# Managing Software Change Request Process: Temporal Data Approach

**A. R. M. Nordin** mohdnabd@udm.edu.my
*Faculty of Informatics*
*Universiti Darul Iman Malaysia, KUSZA Campus*
*21300 K Terengganu, Malaysia*

**S. Suhailan** mohdnabd@udm.edu.my
*Faculty of Informatics*
*Universiti Darul Iman Malaysia, KUSZA Campus*
*21300 K Terengganu, Malaysia*

**Abstract**

Change request management is the process that approves and schedules the change to ensure the correct level of notification and minimal user impact. Complexity level of this process would become complicated should software was distributed in many places. There are several techniques and tools have been produced to perform change request management. But most of these techniques do not give enough attention towards dynamic aspect of information management such as temporal base elements for the associated data. Therefore, this paper presents a new dimension to support software change request management. Temporal elements such as valid time and transaction time are the main attributes considered, to be inserted into the software change request management system database. By inserting these elements it would help the people involved in change request management to organize data and perform activity monitoring with more efficient.

**Keywords**: change request, temporal database, valid time, transaction time.

## 1. INTRODUCTION

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. Presently, many information systems have adopted the concepts of temporal database management such as geographic information systems and artificial intelligence systems [1][3][6][7]. Temporal management aspects of any objects transaction data 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.

Change request is a formally submits artefact that is used to track all stakeholder requests including new features, enhancement requests, defects and changes in requirement along with related status information throughout the project lifecycle. All change history will be maintained with the change request, including all state changes along with dates and reasons for the change. This information will be available for any repeat reviews and for final closing.

Meanwhile, software change request management is one of the main requirements to achieve quality in software maintenance process. Change request managers must devise a systematic procedure to ensure that all the change requests has been performed by the maintainer in required time span and fulfil with the user requirements. To achieve this goal the managers need to introduce mechanism to handle requests evaluation, and authorization of changes. In addition, the manager also collects statistics about components in the software system, such as information which determining problematic components in the system.

In this paper, we will introduce the technique to improve software change request management and monitoring model using temporal elements such as valid time, transaction time and temporal operators. For this purpose, these temporal elements will be stamped into each activity involve in a change request management. The model develops a database that maintains the changes occurred to the software change request information. Therefore, it provides a historical data retrieval platform for an efficient software maintenance project. In Section 2, the concept of temporal data management is reviewed. Section 3 briefly discusses the current approach of software change request process. The proposed model of temporal based software change request process is discussed in Section 4. Conclusion will be discussed in Section 5.

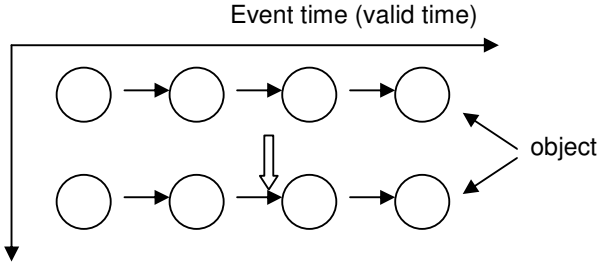
## 2. TEMPORAL DATABASE MANAGEMENT

Most of the data models maintain only concept of current view data which an existing data value is overwritten by a new incoming value during the updating process [12]. To overcome this problem, temporal databases capture the evolution of the modelled reality over time by maintaining many data states e.g., past, current and future. To support this, the use of time stamping is proposed [7]. To date, two well-known time elements which are usually considered in the literature of temporal database are transaction time and valid time.

The valid time of a database fact is the time when the fact is true in the *miniworld*. In other words, valid time concerns with the evaluation of data in 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 [6][7]. Meanwhile, the transaction time of a database fact is the time when the fact is current in the database and may be retrieved. It means, the transaction time is the evaluation time of data with respect to the system where data are stored. Transaction time is necessary when one would like to roll back the state of the database to a previous point in the time. [7] 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

Figure 1 illustrates the object or information will change when any events (respect to valid time) or transactions (respect to transaction time) occurred. These time-elements could be inserted into database systems profile, the database tables or it could be stamped to the records. The vital aspect of this approach is to monitor the events and transactions for the associated data in a better manner. Besides, the process of record management becomes more efficient and effective as a result of no record overlapping.



Transaction time

**FIGURE 1**: Representing Time Element into Two-Dimensions Quantity

In various temporal research papers the theory of time-element can be divided into two categories: intervals and points [6][7][12]. 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. To extend the scope of temporal dimension, [10] have presented a model which allows relative temporal information e.g., "event A happened before event B and after January 01, 2009". [10] have suggested several temporal operators that could be used in describing the relative temporal information: {equal, before, after, meets, overlaps, starts, during, finishes, finished-by, contains, started-by, overlapped-by, met-by and after}.

## 3. CHANGE REQUEST MANAGEMENT PRACTICE

A change request represents a documented request for a change and management of the activities in a change process which is one of a core activity in a software process. Change management means managing the process of change as well as managing all artefacts of an evolving software system. However, it is well known that managing software changes are very difficult, complicated and time consuming. The subject matter of all change request management is the flow of all activities or procedures involved.

A good practice proposed by [13] that suggests several steps to be considered in change request management procedure (Figure 2). The petitioner submit a change request and the project manager evaluates to assess technical merit, potential side effects, overall impact to other components and functions and cost and time estimation. Results of the evaluation are documented as a change request report, which is used to make a final decision on the status approval by the configuration manager or the director of information technology unit. If the change request is approved by configuration manager, the project manager will prioritize the change request project. Before an implementation can be started, project manager will produce service order and deliver it to the project staff. This service order includes new systems specifications, project planning documentation and project contract. Upon completion of the change implementation, a software status report will be produced and a new version will be available.

To accomplish a quality change request management, [5] have suggested that an appropriate database management system is needed to be embedded into the change request management systems. This database is used to keep track of all the transactions within the project running. These transactions includes the recording of change request, project contract, validation of change request, change request approval, change request analysis, prioritizing, implementation and delivering of new version.

[2] have suggested a formal life cycle for a change request process. This life cycle provides a set of specific status for each executed change request. These statuses are: *update, close, reject* and *action*. Figure 3 shows a process flow and a set of status suggested by [2].

From study done by the authors, several weaknesses have been found in the current practice of software change request process model. These weaknesses are as follows:

- Non systematic in the procedure in change request process and it is difficult to recognize the valid time for each stage in the process;

- Many models do not consider the aspect of relative temporal in representing the valid time for each stage in change request process; and
- Most of the existing models maintain only the concept of "current view" of which an existing change request version will be overwritten when the status or information regarding to change request is updated.

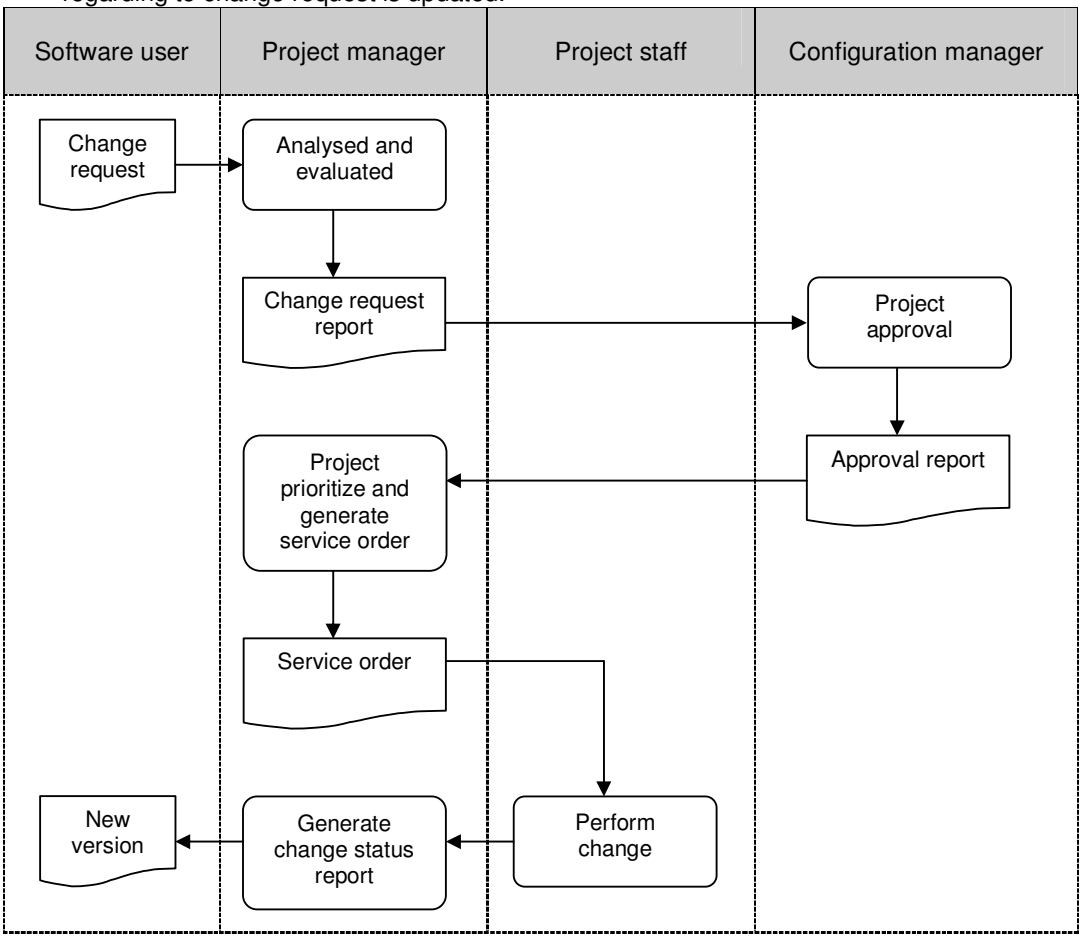



**FIGURE 2:** A change request procedure

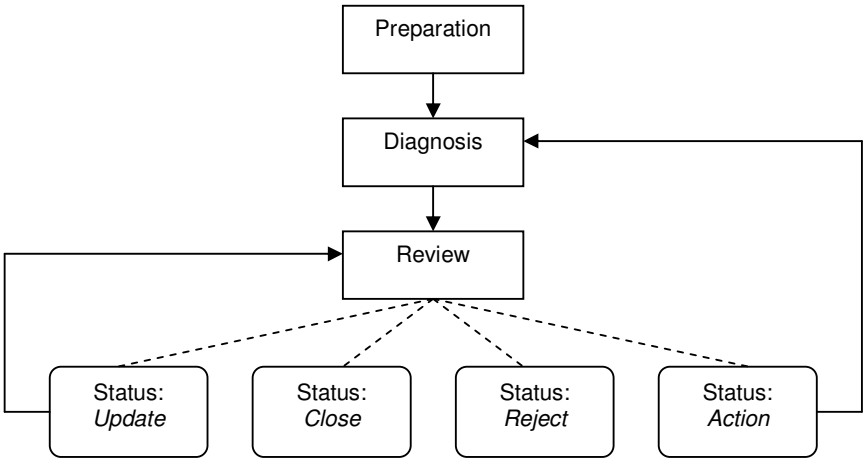



**FIGURE 3:** Change request life cycle

## 4. TEMPORAL BASED CHANGE REQUEST MANAGEMENT

Change request process is one of the main tasks in software configuration management. Updating the document of change request activities is critical to ensure valid and up-to-date information. For any change request in a software life cycle would have its own valid time and a transaction time for each activity. During unplanned changes, it is easy to forget to make updates because of the frantic nature of emergencies. Therefore, the collection of change request should be organized into systematic way for the purpose of retrieval efficiency and valid time recognition. To achieve this goal we strongly believe that, it is very important that the method of time-stamping needs to be embedded into the change request management database.

There are two main functions involved in this proposed model; *update the change request essential information* and *update the temporal information of the change request*. For each change request would have final status and can be categorized into three {In-progress, Completed and Rejected}. Furthermore, the change request would also have three values of priority {High, Moderate, Low}. There are eight change request activities considered and can be denoted as {record, contract, validation, approval, analysis, priority, implementation, deliver}. For each activity in a change request process would have it status and can be classified into two {In-progress, Completed}.

### 4.1 The Model of Temporal Attributes

Temporal elements such as transaction time (tt) and valid time (vt) are the main attributes to be considered in this model. Time element unit considered is in the format of [day/month/year]. Generally, in this model transaction time and valid time could be defined as:

$tt_{(i)} = \{t_i \in P\}$
$vt_{(i)} = \{t_i \in P\}$, if valid time is in point and
$vt_{(i)} = \{\tau_i \in I\}$, if valid time is in interval

where, P = a defined set of point,
I = a set of interval,
$t_i$ = a time point at i and
$\tau_i$ = a time interval at i

Furthermore, this model also considers some temporal operators to be combined with transaction time and valid time. Among the temporal operators used in this model are equal, before, after, meets and met_by. Generally Table 1 shows the definition for each temporal operator base on a time point and a time interval. Meanwhile, figure 4 illustrates the state of temporal operators based on a define time point, $t_i$, and figure 5 shows the state of temporal operators based on a defined time interval, $\tau_i$.

| Temporal Operator | Time Point | Time Interval |
|---|---|---|
| equal | $t = \{(t = t_i) \in T\}$ | $\tau = \{(\tau = \tau_i) \in T\}$ |
| before | $\tau = \{(\tau < t_i) \in T\}$ | $\tau = \{(\tau < \tau_i) \in T\}$ |
| after | $\tau = \{(\tau > t_i) \in T\}$ | $\tau = \{(\tau > \tau_i) \in T\}$ |
| meets | $\tau = \{(\tau \leq t_i) \in T\}$ | $\tau = \{(\tau \leq \tau_i) \in T\}$ |
| met_by | $\tau = \{(\tau \geq t_i) \in T\}$ | $\tau = \{(\tau \geq \tau_i) \in T\}$ |

**TABLE 1:** The definitions of temporal operator base on time point and time interval

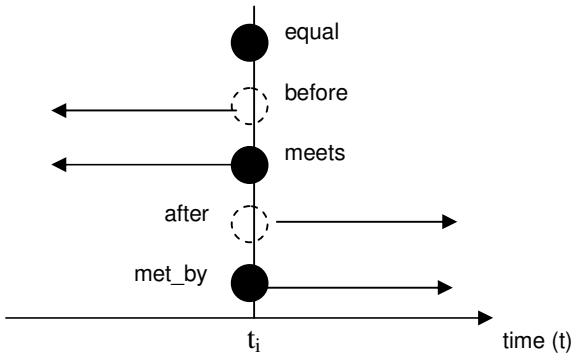



**FIGURE 4**: The state of temporal operators based on a time point

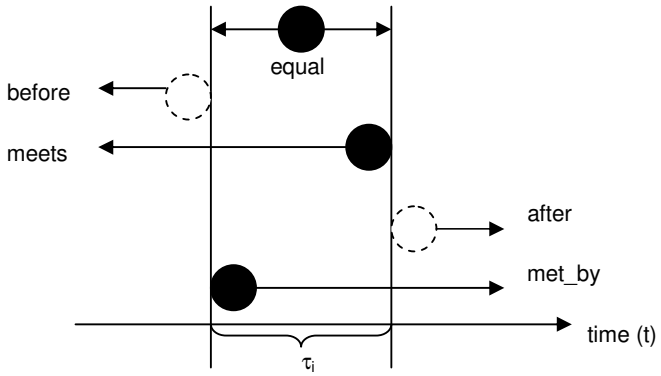



**FIGURE 5:** The state of temporal operators based on a time interval

### 4.2 Temporal Aspects in Change Request Management

As mentioned earlier, temporal elements involved here are transaction time (tt) and valid time (vt) which can be denoted, TE = {tt, vt}. Transaction time represents the date when change request status activity is recorded into the Change Request Database Management System (CRDMS). Meanwhile, valid time represents the change request time-span e.g. the date of change request report is submitted to the date of new version is required. Hence, valid time would be categorizes into two different attributes known as valid-from and valid-until. In this model, only five temporal operators will be considered, and can be denoted as OP = {equal, before, after, meets, met_by}. Therefore, if we have a set of change request signed as, C = $\{c_1, c_2, c_3, \ldots, c_n\}$ then the model is,

$$\text{TEMPORAL}(c_i \in C) \subseteq (tt \cap vt)$$

where, vt = [vt-from, vt-until]

If a change request which has a set of feature attributes $A_i$ then a complete scheme for a temporal based in change request management can be signed as:

$$S = \{A_1, A_2, A_3, \ldots, A_n, tt, op1, vt\text{-}from, op2, vt\text{-}until\}$$

where, $A_i$ = attribute name,
$tt \in P$,
op1, op2 $\in$ OP and
vt-from, vt-until $\in T$

Figure 6 shows in general the insertion point of valid time and transaction time into all stages in software change request management process. As an example, we may illustrate the record transaction representing C_REQ0120's change request life cycle as in Table 2. Regarding to this model, the following convention can be used to retrieve for any temporal attribute in proposed temporal based software change request management:

- Attribute name only ⇒ current time
- Attribute name followed by ALL ⇒ all history
- Attribute name followed by a time point or interval ⇒ specific time period
- Attribute name followed by RESTRICT ⇒ specific time period designated by the condition

| c_req# | tt | activity | op1 | vt-from | op2 | vt-until |
|---|---|---|---|---|---|---|
| CREQ0120 | tt1 | Record | op11 | vf1 | op21 | vu1 |
| CREQ0120 | tt2 | Contract | op11 | vf1 | op21 | vu1 |
| CREQ0120 | tt3 | Validation | op11 | vf1 | op21 | vu1 |
| CREQ0120 | tt4 | Approval | op11 | vf1 | op21 | vu1 |
| CREQ0120 | tt5 | Priority | op11 | vf1 | op21 | vu1 |
| CREQ0120 | tt6 | Priority | op12 | vf2 | op21 | vu1 |
| CREQ0120 | tt7 | Implementation | op12 | vf2 | op21 | vu1 |
| CREQ0120 | tt8 | Implementation | op12 | vf2 | op21 | vu2 |
| CREQ0120 | tt9 | Implementation | op13 | vf2 | op22 | vu3 |
| CREQ0120 | tt10 | Deliver | op13 | vf2 | op22 | vu3 |

**TABLE 2**: Record transaction for CREQ0120 change request

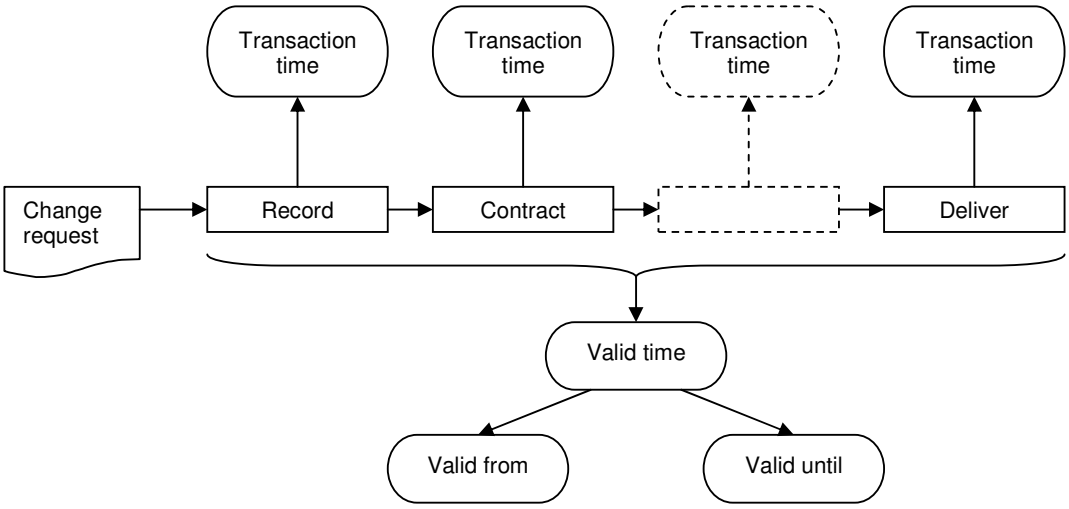



**FIGURE 6:** Temporal aspects in change request management

### 4.3 Model Evaluation

The authors have performed an evaluation process for the developed prototype of the model. In this evaluation process, 18 users (software engineers / programmers) that involved in software configuration management process are selected. The evaluation is based on the

problems that constantly faced by the users. Table 3 tabulates the comments from the users regarding to the use of the model prototype in order to solve their problems.

| Problem | Disagree (%) | Agree (%) | Strongly agree (%) |
|---|---|---|---|
| To validate the current status of each stage in change request process. | 28 | 55 | 17 |
| To track the history information of a maintenance process | 17 | 72 | 11 |
| No CASE tool in managing and monitoring software maintenance process | 6 | 83 | 11 |

**TABLE 3**: The users perception in using the model prototype for solving their problems

## 5. CONCLUSION

This paper introduces a new model in change request management based on temporal elements. Here, an important issue discussed is temporal aspects such as valid time and transaction time which have been stamped on each activity in change request so that the monitoring and conflict management processes can be easily made. The proposed model also suggested that each activity should be associated with a defined progress status. We hope that the material and model presented in this paper will be useful to support future work on. For further improvements, currently, we are investigating to consider more temporal operators and developing a standard temporal model for all configuration items in software configuration managements.

## 6. REFERENCES

1. C. E. Dyreson, W. S. Evans, H. Lin and R. T. Snodgrass, *"Efficiently Supporting Temporal Granularities"*, IEEE Transaction on Knowledge and Data Engineering, Vol. 12 (4), 568 – 587, 2000.

2. C. Mazza, J. Fairclough, B. Melton, D. DePablo, A. Scheffer, R. Stevens, M. Jones and G. Alvisi. *"Software Engineering Guides"*. Prentice Hall. 1996.

3. D. Gao, C. S. Jensen, R. T. Snodgrass and M. D. Soo, *"Join Operations in Temporal Databases"*, The Very Large Database Journal, Vol. 14, 2 – 29, 2005.

4. S. Dart. *"Concepts in Configuration Management Systems"*. ACM – 3rd International Workshop on Software Configuration Management, 1 – 18, 1991.

5. A. Deraman & P. J. Layzell. *"A Computer-aided Software Maintenance Process Model"*. Heuristic – Journal of Knowledge Engineering. 6(1):36 – 42, 1993.

6. H. Gregerson & C.S Jensen. *"Temporal Entity-Relationship Models – A Survey"*. IEEE Transaction on Knowledge and Data Engineering. 11(3): 464 – 497, 1999.

7. C. S. Jensen & R.T. Snodgrass. *"Temporal Data Management"*. IEEE Transaction on Knowledge and Data Enginering. 11(1):36 – 44, 1999.

8. G. Joeris. *"Change Management Needs Integrated Process and Configuration Management"*. 6th European Software Engineering Conference, 125 – 141, 1997.

9. B. Knight & J. Ma. *"A General Temporal Theory"*. The Computer Journal. 37(2):114-123, 1994.

10. B. Knight & J. Ma. *"A Temporal database Model Supporting Relative and Absolute Time"*. The Computer Journal. 37(7):588 – 597, 1994.

11. K. Torp, C. S. Jensen and R. T. Snodgrass, *"Effective Timestamping in Database"*, The Very Large Database Journal, Vol. 8, 267 – 288, 1999.

12. D. H. O. Ling & D. A. Bell. *"Modelling and Managing Time in Database Systems"*. The Computer Journal. 35(4):332 – 342, 1992.

13. R. F. Pacheco. *"Keeping the Software Documentation Up to Date in Small Companies"*. CLEI Electronic Journal. 3(1), 2000.