without the support of some interactive tool. Using description logics has several advantages in the information access phase, basically because the system is able to *reason* on the query. The interaction of a final user and of a knowledge engineer with the information system may be improved with several operations, based on intensional query processing and knowledge navigation, leading to a flexible access to data in many different modalities.

The technology adopted will build upon several projects that have addressed similar problems in the past. In particular, the ITHACA (ESPRIT) and LaSSIE (AT&T) projects will provide the ground for query mechanisms. However, both projects suffer from the proprietariness of input data, restrictions of the query language, and difficulties to efficiently maintain the knowledge base. Other relevant projects are REBOOT (ESPRIT), EuroKnowledge (ESPRIT), and Corinto (italian IBM and Apple).

2 Proposed System Architecture

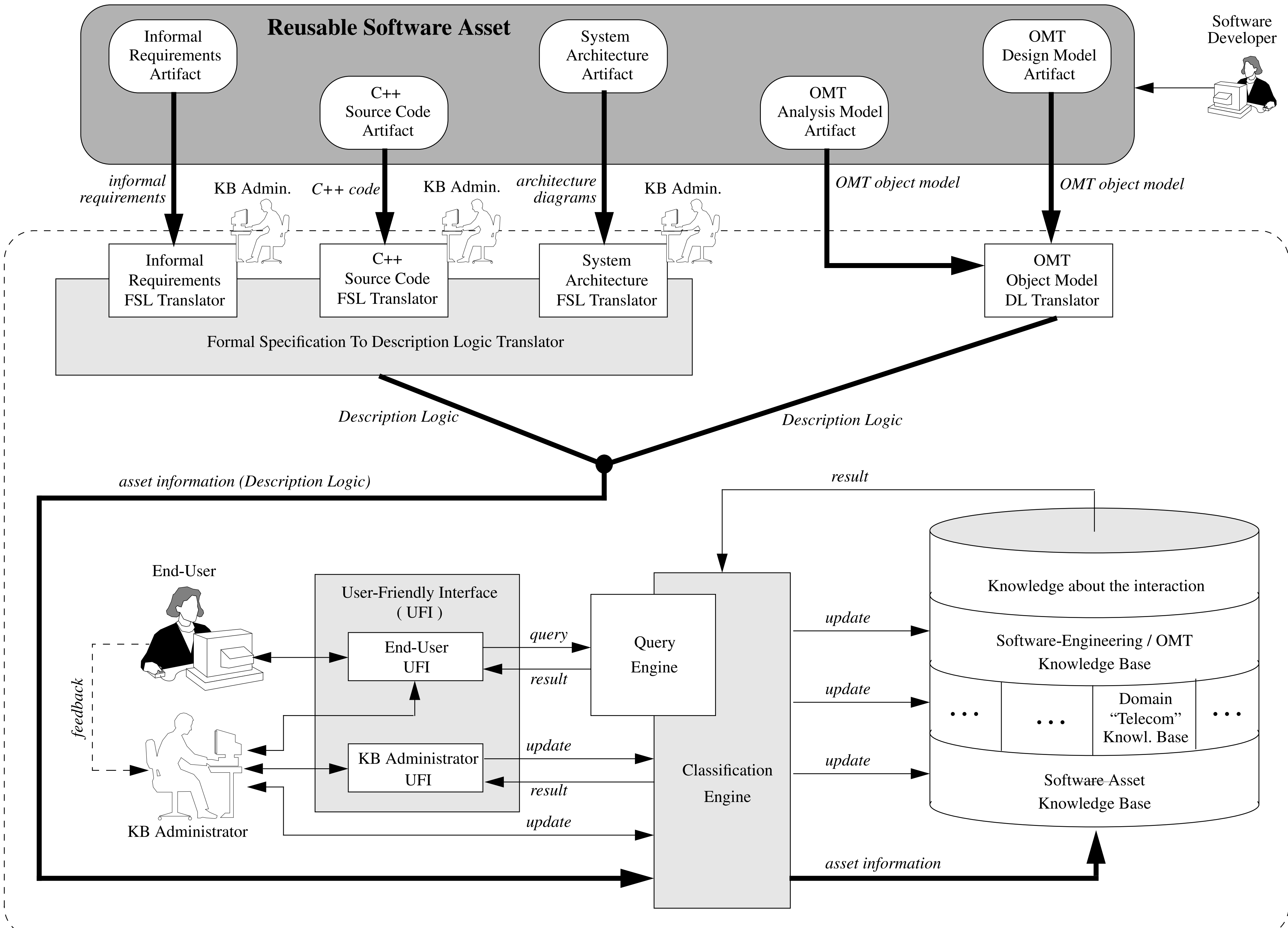
At the top of the figure, several components have the mission to extract information from the software artifacts of an asset. A single component is needed to extract information from OMT analysis and design models. Since a mapping can be defined between an object representation and a description logics representation, the output of this component is directly expressed in description logics. However, in the case of requirements, architecture, and code artifacts, the mapping to description logics requires to first go through an intermediate representation using a formal specifications language.

Once the various description logics description logics artifact representations are produced, they must be combined into a coherent representation at the asset level. This is the input to the classification engine that creates an assertion for the asset descriptor into the knowledge base.

The knowledge base is decomposed into three layers. First, the software engineering layer which includes knowledge about software engineering, such as C++ constructs, traceability concepts throughout the life-cycle, etc. The second layer is in fact composed of a series of partitions. Each partition contains knowledge about a particular domain to which an asset belongs. Finally, the bottom layer contains knowledge relative to individual assets. Strong associations exist between the various layers, and thus a software asset is in fact represented across the three layers.

The classification engine is responsible for producing the classification of a software asset, either when the asset is asserted or when a query is made. It is also responsible for providing knowledge base manipulation services. Its interface is based exclusively on description logics.

The query engine is a component exclusively dedicated to handle user queries to the knowledge base. It always interfaces to the classification engine which provides all services to directly access the knowledge base. In other words, the query engine mainly pre-processes the query issued by a user. Its interface is based exclusively on description logics.


Reusable Software Asset
Informal Requirements Artifact
C++ Source Code Artifact
System Architecture Artifact
OMT Analysis Model Artifact
OMT Design Model Artifact
Software Developer
informal requirements
KB Admin.
C++ code
KB Admin.
architecture diagrams
KB Admin.
OMT object model
OMT object model
Informal Requirements FSL Translator
C++ Source Code FSL Translator
System Architecture FSL Translator
OMT Object Model DL Translator
Formal Specification To Description Logic Translator
Description Logic
Description Logic
asset information (Description Logic)
result
End-User
User-Friendly Interface (UFI)
End-User UFI
KB Administrator UFI
query
result
update
result
update
Query Engine
Classification Engine
feedback
KB Administrator
Knowledge about the interaction
update
Software-Engineering / OMT Knowledge Base
update
. . .
. . .
Domain "Telecom" Knowl. Base
. . .
update
Software Asset Knowledge Base
asset information