Information Systems 34 (2009) 87–107

Contents lists available at ScienceDirect

Information Systems

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

Toward data mining engineering: A software engineering approach

Oscar Marbán *, Javier Segovia, Ernestina Menasalvas, Covadonga Fernández-Baizán

Facultad de Informática, Universidad Politécnica de Madrid (U.P.M.), Spain

ARTICLE INFO

Article history:
Received 19 February 2008
Received in revised form
22 April 2008
Accepted 22 April 2008
Recommended by: D. Shasha

Keywords:
Data mining
Software engineering
Knowledge engineering

ABSTRACT

The number, variety and complexity of projects involving data mining or knowledge discovery in databases activities have increased just lately at such a pace that aspects related to their development process need to be standardized for results to be integrated, reused and interchanged in the future. Data mining projects are quickly becoming engineering projects, and current standard processes, like CRISP-DM, need to be revisited to incorporate this engineering viewpoint. This is the central motivation of this paper that makes the point that experience gained about the software development process over almost 40 years could be reused and integrated to improve data mining processes. Consequently, this paper proposes to reuse ideas and concepts underlying the IEEE Std 1074 and ISO 12207 software engineering model processes to redefine and add to the CRISP-DM process and make it a data mining engineering standard.

1. Introduction

In its early days, software development focused on creating programming languages and algorithms that were capable of solving almost any problem type. The evolution of hardware, continuous project planning delays, low productivity, heavy maintenance expenses and failure to meet user expectations had led by 1968 to the stagnation of software development, causing what came to be known as the *software crisis*, the term coined at the first NATO conference on software development [1]. This crisis was caused by the fact that there were no formal methods and methodologies, support tools or proper development project management, all of which were standard techniques used in projects developed in other classical branches of engineering. The software community realized what the problem was and decided to borrow ideas from other fields of engineering, which it incorporated into software project development. This was the origin of software engineering (SE). As of then process models and methodologies for developing software projects began to materialize.

Software process models describe the tasks to be performed to develop a software system, whereas development methodologies schedule the tasks and specify what methods to use to do the tasks [2]. Software development improved considerably as a result of the new methodologies. This solved some of its earlier problems, and little by little software development grew to be a branch of engineering. This shift means that project management and quality assurance problems are being solved. Additionally, it is helping to increase productivity and improve software maintenance. This is one of the major problems in software development, as it can amount to up to two-thirds of costs throughout the software system's lifetime [2].

The history of knowledge discovery in databases (KDD), now known as data mining (DM), is not much different, at least so far. In the early 1990s, when the KDD processing term was first coined [3], there was a rush to develop DM algorithms that were capable of solving all a company's problems of searching for knowledge in large

* Corresponding author at: DLSIIS, Facultad de Informática, Universidad Politécnica de Madrid (U.P.M.), Campus de Montegancedo s/n, 28660 Boadilla del Monte, Madrid, Spain. Tel.: +34 913367388;
fax: +34 913367393.

E-mail addresses: omarban@fi.upm.es (O. Marbán), fsegovia@fi.upm.es (J. Segovia), emenasalvas@fi.upm.es (E. Menasalvas), cfbaizan@fi.upm.es (C. Fernández-Baizán).

0306-4379/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.is.2008.04.003

volumes of data. Apart from developing algorithms, tools (Clementine [4–6], IBM Intelligent Miner [7,8], Weka [9], DBMiner [10]) were also developed to simplify the application of DM algorithms and provide some sort of support for all the activities involved in the KDD process.

From the viewpoint of DM process models, the year 2000 marked the most important milestone, as this was when the first standard and tool-independent DM process model was published. This standard is known as CRISP-DM (*CRoss-Industry Standard Process for DM*) [11,12].

The number of applied projects in the DM area is expanding rapidly [13]. This growth is confirmed by reports by the Gartner Group [14,15] and Forrester Research [16]. The Gartner Group estimates [14] that there will be an upsurge of DM projects over the next decade (over 300%) to improve customer relationships and help companies listen to customers. Another Gartner Group report [15] claims that enterprises in the DM area grew by 4.8% from 2005 to 2006, and DM is now the area in which companies are investing most. While it is true that a lot of DM projects are being developed, neither all the project results are in use [17–19] nor do all projects end successfully [20,21]. The failure rate is actually as high as 60% [22]. Deployed by about 50% of respondents, CRISP-DM is the most commonly used methodology for developing DM projects [23–25]. However, its use is not becoming any more widespread due to rivalry with other, in-house methodologies developed by work teams, which account for another, almost 30%.

All the above goes to show that while CRISP-DM was an improvement on the earlier state of affairs, the process model is not perhaps yet mature enough to deal with the complexity of the problems it has to address. And this detracts from the effectiveness of its deployment, as it does not produce the expected results.

Are we at the same point as SE was in 1968? Certainly not, but we do not appear to be on a par yet either, DM cannot be considered a mature field as SE [26]. Table 1 compares DM's history with SE's past. Looking at the KDD process and how it has progressed, we find that there is some parallelism with the advancement of software. From this viewpoint, DM project development is at stage 4, and is defining development methodologies to be able to cope with the new project types, domains and applications that organizations have to come to terms with. SE has reached stage 5, where development processes pay special attention to organizational, management or other parallel activities not directly related to development, such as project completeness and quality assurance. CRISP-DM has not yet been sized for these tasks, as it is very much focused on pure development activities and tasks.

This paper is moved by the idea that DM problems are taking on the dimensions of an engineering problem. Therefore, the processes to be applied should include all the activities and tasks required in an engineering process, tasks that CRISP-DM might not cover. Our proposal is to enhance CRISP-DM by embedding other current standards, as suggested in [27], inspired by the work done recently in SE derived from other branches of engineering and from developer experience.

2. DM process models

There is some confusion about the terminology different authors use to refer to process and methodology. Below we describe the definitions of standard SE terminology. These terms are used with the aim of unifying criteria, because they are better established and backed by the International Organization for Standardization (ISO) or the Institute of Electrical and Electronics Engineers (IEEE).

Table 1
Parallelism between DM and SE

SE phase	DM phase	DM characteristics	SE characteristics
Phase 1 (1945–1955)	Phase 1 (…–1990)	Gathering knowledge hidden in data was a hard thing to do Statistical techniques Machine learning	Programming was a hard thing to do Use of machine and assembly language
Phase 2 (1955–1965)	Phase 2 (1990–1995)	DM and a lot of algorithms appeared. DM tools appeared All sorts of things could be done	A host of languages appeared All sorts of things could be done
Phase 3 (1965–1970)	Phase 3 (1995–1999)	DM projects went unfinished Errors, continuous changes, unpredictable costs Nothing could be done DM environments	Program development went unfinished Inefficiency, errors, unpredictable cost Nothing could be done
Phase 4 (1970–1980)	Phase 4 (1999–…)	Process models: CRISP-DM DM methodologies: SEMMA, 5A's	Programming fundaments Design methodologies Program verification
Phase 5 (1980–…)	Phase 5 (?–?)	Unknown	Programming environments Formal specification Automated programming Software quality Human resources management