Computer Standards & Interfaces 33 (2011) 519–532

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

Knowledge Discovery Metamodel-ISO/IEC 19506: A standard to modernize legacy systems

Ricardo Pérez-Castillo *, Ignacio García-Rodríguez de Guzmán, Mario Piattini

Alarcos Research Group, University of Castilla-La Mancha, P° de la Universidad, 4 13071, Ciudad Real, Spain

ARTICLE INFO

Article history:
Received 27 April 2010
Received in revised form 15 February 2011
Accepted 21 February 2011
Available online 27 February 2011

Keywords:
KDM
ISO19506
Knowledge management
Software modernization
Legacy system

ABSTRACT

Legacy systems age over time as a consequence of uncontrolled maintenance, thus they must be evolved while its valuable embedded knowledge is preserved. Software modernization, and particularly Architecture-Driven Modernization, has become the best solution in the legacy systems' evolution. ADM defines the Knowledge Discovery Metamodel specification, now being adopted as ISO/IEC 19506 by the International Standards Organization. The KDM metamodel allows to represent all the software artifacts recovered during reverse engineering techniques at different abstraction levels. This paper presents how to use KDM to modernize legacy systems, making them more agile, preserving the embedded business knowledge and reducing maintenance costs.

1. Introduction

As the history of software engineering reveals, information systems are not static entities, but changeable over time. Information systems degrade and age, and they become *legacy* information systems because the code of these systems was written long ago and now may be technologically obsolete [49]. This problem is known as the software erosion phenomenon [52]. Most often, it is due to maintenance activities carried out over time, since the successive maintenance changes in an information system minimize its quality. In this case, a new and improved system must replace the previous one. However, replacing these systems completely from scratch is very expensive, and also slows down the achievement of ROI (Return of Investment) [48]. Additionally, the software embeds a significant amount of business knowledge over time that would be lost if entirely replaced [44].

In tackling the software erosion phenomenon, evolutionary maintenance is the best solution for obtaining improved systems without discarding the existing systems, which minimizes software erosion effects. Evolutionary maintenance addresses adaptive and perfective maintenance changes [17], and makes it possible to manage controllable costs and preserve the valuable business knowledge embedded in the legacy system.

Reengineering has been the main mechanism for addressing the evolutionary maintenance of legacy systems for a considerable time [8]. Reengineering preserves the legacy knowledge of the systems and makes it possible to maintain software easily at a low cost [7]. This reengineering process consists of the examination and alteration of a legacy system to reconstitute it in a new form and the subsequent implementation of the new form [10].

Nevertheless, a 2005 study [48] states that over 50% of all reengineering projects currently fail, due to two main weaknesses: formalization and automation problems. On the one hand, reengineering processes lack formalization and standardization [21], since these processes are carried out in an *ad hoc* manner. Thus, different reengineering tools that address specific tasks in the reengineering process cannot be integrated or reused in different reengineering projects. And on the other hand, the reengineering of large complex legacy information systems is very difficult to automate [9]; since the reengineering processes cannot be repeatable, as a consequence of the formalization problem. Therefore the cost of maintenance based on reengineering grows significantly.

At this time, software modernization is a new specific kind of evolutionary maintenance paradigm to solve reengineering problems. ADM (Architecture-Driven Modernization) as defined by OMG (Object Management Group) [36], advocates carrying out the reengineering process but considering model-driven development principles. That is, ADM deals with all the involved software artifacts as models in a homogenous manner, and it facilitates the formalization of deterministic transformations between those models [33]. The model transformations can be formalized by means of QVT (Query/Views/Transformations), the model transformation language proposed by OMG [39]. The automation of the model transformations together with the model-driven development principles makes it

* Corresponding author. Tel.: +34 926295300; fax: +34 926295354.
E-mail addresses: ricardo.pdelcastillo@uclm.es (R. Pérez-Castillo), ignacio.grodriguez@uclm.es (I.G.-R. de Guzmán), mario.piattini@uclm.es (M. Piattini).

0920-5489/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.csi.2011.02.007

possible to reuse the models involved in ADM-based processes. As a consequence, the automation problem can also be solved due to the automated transformations together with the reuse of the models.

As part of the effort undertaken by the ADM Task Force of OMG, KDM (Knowledge Discovery Metamodel) is the first standard within a broad set of proposed standards [43], although many of these standards are still in the approval or development stage. KDM forms the core of that set of standards for two main reasons: (i) KDM addresses the main challenges that appear in the modernization of legacy information systems; and (ii) it is the cornerstone of the set of proposed standards, since the other standards are defined around KDM. Despite the fact that KDM had been defined by the OMG, it is being adopted as the international standard ISO/IEC 19506 ADM KDM [19] through the so-called "fast-track" liaison relationship between OMG and ISO.

KDM is a specification of agreed upon facts and how these facts are represented in XML by using OMG standards such as Model Object Facility (MOF) and XML Metadata Interchange (XMI). The availability of this information represented according to the KDM specification makes it possible (i) to store the facts about information systems in a compliant structure; (ii) to analyze and reason about KDM facts; (iii) to exchange KDM facts as XML documents; and (iv) to build tools so that parsing source code in a given programming language to generate language-independent and vendor-neutral facts about information systems.

From the modernization perspective, KDM allows addressing all the stages of modernization processes based on ADM: reverse engineering, restructuring and forward engineering [24]. In the reverse engineering stage, the software modernization process analyzes the legacy system (at a lower abstraction level) in order to identify the components of the system and their interrelationships and build one or more representations of the legacy system (at a higher abstraction level). In addition, the metamodel defined by the KDM standard provides a common repository structure that makes it possible to exchange information in the restructuring and forward engineering stages about all existing software artifacts in legacy systems like source codes, databases, user interfaces, business rules, etc. [42].

The representation of the system's knowledge throughout the software modernization stages has four key challenges that must be addressed:

- Legacy systems must be represented from different viewpoints and different abstraction levels where it must be possible to represent all the software artifacts in legacy systems [20]. In addition, the different views of the systems must be linked in order to support the traceability between elements at different views or levels.
- The knowledge embedded in legacy systems must not only be represented in a suitable and accurate way in the reverse engineering stage, but must also be managed throughout the following stages in the whole software modernization process: the restructuring and forward engineering stages.
- It must be possible for different tools to share the recovered knowledge in order to automate the different modernization tasks.
- It must be possible to discover or deduce new implicit knowledge from the explicit knowledge recovered from legacy systems.

All these challenges are met by KDM standard ISO/IEC 19506. This paper presents in detail how the KDM standard can address the challenges presented. Moreover, this paper shows the way in which KDM assists in the entire software modernization process in order to minimize the effects of software erosion on legacy systems.

The remainder of this paper is structured as follows: Section 2 presents a brief history of KDM in order to understand the role that KDM plays in the modernization of legacy systems. Section 3 summarizes works related to the knowledge representation used in software modernization and similar processes. Section 4 presents the elements and features of the metamodel defined by the KDM standard. In addition, this section presents the KDM standard from an ontological perspective and as a common interchange format. Section 5 shows how KDM can be used in ADM-based processes in order to exploit all their benefits. Finally, Section 6 summarizes and discusses the conclusions of this paper.

2. History of KDM

In June 2003, OMG formed a task force to model software artifacts in the context of legacy systems. Initially, the group was called the *Legacy Transformation Task Force*, but soon, the group changed its name to the *Architecture-Driven Modernization Task Force* (ADMTF). In July 2003, the ADMTF issued a software modernization whitepaper [34].

In November 2003, the ADMTF issued the request-for-proposal of the Knowledge Discovery Metamodel (KDM) specification. The objective of this initial metamodel was to provide a comprehensive view of the application structure and data, but it did not represent software below the procedure level. The request-for-proposal stated that the metamodel of the KDM standard:

- represents artifacts of legacy software as entities, relationships and attributes
- includes external artifacts with which software artifacts interact
- supports a variety of platforms and languages
- consists of a platform and language independent core, with extensions where needed
- defines a unified terminology for legacy software artifacts
- describes the physical and logical structure of legacy systems
- can aggregate or modify, i.e. refactor, the physical system structure
- facilitates tracing artifacts from logical structure back to physical structure
- represents behavioral artifacts down to, but not below, the procedural level.

In May 2004, six organizations responded to the request-for-proposal. However, throughout 2004 and 2005 more than 30 organizations from five different countries have collaborated in the development and review of the KDM standard. In May 2006, KDM was adopted by OMG and moved into the finalization stage in the adoption process. In March 2007, OMG presented the recommended specification of KDM 1.0. In April 2007, OMG started ongoing maintenance of the KDM specification.

In January 2009, the recommended specification of KDM 1.1 was published by OMG [42], and in addition, OMG started the revision of that version. Recently, in March 2009, ISO started the adoption process of the KDM specification, which is known as ISO/IEC ADM KDM 19506 [19].

3. Related work

Knowledge management based on models throughout all the stages of software modernization has been widely studied. Several alternative solutions were proposed prior to the KDM standard.

In most cases, the knowledge recovered through reverse engineering techniques is represented and managed in an *ad hoc* way. Zou et al. [55] propose a framework based on a set of heuristic rules for extracting business processes through the static analysis of the source code. The authors of this work also use an algebraic approach in the study of business processes. Favre [13] proposes an MDA-based framework for modernizing legacy systems. This work defines NERUS, an *ad hoc* metamodeling language, to represent metamodels and metamodel transformations. Perez-Castillo et al. [47] propose an ADM-based framework to generate web services from relational database schemas. The authors of this work use a specific metamodel according to the SQL standard to represent the knowledge about