

TOWARDS THE IMPLEMENTATION OF TEMPORAL-BASED SOFTWARE VERSION MANAGEMENT AT UNIVERSITI DARUL IMAN MALAYSIA

M Nordin A Rahman, Azrul Amri Jamal and W Dagang W Ali

Faculty of Informatics

Universiti Darul Iman Malaysia, KUSZA Campus

21300 K Terengganu, Malaysia

mohdnabd@udm.edu.my, azrulamri@udm.edu.my, wan@udm.edu.my

ABSTRACT

Integrated software is very important for the university to manage day-to-day operations. This integrated software is going through evolution process when changes are requested by the users and finally the new versions are created. Software version management is the process of identifying and keeping track of different versions of software. Complexity level of this process would become complicated should software was distributed in many places. This paper presents a temporal-based software version management model. The model is purposely implemented for managing software versions in Information Technology Centre, Universiti Darul Iman Malaysia. Temporal elements such as valid time and transaction time are the main attributes considered, to be inserted into the software version management database. By having these two attributes, it would help the people involved in software process to organize data and perform monitoring activities with more efficient.

Keywords: version management, temporal database, valid time, transaction time.

1. INTRODUCTION

Software evolution is concerned with modifying software once it is delivered to a customer. Software managers must devise a systematic procedure to ensure that different software versions may be retrieved when required and are not accidentally changed. Controlling the development of different software versions can be a complex task, even for a single author to handle. This task is likely to become more complex as the number of software authors increases, and more complex still if those software authors are distributed geographically with only limited means of communication, such as electronic mail, to connect them.

Temporal based data management has been a hot topic in the database research community since the last couple of decades. Due to this effort, a large infrastructure such as data models, query languages and index structures has been developed for the management of data involving time [11]. Nowadays, a number of software has adopted the concepts of temporal database management such as artificial intelligence software, geographic information systems and robotics. Temporal management aspects of any objects could include:

- The capability to detect change such as the amount of change in a specific project or object over a certain period of time.

- The use of data to conduct analysis of past events e.g., the change of valid time for the project or version due to any event.
- To keep track of all the transactions status on the project or object life cycle.

Universiti Darul Iman Malaysia (UDM) is the first full university at East Coast of Malaysia located at the state of Terengganu. It was setup on 1st January 2006. UDM has two campus named as KUSZA Campus and City Campus. Another new campus known as Besut Campus will be operated soon. To date, KUSZA Campus has six faculties and City Campus has three faculties. The university also has an Information Technology Centre (ITC-UDM) that purposely for developing and maintaining the university information systems and information technology infrastructure.

In this paper, we concentrate on the modelling of a temporal-based software version management. Based on the model, a simple web-based web application has been developed and suggested to be used by ITC-UDM. The rest of the paper is organized as follows: next section reviews the concept of temporal data management. Section 3 discusses on the current techniques in software version management. Current issues in software version management at ITC-UDM are discussed in Section 4. The specifications of the proposed temporal-based software version management model are explained in Section 5. Conclusion is placed in Section 6.

2. TEMPORAL DATA CONCEPT

To date, *transaction time* and *valid time* are the two well-known of time that are usually considered in the literature of temporal database management [2, 4, 6, 9, 10, 11, 12]. The valid time of a database fact is the time when the fact is true in the *miniworld* [2, 6, 9, 10]. In other words, valid time concerns the evaluation of data with respect to the application reality that data describe. Valid time can be represented with single chronon identifiers (e.g., event time-stamps), with intervals (e.g., as interval time-stamps), or as valid time elements, which are finite sets of intervals [9]. Meanwhile, the transaction time of a database fact is the time when the fact is current in the database and may be retrieved [2, 6, 9, 10]. This means, that the transaction time is the evaluation time of data with respect to the system where data are stored. Supporting transaction time is necessary when one would like to *roll back* the state of the database to a previous point in the time. [9] proposed four implicit times could be taken out from valid time and transaction time:

- valid time – valid-from and valid-to
- transaction time – transaction-start and transaction-stop

Temporal information can be classified into two divisions; absolute temporal and relative temporal [9]. Most of the research in temporal databases concentrated on temporal models with absolute temporal information. To extend the scope of temporal dimension, [12] presented a model which allows relative temporal information e.g., “event A happened before event B and after January 01, 2003”. [12] suggests several temporal operators that could be used for describing the relative temporal information: {equal, before, after, meets, overlaps, starts, during, finishes, finished-by, contains, started-by, overlapped-by, met-by and after}.

In various temporal research papers the theory of time-element can be divided into two categories: *intervals* and *points* [6, 9, 11]. If T is denoted a nonempty set of time-elements and d is denoted a function from T to R^+ , the set of nonnegative real numbers then:

$$\text{time_element, } t = \begin{cases} \text{interval, if } d(t) > 0 \\ \text{point, otherwise} \end{cases}$$

According to this classification, the set of time-elements, T , may be expressed as $T = I \cup P$, where I is the set of intervals and P is the set of points.

3. RELATED TOOLS IN SOFTWARE VERSION MANAGEMENT

In distributed software process, a good version management combines systematic procedures and automate tools to manage different versions in many locations. Most of the methods of version naming use a numeric structure [5]. Identifying versions of the system appears to be straightforward. The first version and release of a system is simply called 1.0, subsequent versions are 1.1, 1.2 and so on. Meanwhile, [3] suggests that every new version produced should be placed in a different directory or location from the old version. Therefore, the version accessing process would be easier and effective. Besides that, should this method be implemented using a suitable database management system, the concept of *lock access* could be used to prevent the occurrence of overlapping process. Present, there are many software evolution management tools available in market. Selected tools are described as follows:

- *Software Release Manager* (SRM) – SRM is a free software and supported on most UNIX and LINUX platforms. It supports the software version management for distributed organizations. In particular, SRM tracks dependency information to automate and optimize the retrieval of systems components as well as versions.
- *Revision Control System* (RCS) – RCS uses the concepts of tree structures. Each *branch* in the tree represents a variant of the version. These branches will be numbered by an entering sequence into a system database. RCS records details of any transaction made such as the author, date and reason for the updating.
- *Change and Configuration Control* (CCC) – CCC is one of the complete tools for software configuration management. It provides a good platform for an identification, change control and status accounting. CCC allows a simultaneously working for a same version via virtual copies. This can be merged and changes can be applied across configurations.
- *Software Management System* (SMS) – SMS allows all the aspects in software configuration management such as version control, workspace management, system modelling, derived object management, change detection in the repository etc. SMS possesses the desired characteristics, providing resources of version control of systems and having a good user interface.

