Bitemporal Support for Business Process Contingency Management

John Wondoh[1], Georg Grossmann[1], Dragan Gasevic[2], Manfred Reichert[3], Michael Schrefl[4], and Markus Stumptner[1]

[1] University of South Australia, Australia
john.wondoh@mymail.unisa.edu.au,
{georg.grossmann,markus.stumptner}@unisa.edu.au
[2] University of Edinburgh, UK
dgasevic@acm.org
[3] Ulm University, Germany
manfred.reichert@uni-ulm.de
[4] Johannes Kepler University of Linz, Austria
schrefl@dke.uni-linz.ac.at

Abstract. Modern organisations are increasingly moving from traditional monolithic business systems to environments where more and more tasks are outsourced to third party providers. Therefore, processes must operate in an open and dynamic environment in which the management of time plays a crucial role. Handling time, however, remains a challenging issue yet to be fully addressed. Traditional processing systems only consider business events in a single time dimension, but are unable to handle bitemporal events: events in two time dimensions. Recently, back-end systems have started to provide increased support for handling bitemporal events, but these enhanced capabilities have not been carried through to business process management systems. In this paper, we consider the possible relationships that exist between bitemporal properties of events and we show how these relationships affect a business process. In addition, we demonstrate how bitemporal events can be handled to prevent certain undesired effects on the business process.

Keywords: Bitemporal events, guard-stage-milestone (GSM), artifact-centric business process modelling, bitemporal business rules

1 Introduction

Time is an essential aspect in business process modelling and, therefore, has gained much attention over the years [9, 8, 11]. Business process management systems (BPMSs) need to reflect what is happening in the real world in a timely manner. The ability to model temporal dimensions of the real world within BPMSs is important to enable business processes to be time-aware, flexible and adaptive. Events and objects are the main parts of a business process captured in temporal databases. An event occurs at a specific point in time and while an object exists within a time interval. In general, a temporal database is a database system that supports time perspective and, thus, is able to manage time-varying data [12]. We focus on business events as they can be used to instantiate business processes or activities and monitor the progress of business processes.

There are two main time dimensions in temporal databases as discussed in [13]: (i) *business time (or valid time)* captures the time a fact becomes true in reality; (ii) *system time (transaction time)* records the time the fact was captured in a database. A *bitemporal database* captures both business time and system time. A *single time dimension* is used to refer to either business or system time while *two time dimension* is used when both time dimensions are considered.

Traditional BPMS are built to handle ideal situations where it is assumed that there are no system outages nor communication delays that can result in delayed event consumption [15, 3]. This results in three main problems with traditional systems:

- Events are considered in only a single time dimension and are ill-equipped to handle situations that require the analysis of two time dimensions.
- In a distributed event-based system, the difference between the time an event is sent by an external system (external system time) and the time the event is received by the local system (local system time) is not considered.
- Delays in processing an event after it has been received are not considered.

Consider the scenario where home care is provided by an organisation to some patients discharged from hospitals. Work in such organisations is planned well in advance at a higher level. Typically, workplace legislation requires that the schedules for staff to be in place weeks in advance. This is often produced by an automated scheduling tool based on optimisation software that must consider many different factors such as: general skill level of the nurse; required level of support for the patient; minimum travel distance of medical personnel to patients residence; and the same set of medical personnel for a given patient to establish a relationship with the patient.

Once in place, however, the actual schedules are subject to frequent disruption due to variation in the timing of real world events. For example, a patient scheduled for home care might be discharged from the hospital earlier than expected. The moment this becomes known, the organisation must react to this contingency by making adjustments to its original schedule. Notably, these changes do not involve rerunning the BPMS as that would be slow to respond and disruptive to long-term planning. Instead, the BPMS should be adaptive and flexible enough to incorporate these changes on the fly.

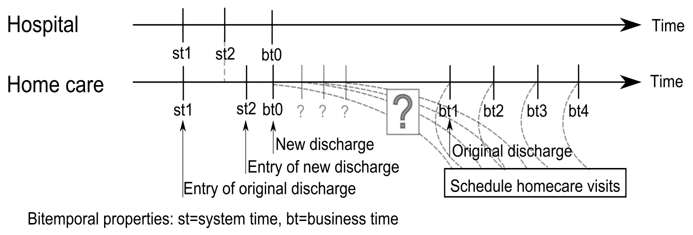

Fig. 1. Bitemporal Scenario in Hospital–Home Care Process

A concrete scenario is shown in Figure 1. At time st_1, a patient is scheduled to be discharged at time bt_1. Following this, regular home care is scheduled for times bt_2, bt_3 and bt_4. Further, assume that the patient is discharged from the hospital earlier (at time bt_0) compared to the scheduled time (bt_1). In addition, this fact is updated in the hospital system at st_2, which may not be immediately communicated to the home care organisation. Once the new discharge time is known to the home care organisation, it will prepare and provide home care for the patient within a shorter time interval.

A lack of support for two time dimensions in business processes might result in delayed reactions to discrepancies, inappropriate actions being taken, and contract and legal issues in an inter-organisational process. In addition, the external system time and the processing time of an event must be considered during decision making. Considering bitemporal properties of events introduces additional complexity to the execution of business processes, which traditional BPMS are incapable of handling. Therefore, there is the need to consider respective times during event processing in business processes.

This paper addresses the issue of discrepancies between system time and business time by providing bitemporal support for handling two time dimensions within business processes. We first provide a classification of the relationships between local system time and business time as well as the external system and processing system time of an event. We proceed to discuss the impact of these relationships to a business process and our approach to handling them.

The remainder of this paper is organised as follows: Section 2 provides a brief background to bitemporal databases. Section 3 discusses bitemporal properties of business events and the permitted relationships that exist between them and Section 4 presents an approach to handling these relationships in BPMSs. Section 5 discusses related work and Section 6 concludes the paper and discusses future work.

2 Preliminaries

Our approach relies mainly on the concepts of bitemporal databases, which considers time in two dimensions; i.e, business time and system time. Recently, commercial database systems have provided support for two time dimensions. The SQL:2011 standard [7] as implemented in IBM DB2 [10] and Teradata [6], for example, provides a means to define time periods and to associate them with other attributes. Furthermore, users are provided with the means to distinguish between system time and business time, as well as to choose a preferred time dimension. A business event captured within a bitemporal database is denoted as *bitemporal event* and is given the following temporal properties: business time (bt), and system time (st).

System time records the time when changes are applied to data within the database, i.e., when data is updated in the system. A history of updated and deleted rows is maintained and can be accessed whenever needed. Business time captures the time at which changes to business objects occur in the real world. Usually, it is entered into the database by an external user. As another significant difference, system time only captures past and present times, while business time captures future time as well. Other databases (e.g. Oracle) support bitemporality with a syntax different from that of SQL:2011. A comparison of bitemporal database systems in provided in [6].

In the SQL syntax, both business and system time are time intervals, having a start time and an end time. Since we consider atomic business events occurring at a specific point in time, we adopt the approach presented in [?] to represent both times. The business and system time of an event are represented by the start time of their respective time intervals. The end time is either infinite (its effect is not yet completed) or represent the time the effect of the event is completed in the process. Not considering the end time of an event, its business and system time can be captured as a single time.

3 Bitemporal Properties of Events

This section, addresses the relationships permitted between bitemporal properties of events and how these relationships affect the business process. We first consider the possible relationships that exist between the local system time and the business time of an event. Further, we consider the distributed event-based system where an event may have a system time from its source and a system time from its receiver. Unlike traditional BPMSs, we consider the possibility of delays since the latter occur in the real world and have a potential impact on the execution of a business process. We proceed to consider the time an event is processed, how this time can differ from the time an event is received, and implications of this discrepancy.

3.1 Bitemporal Property Categorisation

There is a number of relationships that may exist between the system and business time of an event. In this section, local system time shall be referred to system time for simplification. Each relationship affects the business process in a particular way. We sub-divide these relationships into four main categories based on their time of occurrence and their update st............................ as shown in Figure 2. Further, we discuss the nature of each relationship and how it affects a business process. This will equip business process designers with the knowledge of how to handle each type of relationship.

Category 1 (Past and Present). First, we consider the three basic relationships that exist between business and system times of an event occurring within a business process. Assuming e is a bitemporal event that has occurred with properties st_1 and bt_1 by comparing the bitemporal properties of the event, two relationships can be derived: $bt_1 = st_1$ and $bt_1 < st_1$. The former represents *On-Time*, which indicates that system time is equal to business time, i.e. the event is on time. No discrepancy exist and the event is detected in the real world at the same time it is captured in the database system. A late, $bt_1 < st_1$, event may be the result of a delayed detection of an event which may result in delayed processing of the event and deadlocks within the process. An event may not be processed at all or may be considered irrelevant if the process has no late event handling capabilities.

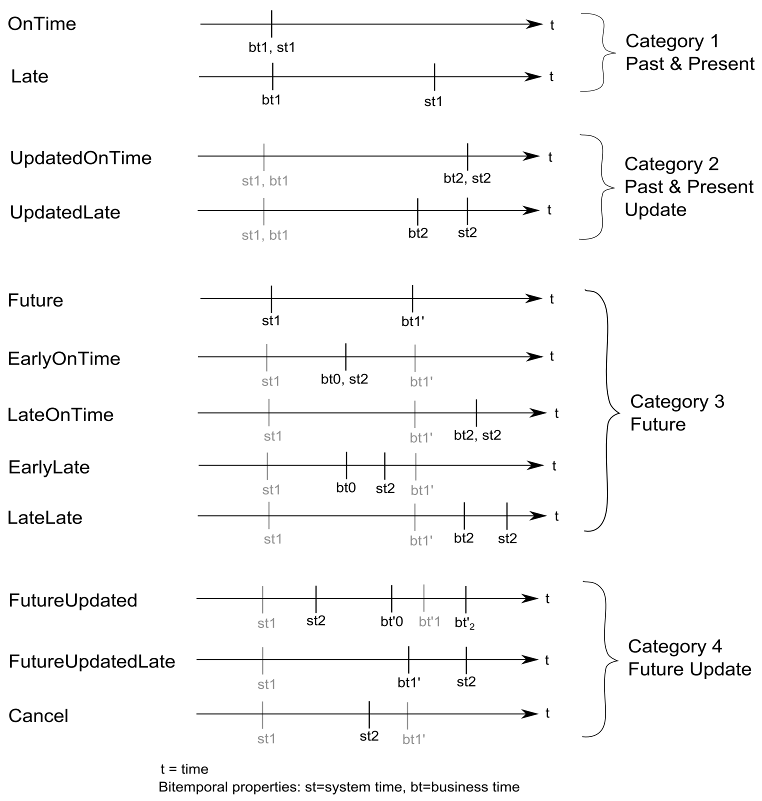

Fig. 2. Relationship between System Time and Business Time

Category 2 (Past and Present Update). An event that has already occurred within a business process may be updated when additional information is obtained about the event or when some information needs to be changed. In this category, we consider updates applied to events in Category 1. Once the update has occurred, the original business and system time (bt_1, st_1) will change to their respective new times (bt_2, st_2). We assume that the original event is an OnTime event, even though this may not always be the case. We draw a distinction between *UpdatedOnTime* and *UpdatedLate* as shown in Figure 2. UpdatedOnTime means the new business and system time are equal, while UpdatedLate has a system time that is a later time compared to its business time.

Category 3 (Future). Within business processes, certain events are planned to occur in future or expected to happen sometime in future. These events, denoted as future events, have a business time, bt'_1, that is a future time. Being able to capture future events and changes that can occur to these events during business process execution is essential to managing contingencies within the process. The relationships in this category corresponds to changes in the business time as events occur in the real world.

1. *Future*: A future event is an event that has not yet occurred in the real world.
2. *EarlyOnTime*: A future event occurs at an earlier time w.r.t its planned time. The system time, st_1, is updated with a new timestamp, st_2 and the future business time, bt'_1, is replaced by an earlier time, bt_0. The system is notified immediately about the change in the business time, i.e $st_2 = bt_0$.
3. *LateOnTime*: A future event that occurs at a later time compared to its expected time. Its system time, st_1, is updated to st_2 and the future business time, bt'_1, is replaced with bt_2. The system is notified immediately about the change in the business time, i.e $st_2 = bt_2$.
4. *EarlyLate*: same as 2 (EarlyOnTime) except that the system is notified late about the change in the business time, i.e $bt_0 > st_2$.
5. *LateLate*: same as 3 (LateOnTime) except that st_2 occurs sometime after bt_2.

Category 4 (Future Update). In business processes, updates can be made to a future event before its occurrence. The event information is updated and its original system time is updated to a new system time. The future business time may change to an earlier or later future time with respect to the original future time as shown in Category 4 in Figure 2. There are three relationship types in this category: *FutureUpdated*, *FutureUpdatedLate*, and *Cancel*. For FutureUpdated, the update is made before the future business time; i.e, bt'_1. The system time, however, is updated from st_1 to st_2. A new future business time may be set, which may be earlier, i.e bt'_0 or later, i.e bt'_2 with respect to bt'_1. FutureUpdatedLate is similar to FutureUpdated, except that, the update occurs after the bt'_1 has occurred. A later business time is set which may be a future time bt'_2. The future event can be also prevented from happening by cancelling it, in which case bt'_1 becomes become irrelevant and st_1 is updated to st_2 (timestamp corresponding to when the cancellation took place).

3.2 Local and External System Time

So far, we have categorised and discussed the relationships that exist between the business and system time of an event. In a distributed event-based system, discrepancies may exist between the time an event was produced and when it is detected by a BPMS. As discussed in Section 1, this may result in delayed reactions to discrepancies, inappropriate actions being taken, and contract or legal issues in an inter-organisational process. Therefore, this discrepancy needs to be accounted for. Events in a distributed event-based system consist of two system times, i.e st_e and st_l, where st_e is the system time obtained from the event producer (external system time) and st_l is the system time registered locally in the event consumer (local system time). When an event is received by a local system immediately it is sent by an external system, then $st_l = st_e$. On the other hand, $st_l > st_e$ indicates that the event was received late. Since these two system times may differ due to system outages and delayed or disrupted communication, we need take both system times into consideration in our processing.

3.3 Processing Time

The time an event is processed constitutes an important temporal aspect to be considered in a BPMS since events are not always processed immediately upon receiving them. In other words, an event may be processed either immediately when receiving it or it may be processed late. Delayed event processing may be either planned or unplanned. Planned delays include event queuing systems and event prioritization, whereas unplanned delays include communication disruption and system outages. Either type of delay, however, requires the consideration of the time the event was received during the event processing. This is important to ensure that correct event records are maintained as well as avoiding violation of temporal requirements.

Let us denote the time an event, e, is processed within a business process to be st_p. In turn, st_l is the local system time, which is the time the event is received by the system. The permitted relationship between these two times is as follows: $st_p \geq st_l$. That is, an event cannot be processed before it is received. If $st_p = st_l$ then the event is processed immediately when it is received without any time lag. Else if $st_p > st_l$ then the event is processed at a later time compared to the time it was received.

4 Handling Bitemporal Events in Business Processes Execution

Information from bitemporal event properties in combination with artefact structure and process execution information such as execution traces, can increase the automation of contingency management. In this section we provide an overview and examples of some of the benefits. Fig. 3 provides the main components of process engine architecture capable of handling bitemporal events and indicates the information flow between the components.

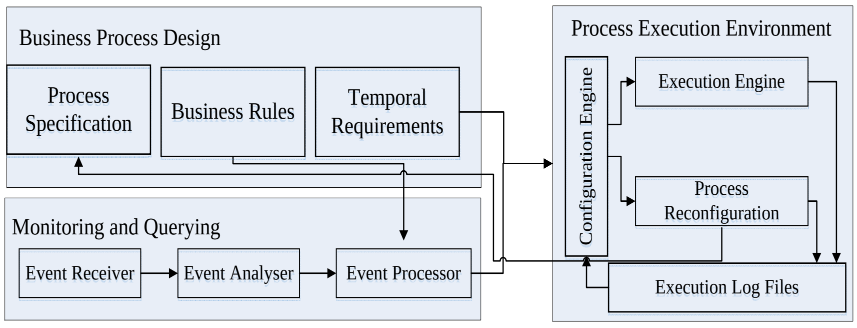

Fig. 3. Dynamic Bitemporal Event Handling Architecture

4.1 Monitoring and Querying

The monitoring and querying component is responsible for receiving, analysing and processing events within the business process. The subcomponents determine the relationships between bitemporal properties and temporal requirements of the process: First, bitemporal event information is obtained by the *Event Receiver* and analysed by the *Event Analyser*. The analyser extracts bitemporal properties from the event and determines if the event was received late or on time, and compares the local system time, business time and external system time. The event processor matches the information from the analyser with event and condition information from business rules and executes relevant actions if event and condition of a rule is met.

4.2 Process Execution Environment

The business process execution environment controls the execution of a business process, manages process instances and re-configures business the business process. Bitemporal information obtained from the *Monitoring and Querying* component is used to adapt the process if temporal requirements are violated. The input for the process execution are temporal requirements of the process as well as bitemporal event information. This information is used to determine how the process shall proceed during execution. During process execution, all information pertaining to activities within the process are stored in the execution log files system. This includes information about the state of actives, i.e., it keeps a record of whether an activity has been completed, is in progress, or is uninitiated. This information, the temporal requirements and bitemporal information obtained from the monitoring and querying stage serve as the inputs to the configuration planner. The configuration planner is responsible for deciding how to process should proceed. [There are three options available in the process execution stage: normal execution, alternative execution, and process reconfiguration.]

Normal execution is when no change is required to occur to the process and execution can continue as planned without violating the temporal requirements of the process. For example, the scheduled home visits designed by a home care organisation is executed as planned. Alternative execution involves the anticipation of scenarios where bitemporal information received may indicate violation of the temporal requirements of a process. Alternative paths are provided at the process design time such that during such anticipated scenarios, the most appropriate path that will result in compliance with the temporal regulations of the process is selected. For example, a patient discharged earlier than the expected time (EarlyOnTime or EarlyLate), requiring an emergency scheduling process to be put in action. This can be anticipated at design time and this emergency schedule process put as an alternative to the normal scheduling process.

A business process model or instance may require rearrangement of its components as well as eliminating existing components or adding new components in order to avoid violating the temporal requirements of the process. Process reconfiguration equips the business process with such functionalities. There are two approaches to process reconfiguration: late binding and late modelling. With late binding, process fragments are created and stored in a process repository and added to the process model when needed. Alternative execution is a type of late binding with the binding options set out at the design time of the process. In late binding, the binding options may not necessarily be available at the design time of the process and may only be realised during execution. Late modelling is similar to late binding but requires the process fragments to be modelled during the execution of the process.

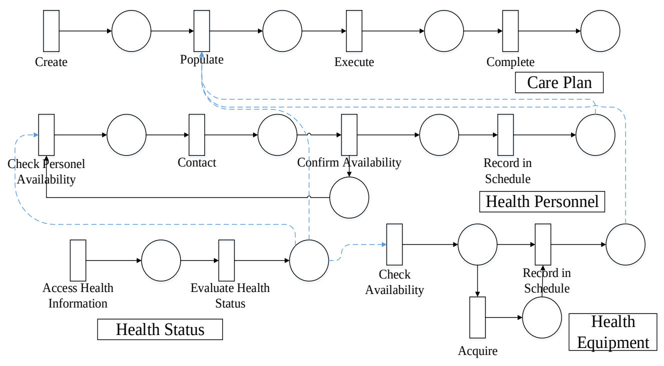

Fig. 4. Home Care Planning Process

The example of such a process is shown in Figure 4 with the care plan artifact being the main process. Other artifacts such as the health personnel, health equipment and health status are required to populate the care plan. The health status artifact is required to be completed in order for the health personnel and health equipment artifact to begin. Once the care plan has been populated with all the necessary information and meeting the requirements given above, execution can start in a timely manner.

In Figure... provide an event with the following properties is sent to the home care facility

event name: Home Care Request: Patient A business time:

5 Related Work

Temporal support for business processes has been of great interest to the community with most research focusing on preventing violations of temporal constraints in business process or provide contingencies when they occur. Some notable examples of works in this area include [11, 14, 8, 9]. Lanz et al [8] defines basic modelling elements for design time-aware business process schemas. They take into consideration the dynamic nature of process instance and temporal constraints on some processes. An approach for avoiding deadline violation during business processes execution is provided in [11], where activities within the process are performed in a flexible and time-aware manner in order to proactively avoid violating the set deadline. While these approaches provide support for temporal aspect of business process they do not take into consideration the two dimensional nature of time. Therefore, they do not consider the effect of these properties on the business process.

An approach that considers time in two dimension is presented in [1], where they distinguish between occurrence time and detection time of an event with the aim of handling uncertainties with event occurrences. They consider the situation where the detection time of an event known but its exact occurrence time is unknown. In this work, we do not deal with uncertainty because we assume that all the bitemporal information of an event needed is captured within the bitemporal database.

The most closely related work was presented by Furche et al [4]. This work investigated bitemporal complex event processing of web events. They distinguished between occurrence time and detection time and proposed an event processing language that can be mapped to standard SQL. The main difference to this work is that they did not consider how this applies to business processes and how it can be handled. In addition, they did not draw a distinction between external system time, local system time and processing time of an event and assumed these can be represent by a single system time (i.e external system time = local system time = processing time).

6 Conclusions

This paper introduces the concept of bitemporality of events into business process management. We discuss the permitted relationships that exist between bitemporal properties and how the impact these relationships have on a business process. We proceed to propose a contingency approach to handle these relationships during business process execution to as to avoid violation of temporal requirement. Future works includes implementing the architecture on top of a bitemporal database and then evaluating the framework.

References

1. Artikis, A., Etzion, O., Feldman, Z., Fournier, F.: Event processing under uncertainty. In: Proc. DEBS. pp. 32–43. ACM (2012)
2. Cheikhrouhou, S., Kallel, S., Guermouche, N., Jmaiel, M.: On enabling time-aware consistency of collaborative cross-organisational business processes. In: Proc. ICSOC, LNCS, vol. 8831, pp. 351–358 (2014)
3. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A.: Fundamentals of business process management. Springer (2013)
4. Furche, T., Grasso, G., Huemer, M., Schallhart, C., Schrefl, M.: Bitemporal complex event processing of web event advertisements. In: Proc. WISE 2013, LNCS, vol. 8181, pp. 333–346 (2013)
5. Gani, K., Bouet, M., Schneider, M., Toumani, F.: Formal modeling and analysis of home care plans. In: Proc. ICSOC, LNCS, vol. 8831, pp. 494–501 (2014)
6. Kaufmann, M., Fischer, P.M., May, N., Kossmann, D.: Benchmarking bitemporal database systems: Ready for the future or stuck in the past? In: Proc. EDBT. pp. 738–749 (2014)
7. Kulkarni, K., Michels, J.E.: Temporal features in sql:2011. SIGMOD Rec. 41(3), 34–43 (Oct 2012)
8. Lanz, A., Posenato, R., Combi, C., Reichert, M.: Controllability of time-aware processes at run time. In: Proc. OTM, LNCS, vol. 8185, pp. 39–56 (2013)

9. Lanz, A., Weber, B., Reichert, M.: Time patterns for process-aware information systems. Requirements Engineering 19(2), 113–141 (2014)
10. Nicola, M.: Best practices: Temporal data management with db2. Tech. rep., IBM (2012)
11. Pichler, H., Wenger, M., Eder, J.: Composing time-aware web service orchestrations. In: Proc. CAiSE, LNCS, vol. 5565, pp. 349–363 (2009)
12. Salzberg, B., Tsotras, V.J.: Comparison of access methods for time-evolving data. ACM Comput. Surv. 31(2), 158–221 (Jun 1999)
13. Snodgrass, R.: Temporal databases. In: Theories and Methods of Spatio-Temporal Reasoning in Geographic Space, LNCS, vol. 639, pp. 22–64 (1992)
14. Sun, H., Yang, J., Zhao, W., Su, J.: Ticobtx-net: A model to manage temporal consistency of service-oriented business collaboration. Services Computing 5(2), 207–219 (April 2012)
15. Wieringa, R.J.: Design methods for reactive systems: Yourdon, statemate, and the UML. Elsevier (2003)