Knowledge Sharing Mechanism of the TOS

Abad A. Shah[1], Farshad Fotouhi[2], and William Grosky[2]

[1]Computer Science Department, King Saud University, P.O. Box 51178, Riyadh 11543, Saudi Arabia

[2]Computer Science Department, Wayne State University, Detroit Michigan, 48202 USA

Email: F60C009 at SAKSU00.BITNET

Abstract

In [5], [10] we proposed TOS: A Temporal Object System that handles the structure and stature changes in an object in a uniform and temporal fashion. The TOS takes a hybrid approach of c class-based and prototype-based approaches that makes it more flexible than other two approaches.

In this paper, we propose a formal model for knowledge sharing mechanism (Share-kno) for hierarchy of objects of the TOS, and formally prove that the mechanism Share-kno encapsulate more knowledge than inheritance and delegation models which are proposed by Stein in [12].

Key Words: object-oriented database, temporal object, knowledge sharing mechanism, temporal databases

1. Introduction

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 and methods. The *state* (ST) of an object assigns data values to the instance variables of the objects, and methods which operate on them. In the object-oriented paradigm there are two techniques class-based and prototype-based, to represent knowledge of objects [3],[7]. The first technique is based on the mathematical concept of *set*, 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 directed acyclic graph (DAG). The second technique is *prototyping* that represents an object by its *default knowledge* [3],[7],[12]. The prototype-based technique is a classless approach where all objects are at same level. The class-based technique considers structure and state of an object as two objects in their own right [7]. These techniques use *inheritance* and *delegation* mechanisms, respectively, for knowledge sharing in a hierarchy of objects. The delegation mechanism is commonly considered more powerful than the inheritance mechanism [7]. But Stein formally proved that both mechanisms are equivalent in power [12].

In the existing object-oriented database systems, changes in the state of an object are maintained via *version management* [1]. Also, structural changes are supported in most object-oriented database systems. Such changes to a class are referred to as *schema evolution* in the literature [8]. Current object-oriented database systems keep only the current version of each class structure. *After* any change, it is necessary to reload a previous version of the database to retrieve any information from the previous version of a class structure.

In [5],[10] we introduced a temporal object 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/or 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*. (for details see [11]). A complex family enhances the knowledge sharing among non-homogeneous temporal objects and their transportability. A temporal object system (TOS) is a collection of families which are defined at different time instances.

In this paper, we propose a formal model of Share-kno (SK) for a hierarchy of a TOS, and formally prove that the SK model includes both the inheritance and delegation models which are proposed by Stein [12]. The remainder of this paper is organized as follows: In Section 2, we describe TOS. Section 3 discusses inheritance and delegation models. In Section 4 we give a formal model of Share-kno (SK) of TOS. In Section 5 we prove the inclusion of inheritance and delegation models in SK model. Finally, in Section 6 we give our concluding remarks and future research directions.

2. Temporal Object System

As mentioned in the previous section Temporal Object System (TOS) is defined as a collection of 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 [5],[10]. Figure 1 shows a schema of the TOS, where RTOS represents the root node of the system with n families, i.e., $F_1, F_2, \ldots, F_n$ as its children.

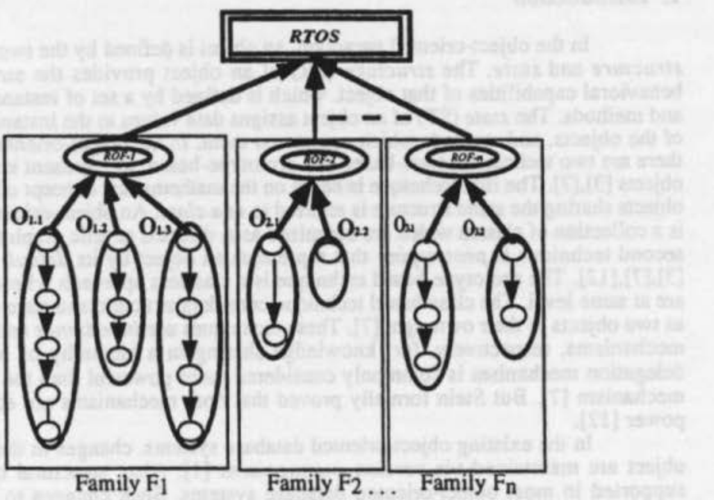

Figure 1. Schema of the temporal object system

In the figures, double rectangle, double oval, rectangle, single oval, circle represent RTOS, root-of-family (ROF), family, temporal object, and stage, respectively. Single arrow represents a structure change or state change to a temporal object. The terms ROF, family, and stage are formally defined in the next section.

2.1 Temporal Objects and their Families

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. 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 = $< (SR_{t_1}, ST_{t_1}), (SR_{t_2}, ST_{t_2}), \ldots, (SR_{t_n}, ST_{t_n}) >$ where $t_i \leq t_{i+1}$ for all $1 \leq i < n$, where the ordered pair (SR_{t_i}, ST_{t_i}) is the i-th object of the temporal object which is constructed at the time instance t_i with structure SR_{t_i} and state ST_{t_i}. An i-th object of the temporal object is referred to as its i-th stage [5],[6].

Figure2. A temporal object TO_a

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 structure 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 2 the temporal object TO_a of the family F_i 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 $S_{1.a}$ in Figure 2) and the *current stage* (stage $S_{n.a}$ in Figure 2) of the temporal object. A new stage is appended to a temporal object if the structure and/or state associated with its stage changes (see [5],[10] for more details).

The concept of a family is used to assemble a group of temporal objects sharing a common context. 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 [6],[10],[11] for more details). Temporal objects of a family can be defined only *after* the construction of the *ROF* of the family. In a family, 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 affect the *ROF*. The *ROF* of a family is *read-only*, it does not change with passage of time.

In TOS two types of families, *simple families* and *complex families*, can be defined. A *simple family* represents an independent object development environment in which temporal objects can be constructed without sharing any knowledge of other families. A *complex family* provides a facility for the integration of non homogeneous temporal objects of different families in order to build another temporal object of a higher level of abstraction which is referred to as *temporal complex object* (TCO) [6], [11].

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 user as an instance variable. 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*.

3. Existing Knowledge Sharing Mechanisms

As described before, the object-oriented paradigm has inheritance and delegation knowledge sharing mechanisms. Stein proposes two formal models for each mechanism, and simulated one model into another [12] to prove that the both mechanisms are equivalent in term of power. His models deal with only simple inheritance and single-parent delegation, details can be seen in [12]. In the following paragraph we briefly describe his inheritance model for a hierarchy [12].

The inheritance hierarchy I of a system is defined as follows:

I = {C, I, Y, W} where
C: is a set of classes in a class-hierarchy,
I: is a set of instances,
Y: is a set of attributes (instance variables and methods), and
W: is a set of attribute values in Y.

Each class structure has two sub-structures: locally defined substructure and substructure which is inherited from its super-class. The substructure that is defined locally in a class is referred to as *class-attributes*. The substructure that is inherited from a super-class is referred to as *instance-template*. In the model I, structure of each object of a class $c \in C$ is defined by *instance-template* and *class-attributes* where *class-attributes* are inherited from its superclass *super*(c). The *attributes* of a class c are *attributes*(c) = *class-attributes*(c) $\cup$ *instance-template*(c). The values of the *attributes* are taken from the set $V(c) = \{val_c(y) \mid y \in attributes(c)\}$ where $c \in C$. An instance i $\in$ I of a class c is defined by two sets of *attributes* (i) = *class*(i) which are *value* V(i).

Similarly, a model for delegation model **D** for a hierarchy is proposed by Stein and can be seen in [12].

4. Share-kno Model

In this section, we formally define Share-kno (**SK**) that is knowledge sharing mechanism of the TOS.

A model of a systems is defined as an abstraction of the system in order to understand the system before building it [4], [9]. For example, in [4], a model for a language **L** is defined by a pair <A, T> where A is its *universe* which is a non-empty set, and T is a set of *relations*. We refer to the pair <A, T> as *elements* of the model **L**.

We say a model A is *more knowledgeable* (or super model) than a model B *iff* B is a *sub-model* of A i.e., $B \subseteq A$ (or $B \Rightarrow A$). In other words, one or more elements of the model B are subset of the corresponding elements of the model A.

We propose a model **SK** for a hierarchy of temporal objects of TOS to share knowledge among its objects. Since Stein's inheritance and delegation models support simple inheritance and single-parent delegation, receptively, therefore our model **SK** also confines to only simple families to make **SK** model compatible with the Stein's models. In defining **SK** model and proving theorems we use the same terminology which is used by Stein in [12] for his models.

The **SK** model for a hierarchy of a TOS is defined as follows:

$$\mathbf{SK} = \{F, TO, S\}$$

where F is a finite set of families and the set F is a union of *ROF*'s of simple families, TO is a set of temporal objects, and S is a set of stages in the TOS. The sets F, TO and S are referred to as *elements* of the model **SK**.

Basic axiom of **SK** hierarchy of the TOS are defined as follows:

Axiom: The set S of stages of a family can be partitioned into three subsets as follows:

$S = S_{sr} \cup S_{st} \cup S_{sr+st}$ where

S_{sr} : a set of stages which is defined due to the structural changes to temporal objects of the TOS,

S_{st} : a set of stages which is defined due to the stature changes to temporal objects of the TOS,

S_{sr+st} : a set of stages which is defined due to both structural and stature changes simultaneously to temporal objects of the TOS. The three subsets of the set S are mutually disjoint, i.e., $S_{sr} \cap S_{st} \cap S_{sr+st} = \phi$.

As each temporal object $TO_i \in TO$ is defined in a family $f \in F$ as a set of temporally ordered stages. The j-th stage $S_{j,i} \in S$ of the temporal object TO_i may be a member of the set S_{sr}, S_{st}, or S_{sr+st} based upon the type the stage. To prove that the I model of a hierarchy is a *sub-model* of the **SK** model of the same hierarchy, we prove that for each object in the I model there exists an object in the **SK** model, but the object (temporal object) in the **SK** model may not be completely contained by the I model. We assume that a time instance t_1, we model two versions of the object simultaneously in the inheritance model I and a Share-kno model **SK**. Then, at a time instance $t_2 > t_1$, the **SK** model is *more knowledgeable* (or super-model) than the model I with respect to knowledge of objects.

In a model I of a hierarchy, at a time instance $t_2 > t_1$ the structure of the object represents an accumulative affect of all structural changes which have occurred during the time interval $[t_1, t_2]$. In this model, details of the structural changes to an object are not traceable. Also, if both structural and stature changes occur simultaneously to an object, then a model I is not capable of handling all such changes at a time instance. On other hand, a **SK** model can handle all such changes to an object and the set S_{sr+st} represents such type of changes.

Definition: A model I is a *sub-model* (or less knowledgeable) of a model **SK** which is written as $I \Rightarrow \mathbf{SK}$ *iff* for an object TO in a **SK** model, if there exists at least one change (stage) to the object TO which does not corresponds to any change to the object O in an I model.

5. Proof of $\sigma(I) \Rightarrow SK$

Now we define a function σ which takes an inheritance model I = {C, I, Y, W} of a hierarchy, and formally prove that it is a sub-model (or *less knowledgeable*) of a model **SK** of the same hierarchy at some time instance i.e., $\sigma(I) \Rightarrow$ **SK**. The function σ is defined as follows:

(i) $\sigma(\phi) = \phi$

(ii) $\forall x \in Y, \sigma(x) \in S_{sr} \vee \sigma(x) \in F$

(iii) $\forall u \in W, \sigma(u) \in S_{st}$

(iv) $\forall c \in C, \sigma(c)$ is defined as follows:

$\sigma(c) \subseteq f$ where $f \in F$ and σ (*attributes*(c)) = $\{\cup (\sigma(x) \mid x \in$ *class-attributes*(c)$\} \cup \{\cup (x) \mid x \in$ *instance-template*(c)$\}$. Here $\{\cup (\sigma(x) \mid x \in$ *class-attributes*(c)$\} \subseteq S^f_{sr} \cup f$, if x does not belong to current stage of any temporal object, and $\{\cup \sigma(x)\ x \in$ *instance-template*(c)$\} \subseteq S^f_{sr}$ if x belongs to current stage, $S^f_{sr} \subseteq S_{sr}$ and the subsets S^f_{sr} belongs to a specific family f. Note that *attributes*(c) is union of the sets *class-attributes*(c) and *instance-template*(c).

σ(*state* (c)) = $\{\cup \sigma(v) \mid v \in$ *state*(c)$\} \subseteq S^f_{st}$ where $\forall x \in$ *attributes*(σ(c))

$val_{\sigma(c)}(x) = val_c(x)$, and $S^f_{st} \subseteq S_{st}$. The set S^f_{st} belongs to a specific family f.

(v) $\forall i \in I, \sigma(i)$ is defined as $\sigma(i) = \sigma$ (*class*(i)) $\subseteq (TO_j)_f$ where $(TO_j)_f$ represents the j-th temporal object of the family f in the **SK** model. Here *class*(i) = *attributes*(*class*(i)) $\cup$ V(*class*(i)). *State* of the temporal object $(TO_j)_f$ corresponds to the i-th instance in a model I. σ(*attribute*(*class*(i))) = $\{\cup \sigma(x) \mid x \in$ *class-attributes*(c) $\wedge$ c $\in$ *super*(*class*(i))$\} \subseteq \cup S^f_{i,j}$ for all $1 \le i \le n$, where each member of the set $\cup S^f_{i,j}$ belongs to the set S^f_{sr}.

σ(*state* (*class*(i))) = $\{\sigma(v) \mid v \in$ *state* (i)$\} \subseteq \cup S^f_{k,j}$ for all $1 \le k \le m$, where each member of the set $\cup S^f_{k,j}$ belongs to the set S^f_{st}.

Assume there are p number of stages of the temporal object $(TO_j)_f$ belong to the set S^f_{sr+st}, and the instance $i \in I$ is unable to simulate those p number of stages, because the model I of the hierarchy does not support a simultaneous structural and stature change to an object. Therefore, an instance i in the model I can map at the most (m+n) stages out of (m+n+p) stages of the temporal object $(TO_j)_f$, and I model is unable to map p number of stages as an I model has no provision to handle a combined structural and stature change to an object at a time instance.

(vi) By using the function σ, we can show that some *elements* of a model I are subsets of some *elements* of a model SK, i.e., $\sigma(I) \Rightarrow$ **SK** where

$(S_{st} \cup S_{sr} \cup S_{sr+st} \cup F) \supseteq C$ as $(S_{st} \cup S_{sr} \cup F) = C$ ....(1)

$TO \supseteq \sigma(c) \cup \{\sigma(i) \mid i \in I\}$ ....(2)

$(S_{st} \cup S_{sr+st}) \supseteq I$ as $S_{st} = I$, it is true from (1) ....(3)

$(S_{sr} \cup S_{sr+st} \cup F) \supseteq Y$ as $(S_{sr} \cup F) = Y$ ....(4)

$(S_{st} \cup S_{sr+st}) \supseteq W$ as $S_{st} = W$ ....(5)

Theorem: Assume an object is defined simultaneously in the both models I and **SK** of the same system at a time instance t_1. The first version (non-temporal version $O_i \in c \in C$) of the object is in a model, I, and the second version (temporal version $TO_i \in f \in F$) of the object is in a **SK** model. If both versions of the object (O_i and TO_i) represent the same entity, then the temporal version TO_i of the object is *more or equally knowledgeable* than the non-temporal version O_i of the object at a time instance $t_2 > t_1$, that is, *attribute*(O_i) $\subseteq$ *structure*(TO_i) and *state*(O_i) $\subseteq$ *state*(TO_i) at any time instance t_2.

Proof: At the time instance t_1 both versions (O_i and TO_i) take birth in their respective systems. Therefore, at the time instance t_1 the following inequalities are true:

(*attributes*(O_i) = *structure*(TO_i)) and (*state*(O_i) = *state*(TO_i)). The class structure of the O_i, and *attributes*(O_i) = $\{x \mid x \in$ *class-attributes*(c), $O_i \in c\} \cup \{y \mid y \in$ *instance-template*(c)$\}$ where *attributes*(O_i) $\subseteq Y' \subseteq Y$.

structure(TO_i) = $\{x \mid x \in S_{sr}\} \cup \{ y \mid y \in f, TO_i \in f\} \cup \{z \mid z \in S_{sr+st}\}$ where *structure*(TO_i) $\subseteq (S_{sr} \cup S_{sr+st} \cup f)$ and from (1) and (4) $Y' \subseteq (S_{sr} \cup S_{sr+st} \cup f)$.

state(O_i) = $V(O_i)$ = $\{$*val* $_{O_i \, \varepsilon \, c}(x) \mid x \in$ *attributes*(O_i)$\} \subseteq W' \subseteq W$ and

state(TO_i) = $\{ x \mid x \in S_{st}\} \cup \{y \mid y \in S_{sr+st}\} \subseteq (S_{st} \cup S_{sr+st})$.

From properties (4) and (5) $W' \subseteq (S_{st} \cup S_{sr+st})$. Hence, at the time instance t_1, knowledge in both models about the object is equal.

Now assume that the time instance $t_2 > t_1$ the object has gone under some type of change. The types of changes are as follows:

(1) An object suffers a change only to its state at the time instance t_2. It means (*attribute*(O_i) = *structure*(TO_i)) and (*state*(O_i) $\subseteq$ *state*(TO_i)) where *state*(O_i) represents a new version of the state of the object O_i in the I model, and *state*(TO_i) $\subseteq S_{st}$, where the temporal object $TO_i = \{S_{1,i} \; S_{2,i}\}$ at time instance t_2, and $S_{2,i} \in S_{st}$ is current stage of the temporal object TO_i. The current stage reflects an update of stature change to the object. Hence, both versions of the objects, O_i and TO_i, are *equally knowledgeable* at the time instance t_2 in both models, as there is no loss of knowledge about the object in both model.

(2) An object suffers a change to its structure only at the time instance t_2. This type of change can further be subdivided into two types as follows:

(i) An instance variable and/or a method is added to an object. Due to addition of an instance variable and/or a method there is no loss of knowledge in both models. So, at the time instance t_2, *attribute*(O_i) = *structure*(TO_i) and *state*(O_i) = *state*(TO_i) where *attribute*(O_i) is an updated version of the *class-attributes*, *structure*(TO_i) $\subseteq S_{sr}$, and *state*(TO_i) $\subseteq S_{st}$. Here cardinality of the *class-attributes* at the time instance t_1 is less than the cardinality of the *class-attributes* at the time instance t_2 in the model I. The cardinality of the *tructure*(TO_i) at the time instance t_1 is also less than the cardinality of the *structure*(TO_i) at the time instance t_2 in the **SK** model. But the **SK** model keeps both previous and current knowledge of each temporal object, and the I model keeps only an updated version of a *class-attributes* without keeping a history of changes to the *class-attributes*. We conclude that in this case, a TO_i in a

SK model is *more knowledgeable* than an object O_i in a model I at the time instance t_2, as the previous *class-attributes* of attribute(O_i) is lost *after* the update.

(ii) An instance variable and/or a method is deleted from an object. In this case, previous knowledge of the object in the model I is lost due to deletion and the cardinality of the *class-attributes* at time instance t_1 is greater than cardinality of the *class-attributes* at the time instance t_2. As the model **SK** keeps history of a change to the object *before* and *after* update of the change even the cardinality of the *structure*(TO_i) *before* the update is greater than cardinality of the *structure* (TO_i) *after* the update. So, at and after the time instance t_2, *attribute* (O_i) $\subseteq$ *structure*(TO_i) $\wedge$ *state*(O_i) $\subseteq$ *state*(TO_i), because the deleted instance variable and method are not present in the *attribute*(O_i) at the time instance t_2. It means that the temporal version TO_i, of the object retains both *present* and *past* knowledge of the object. Hence we conclude that a TO_i in a **SK** model is *more knowledgeable* than an O_i in an I model.

(3) In the third type of change when the above two types (i and ii) of changes occur together to the object at the time instance t_2, then the I model is unable to update itself after the change. Whereas, the **SK** model can handle update the object *after* the change by creating a new stage which is a member of the set S_{sr+st}. *After* update of the change *structure*(TO_i) $\subseteq S_{sr} \cup S_{sr+st} \cup f$. Since an I model is incapable to handle such changes, whereas a **SK** model is quite capable and it keeps history of the changes.

Therefore, we conclude that a **SK** model is *more or equally knowledgeable* than an I model at the time instance t_2. Hence, from the above results we conclude that the **SK** model is *more or equally knowledgeable* than the I model at time instance t_2 (i.e.,$\forall$ i $O_i \subseteq TO_i$ where $O_i \in c \in C$ and $TO_i \in f \in F$).

6. Conclusions

In this paper, we proposed a formal model for knowledge sharing mechanism, Share-kno, of the TOS. The proposed model takes a hybrid mechanism of the inheritance and delegation mechanisms. The mechanism, Share-kno, is capable to accommodate more knowledge of temporal objects than inheritance and delegation mechanisms, and to share them with other temporal objects of the same simple family. We formally proved that Share-kno super-model of the inheritance and delegation models proposed by Stein.

References

[1] R. Agrawal et al, "Object Versioning in Ode, " *Proc. of Data Engineering*, 1991.

[2] A.M. Alashqur, S.Y.W. Su, and H. Lam, "OQL: A Query Language for Manipulating Object-Oriented Databases". *Proc. of 15th Intl. Conf. on VLDB*, 1989.

[3] A.H. Borning,"Classes Versus Prototypes in Object-Oriented Languages, "*ACM/IEEE Fall Joint Conf.*, November, 1986.

[4] C. C. Chang and H. J. Keisler, "Model Theory, "*Third Edition, North-Holland*, 1990.

[5] F. Fotouhi, A. Shah and W. Grosky, "TOS:A Temporal Object System, "*the 4th Intl. Conf. on Computing. & Info.*, Toronto, Canada, 1992.

[6] F. Fotuhi, A. Shah and W. Grosky, "Complex Objects in the Temporal Object System, "*IEEE Post-Proceeding of the 4-th International Conference on Computing & Information*, 1992.

[7] H. Lieberman, "Using Prototypical Objects to Implementation Shared Behavior in Object-Oriented Systems, " *Proc. of ACM OOPSLA*, 1986.

[8] G.T. Nguyen and D. Rieu, "Schema Evolution in Object-Oriented Database Systems, "*Data & Knowledge Engineering Journal, North-Holland*, No.4, 1989.

[9] J. Rumb'ugh, et al, "Object-Oriented Modeling and Design, " *Prentice-Hall*, 1991.

[10] Abad A. Shah, "TOS: A Temporal Object System, " *Ph.D. Dissertation*, Wayne State University, Detroit, Michigan, 1992.

[11] Abad A. Shah, F. Fotouhi, and W. Grosky, " Renovation of Complex Objects in the Temporal Object System, " *IEEE Intl. Phoenix Conf. on Computers and Communications*, Scottsdale, Arizona, 1993.

[12] L.A. Stein, "Delegation Is Inheritance, "*Proc. of ACM Conf. OOPSLA*, 1987.