Offstage Objects and their Renovations in the Temporal Object System TOS

Abad A. Shah¹, F. Fotouhi², W. Grosky², S.P. Rana², and A. Vashishtha²

¹Department of Computer Science, King Saud University, Riyadh, Saudi Arabia
²Department of Computer Science, Wayne State University, Detroit, Michigan 48202, U.S.A.

email: fotouhi@cs.wayne.edu

ABSTRACT

TOS, the temporal object system proposed in¹,²) maintains changes to both the structure and state of an object. Objects in TOS are referred to as temporal objects. In this paper we show how the aggregation of temporal objects in a family, i.e., collection of temporal objects with the same common knowledge, allows the construction of an offstage object. During its life-span, an offstage object may go through changes due to the changes in its participant temporal objects. We refer to this as the renovation of an offstage object. Here, we discuss two types of renovations for such an object.

1. INTRODUCTION

Relational databases and their temporal extensions are suitable for simple record-based applications, but are not suitable for engineering and other complex database applications due to their limitations in defining a complex object directly and at one place¹). Several types of object-oriented databases have been proposed to support the development of these application systems. In the object-oriented paradigm, an object is defined by the two parameters structure and state. The structure (SR) of an object provides the structural and behavioral capabilities of that object, which is defined by a set of instance variables, methods, and rules (or integrity constraints). The state (ST) of an object assigns data values to the instance variables of the objects and methods which operate on them. A set of objects sharing the same structure is referred to as a class. An object-oriented database is a collection of classes which are organized as a DAG. Figure 1 shows the schema of an Automobile Manufacturing Database. In this figure bold lines represent part-of relationships.

In the existing object-oriented database systems, changes in the state of an object are maintained via version management¹,⁶). Also, structural changes are supported in most object-oriented database systems. Such changes to a class are referred to as schema evolution in the literature⁶). There are three possible scenarios for a class to change its structure. These are:

Type I) Adding new instance variables, methods, and/or rules
Type II) Deleting instance variables, methods, and/or rules
Type III) Changes to instance variables, methods, and/or rules

Figure 1. Schema of the class Vehicle

Figure 2. Type II change for the class Vehicle

For type I changes, there is no loss of knowledge of a class because the previous knowledge of the class structure is also retained along with the new one. On the other hand, for type II and type III changes, the history of changes in a class structure is not readily available, as it is overwritten or deleted in the latest version of the class structure. Current object-oriented database systems keep only the current version of each class structure. After any one of the type II or type III changes, it is necessary to reload a previous version of the database to retrieve any information from a previous version of a class structure. For example, Figure 1 and Figure 2 are two snapshots of the class structure for the class Vehicle at time instances t₁ and t₂ where t₁ ≤ t₂, respectively. In Figure 2, instance variable Carburetor in the class Engine is replaced by a new instance variable Fuel-Injector. After this change,
all queries about the class Engine can be answered from its current version of the class structure (Figure 2) unless the user explicitly asks for the previous version of class Engine.

In 4,5), we introduced a temporal object-oriented system (TOS) which maintains the history of changes to both the structure and the state of an object in a consolidated and elegant manner. We associated time (point model) to both structure and state of an object. Such an object is referred to as a temporal object. A temporal object evolves over time by changing its state or structure. A set of temporal objects which share a common knowledge (i.e., structure and state) is referred to as a family. The TOS also facilitates the construction of a complex family which is an aggregation of temporal objects from various families. The objects in a complex family are referred to as temporal complex objects. A complex family increases the knowledge sharing of non-homogeneous temporal objects and the transportability of them. A temporal object system (TOS) is then a collection of families which are defined at different time instances.

In this paper we briefly discuss TOS, simple and complex families, temporal objects, temporal complex objects, and offstage objects. We also describe renovation of offstage objects. The remainder of this paper is organized as follows: In Section 2, we describe the TOS and its objects. In Section 3, we discuss renovation of the offstage objects and define two types of renovations. Finally, in Section 4 we give concluding remarks and future research directions.

2. TEMPORAL OBJECT SYSTEM

As mentioned in the previous section Temporal Object System (TOS) is a collection of simple and complex families which are defined at different time instances. A family is a collection of temporal objects and a temporal object is a collection of stages. Figure 3 shows a schema of TOS, where RTOS represents the root node of the system with n families, i.e., F₁, F₂, ..., Fₙ as its children. The figurative notations which are used in Figure 3 and in other figures are give in the appendix.

Now we first give a brief overview of the temporal objects which are building blocks of the family, and then we discuss the concept of families.

2.1 Temporal Objects

As mentioned in the previous section, an object is represented by its structure and state. With the passage of time an object may change its structure and/or its state. By associating time to both the structure and the state of an object, we can keep the history of changes to that object. Therefore, we define a temporal object (TO) to be an ordered set of objects which is constructed at different time instances. A temporal object is represented as TO = < (SR₁, ST₁), (SR₂, ST₂), ..., (SRₙ, STₙ) > where ti ≤ ti⁺ for all 1 ≤ i < n, where the ordered pair (SRi, STi) is the i-th object of the temporal object which is constructed at the time instance ti with structure SRi and state STi. An i-th object of the temporal object is referred to as its i-th stage 4,5).

A stage is maintained in a prototypical form, i.e., a structure, a state, or a combination of the two 3). For example, if a temporal object suffers a structural change, then new stage of the temporal object captures only the structural change. We are using time instance as a physical time and time point. A temporal object may also be referred to as an ordered set of stages. For example, in Figure 4 the temporal object TO of the family F₁ has n stages. The first and last stages of a temporal object are significant because they hold the initial and current knowledge of the temporal object. We refer to these stages as the birth stage (stage S₁ᵢ in Figure 4) and the current stage (stage Sₙᵢ in Figure 4) of the temporal object. The time from the birth stage of a temporal object to the time of its current stage is referred to as the life-span of the temporal object. The set of temporally ordered stages of a temporal object is referred to as the life-sequence of the object. A new stage is appended to a temporal object if the structure and/or state associated with its stage changes (see 4,5,11) for more details).

Figure 3. Schema of the Temporal Object System (TOS)

Figure 4. A participant object TOₐ and an offstage object TOₐₑ
2.1.1 Offstage objects

Within a family TOS allows the creation of a new temporal object from a set of existing temporal objects. We refer to such a new temporal object as an offstage object and the existing objects as participant objects. In Figure 6 the temporal object TO_a,e is an offstage object and the stage S_k,a of the temporal object TO_a is referred to as an offstage. The temporal object TO_a is the participant object of the offstage object TO_a,e. The offstage S_k,a is the stage from where the offstage object TO_a,e starts sharing the knowledge of the participant object TO_a. In other words, the offstage object TO_a,e has taken its birth at time instance t_1,a,e by sharing the set of stages \{S_k,a, S_{k-1,a}, ..., S_1,a\} of the participant object TO_a. Note that the temporal object TO_a and offstage object TO_a,e have different object identities.

The set of time-spans of stages and life-span of the offstage object TO_a,e are as follows:

\[ T_{a,e} = \{t_{1,a,e}, t_{2,a,e}, t_{3,a,e}, ..., t_{j,a,e}, now\} \]

\[ \text{life-span}_{a,e} = [t_{1,a,e}, now] \]

Figure 5. An example of offstage object "Eng-Maths" in the "Course" Family

Figure 5 and Figure 6 exhibit an offstage object "Eng-Maths" which is constructed in the "Course" Family. The offstage object "Eng-Maths" is defined at the time instance 1965 by sharing knowledge of two participant objects "Statistics-2" and "Advance-Math" through the offstages S_1,Statistics-2 and S_1,Advance-Maths, respectively. Note that these offstages are also birth stages of the participant objects. Each participant objects has one stage (see Figure 5) at the time instance when the offstage object is defined. Figure 6(a) shows the ROF of the family "Course". The stages S_1,Statistics-2 and S_1,Advance-Maths of the participant objects are shown in Figure 6(b) and Figure 6(c), respectively. Figure 6(d) shows the birth stage of the offstage object "Eng-Maths". A temporal condition (Eng-Maths.1975 ≥ Statistics-2.1970) and (Eng-Maths.1975 ≥ Advance-Maths.1969) must hold before defining the offstage object, where 1970 and 1969 are the time instances when the offstages of the participant objects are defined.

If an offstage object shares knowledge of only one participant object, then this is analogous to simple inheritance in the class-based approach, and if it shares knowledge from more than one participant objects, then this is similar to the concept of multiple inheritance in the class-based approach. An offstage object has some similarities and differences with an object which is defined by using the class-based approach.

In TOS two types of families, simple families and complex families, can be defined. They are described in the next two sections.

2.2 Families of the TOS

The concept of a family is used to assemble a group of temporal objects sharing a common context. For example, various Engine temporal objects form an Engine family. All temporal objects within a family can be handled in a similar fashion by responding uniformly to a set of messages. A set of similar structures and/or states defines a common context of a family. The common context of a family is referred to as the root-of-family (ROF) where common knowledge about all its temporal objects is maintained (see 4.5 for more details). Temporal objects of a family can be defined only after the construction of the ROF of the family.

In the class-based object-oriented systems class is used to assemble a set of objects which share some common assets as it is with the definition of family in TOS. However, a family encapsulates more features than a class. For example, in a class, the structure of the class is always shared by all its states (or instances), and a change in its structure affects all states, subclass structures and their corresponding states. Also, if one deletes Color instance variable from the class Vehicle of Figure 1, then all its instances are going to delete data value of the instance variable Color. In a family, however, the structure or state of each temporal object of the family shares the ROF only at the time instance of its birth. After that each temporal object is independent and a change in a particular temporal object does not effect the ROF or any other objects of the family. In other words, the ROF of a family is read-only, it does not change with passage of time. Time is associated with a temporal object and ROF of family.

In TOS two types of families, simple families and complex families, can be defined. They are described in the next two sections.

2.2.1 Simple families

A simple family represents an independent object development environment in which temporal objects can be constructed without sharing any knowledge of other families. For example, in Figure 5, the family "Course" is a simple family. Two simple families do not share any knowledge, and in term of knowledge sharing they are mutually disjoint. A simple family is analogous to a class in the class-based approach, which has no super-class (expect system class or root class).
2.2.2. Complex families

In existing class-based object-oriented systems, a complex object is defined as an object which can have another object as the value of a particular instance variable. For example, Figure 7 shows the structure of complex object Vehicle. Here, we extend our definition of family to complex family which provides a facility for the integration of nonhomogeneous temporal objects of different families in order to build another temporal object which is referred to as temporal complex object (TCO). The components of a TCO are temporal objects of nonhomogeneous families (or independent families), and the temporal objects which take part in the construction of a TCO are called subobjects (or components) of the TCO.

(a) The ROF the family "Course"

(b) The participant object "Statistics-2"

(c) The offstage object "Eng-Maths"

(d) The participant object "Advance-Maths"

Figure 6. The details of the offstage objects and it's participant objects

Figure 7. The TCO Vehicle-80 and its subobjects

A new TCO, TCO\textsubscript{o}, can be defined in a family \textit{F} at a given time instance \textit{t}_{1,o} with \textit{r} subobjects from \textit{r} different simple families. The birth stage of TCO, TCO\textsubscript{o}, may be created at time instance \textit{t}_{1,o} if the temporal condition (\textit{t}_{1,k} \leq \textit{t}_{1,o} ) \land (\textit{t}_{p,k} \leq \textit{t}_{1,o} ) is true for all \textit{k} such that 1 \leq \textit{k} \leq \textit{r}, where \textit{t}_{p,k} is the time instance when the complex family \textit{F}_k was created, and \textit{t}_{1,k} is the time instance when birth stage of the \textit{k}-th subobject was created. This temporal condition ensures that all temporal subobjects and the complex family exist before the existence of TCO. Figure 7 shows the birth stage of the complex family Vehicle which is an aggregation of three simple families Engine, Body and Wheel. In this figure the TCO Vehicle-80 is created at time instance 1979 (denoted by Vehicle-80.1979) if the temporal condition, (ENG-66.1965 \leq Vehicle-80.1979) \land (Body-70.1968 \leq Vehicle-80.1979) \land (Wheel-75.1976 \leq Vehicle-80.1979) is true, where ENG-66.1965, Body-70.1968 and Wheel-75.1976 are the time instances when the subobjects ENG-66, Body-70 and Wheel-75 are created respectively, and Vehicle.1970 is the time instance when the family Vehicle is created. A TCO has all temporal parameters, time-span, life-span, and life-sequence, like the temporal objects of the simple family. ROF of the family Vehicle at a
time instance 1970 can be defined as follows:

\[ R(O_F(Vehicle)) \]

Aggregation-of: (Engine, Body, Wheel)

Instance-variables: \{ time=1970, model#, year-of-model, net-weight \}

Methods: \{ assemble, test-it \}

Note that the concept of an offstage object enhances reusability within a family, and the concept of a TCO enhances reusability within a TOS.

Within the boundary of a simple family, we use the offsprin technique and among the families we prefer the copying technique for knowledge sharing\(^2\). The aggregation and integration of temporal objects into a TSO can generate certain conflicts and compatibility problems such as naming and scaling between a TCO and its subobjects. For example, Naming conflicts occur when two or more subobjects of a TCO contain instance variables or methods with the same name such as the instance variable "weight" which has been defined in subobjects Engine and Body as well as the TCO Vehicle. We are currently investigating these issues.

2.3 Time Dimension in TOS

A time dimension is associated with the creation of a stage, a temporal object, and a family in TOS. The time is explicitly defined by an instance variable. While introducing a new family, data type of the time is defined and it is followed in creation of a temporal object and its stages. The granularity of time depends on the application domain. In TOS, we use time point model for creation of families, temporal objects and stages. A time point is referred to as a time instance. A time instance is a distinct and discrete point on the time-line and a dimension less entity. Time interval is also used in TOS to represent time duration between two time instances. For example, time-span of a stage and life-span of a temporal object. In this paper we use an abstract time "year" in each stage, temporal object and family for the sake of simplicity.

2.4 T02L: A Query Language for TOS

In\(^11\),\(^12\) we proposed a temporal object-oriented query language (T02L) for the TOS. The query Language T02L is a superset of SQL. The proposed language can answer the temporal queries on families (simple and complex). The T02L uses a set of logical operators \{ =, >, <=, => \} and a set of temporal operators \{ Before, After, During, Equivalent, Adjacent, Overlap, Follows, Precedes \} of the SQL and TSQL, respectively\(^9\). The TSQL also adds a new clause WHEN. This clause evaluates temporal predicates by checking relative chronological ordering of timings in a relation. We also use this clause but in our case it works on families and temporal objects. We introduce a parameter Time which is time instance variable in ROF of family and each stage of temporal object. When we mention in a query Time During \([a, b] \), it means that value of instance variable Time lies in the time interval \([a, b] \).

The T02L can answer both temporal and non-temporal queries for TOS. A temporal query has a WHEN clause which contains time, whereas, a non-temporal query does not include this clause. Non-temporally TOS gives a latest and aggregated view of the objects and families. As a temporal query includes a time parameter in WHEN clause, therefore its search space for temporal object and family is time dependent. On the other hand a non-temporal query does not include any time parameter, therefore its search space is complete life-spans of temporal object and family. In general, we can say that search space for a non-temporal query is greater than search space for a temporal query.

The following is an example of a query on simple family Engine.

Q1: 'List all 6 cylinder Engines from 1965 to 1980'.

SELECT Engine
FROM Engine Family
WHERE # of pistons = 6
WHEN Time During [1965, 1980]

The following is an example of a query on an offstage object.

Q2: 'List the most recent outline of the course Eng-Maths'.

SELECT outline
FROM Course Family
WHERE Course=Eng-Maths

3. RENOVATION OF OFFSTAGE OBJECTS

During its life-span, an offstage object may go through changes. These changes may occur to its participant objects. For example, in the offstage object "Eng-Maths" the course "Advance-Maths" at time instance 1969 was used as the participant object at a time instance 1975 when the offstage object was originally designed. At some later time instance, "Advance-Maths" at time instance 1969 may be replaced by an updated version, say course "Advance-Maths" at time instance 1977 (see Figure 8 and Figure 9), or added another course "Complex Analysis" (see Figure 10) in the offstage object "Eng-Maths". This replacement or addition in the offstage object is referred to as the renovation of the offstage object. In this paper we consider two types of renovations, and refer to them as Type-I and Type-II renovations. These two types of renovations are described in the next two sections.

3.1 Type I Renovation

In the first type of renovation, an offstage object is updated automatically if it goes through changes. For example, in the offstage object "Eng-Maths" one stage (only birth stage) of the participant object "Advance-Maths" was shared while defining the offstage object. At some later time when the object "Advance-Maths" goes through a stature change, then this change (i.e., Bessel Functions and Bessel Trans. are added in the instance variable outlines at time instance 1977 as shown in Figure 8) is also needed to be recorded in the offstage object "Eng-Maths". This renovation of the offstage object "Eng-Maths" is performed at the time instance 1988 by defining a new stage at that time instance as shown in Figure 9. The renovation will occur if the temporal
condition \((\text{Advance-Maths.1977} \leq 1988) \land (\text{Eng-Maths.1975} \leq 1988)\) is true, where \text{Advance-Maths.1977} is the time instance when the current stage of the participant object "Advance-Maths" is defined and 1988 is the time instance when the offstage object is renovated. If more than one participant object participates in renovation of an offstage object, then all participant objects must satisfy the temporal condition. The double arrow shows re-sharing (or renovated) from the participant object "Advance-Maths" to the offstage object at the time instance 1988. Also, the offstage object must satisfy a temporal condition which ensures the existence of the new participant object or new version of the participant object (or participant objects) before renovation of the offstage object.

Before \((\text{Before} \leq 1988)\) renovation the offstage object "Eng-Maths" was sharing only one stage (birth stage) of the participant object "Advance-Maths" (see Figure 9), and it is shown by a single dark arrow in Figure 5. After \((\text{After} < 1988)\) the renovation, the offstage object is sharing two stages of the participant object (see Figure 9). The participant object "Advance-Maths" is shared two times in the offstage object at the time instances 1969 and 1988. The life-sequence of the participant object was different at different time instances. The life-sequence of the participant object "Advance-Maths" is \(\{S_1, \text{Advance-Maths}\}\) and \(\{S_1, \text{Advance-Maths}, S_2, \text{Advance-Maths}\}\) at time instances 1969 and 1988, respectively. In type I of renovation, object identities of the participant objects remain the same.

3.2 Type II Renovation

In type-II renovation, a participant object of an offstage object is replaced by a new participant object of the family at the same time instance. The new participant object will be completely compatible with the old participant object, that is, the new participant object and old participant object perform the same function. For example, if a new participant object, "Complex Analysis", is added to the offstage object (see Figure 10) then both participant objects ("Advance-Maths" and "Complex Analysis") will have different object identities as they are two different temporal objects of the family "Course".

The type-II renovation can be performed on an offstage object when an existing participant object of the offspring object is replaced by a new and different participant object of the same family. Note that the temporal condition should hold before the renovation can take place.

Figure 9. Renovation of Type I of the offstage object "Eng-Maths"

Before (\(\text{Before} \leq 1988\)) renovation the offstage object "Eng-Maths" was sharing only one stage (birth stage) of the participant object "Advance-Maths" (see Figure 9), and it is shown by a single dark arrow in Figure 5. After (\(\text{After} < 1988\)) the renovation, the offstage object is sharing two stages of the participant object (see Figure 9). The participant object "Advance-Maths" is shared two times in the offstage object at the time instances 1969 and 1988. The life-sequence of the participant object was different at different time instances. The life-sequence of the participant object "Advance-Maths" is \(\{S_1, \text{Advance-Maths}\}\) and \(\{S_1, \text{Advance-Maths}, S_2, \text{Advance-Maths}\}\) at time instances 1969 and 1988, respectively. In type I of renovation, object identities of the participant objects remain the same.

3.2 Type II Renovation

In type-II renovation, a participant object of an offstage object is replaced by a new participant object of the family at the same time instance. The new participant object will be completely compatible with the old participant object, that is, the new participant object and old participant object perform the same function. For example, if a new participant object, "Complex Analysis", is added to the offstage object (see Figure 10) then both participant objects ("Advance-Maths" and "Complex Analysis") will have different object identities as they are two different temporal objects of the family "Course".

The type-II renovation can be performed on an offstage object when an existing participant object of the offspring object is replaced by a new and different participant object of the same family. Note that the temporal condition should hold before the renovation can take place.

Figure 10. Renovation of Type II of the offstage object "Eng-Maths"

4. CONCLUSIONS

In this paper we discussed the Temporal Object System (TOS) which maintains changes in both the structure and the state of an object. An object in a TOS is referred to as a temporal object which evolves over time. Changes to both the structure and the state of the object are maintained throughout its various stages. In this paper we showed how the aggregation of temporal objects (participant objects) in a family allows the construction of an offstage object. Renovation of an object improve reusability in a family. We discussed two types of renovations for offstage objects. In the first type of renovation an offstage object changes when one or more of its participant objects change. For the second type of renovation, an offstage object changes when it tries to use a different participant object from the same family. Currently we are working on implementation of such a system and also improving on its query language.

REFERENCES


APPENDIX

The following set of figurative notations are used to represent different entities of the TOS.

- A stage of a temporal object
- A root-of-family (ROF)
- Root of the TOS (RTOS)
- Transition to the next stage
- Temporal inheritance
- Renovation