QoS-Based Compensation of Multimedia Objects *

1X－1

Motokazu Yokoyama, Katsuya Tanaka, and Makoto Takizawa †
Tokyo Denki University ‡
Email : {moto, katsu, taki}@takilab.k.dendai.ac.jp

Abstract

In distributed applications, QoS of a multimedia object is manipulated in addition to the state. While objects are manipulated through methods, the manipulations on the objects have to be undone in designing multimedia systems and recovering from the system fault. In this paper, we discuss how methods performed are compensated by other methods. Novel types of compensating methods are defined to obtain a state and QoS of the object which satisfy requirements. We discuss how to find a cheaper way to compensate a sequence of methods.

1 Introduction

Distributed applications are composed of multimedia objects. Here, quality of service (QoS) of a multimedia object is manipulated as well as the state.

In manipulating a multimedia object, an application might like to undo the manipulation, for example, for interactively designing and implementing an application. In another example, an object is rolled back due to the fault of the object. Suppose that an application changes a colored *movie* object to a monochrome one by a method *grayscale* after adding a *red* car by a method *add-car*. Here, the *movie* object is monochrome. Next, suppose the application would like to undo the manipulation done here. According to the traditional ways, the *movie* object is rolled back to the previous one saved at a checkpoint, i.e. colored object without the *car* object. Another way is to compensate a computation sequence of *add-car* and *grayscale* by other methods. *del-car* is a method where a *car* is removed. *color* is a method where a *scene* object is changed to be colored. If *color* is performed after *del-car*, the object is recovered to the previous state. Here, *del-car* and *color* are referred to as *compensating* methods of *add-car* and *grayscale*, respectively. If the application is not interested in how colorful the *movie* object is, only the *car* object can be removed without changing the color. That is, the sequence of methods *add-car* and *grayscale* can be just compensated by one method *del-car* with respect to QoS required by the application.

In section 2, we discuss relations among methods. In section 3, we discuss compensating methods. In section 4, we discuss how to compensate a sequence of methods.

2 QoS-based Relations of Methods

An object-based system is composed of *classes* and *objects*. A class c is composed of *attributes* $A_1, \ldots, A_m$ $(m \geq 0)$ and *methods*. An object o is created from the class c by giving values to attributes. A collection $\langle v_1, \ldots, v_m\rangle$ of values is a *state* of the object o where each v_i is a value taken by A_i $(i = 1, \ldots, m)$.

A class c can be composed of *component* classes $c_1, \ldots, c_n$ in a *part-of* relation. Let $c_i(s)$ denote a projection of a state s of the class c to c_i. A state of an object is changed by performing a method *op*. Let $op(s)$ and $[op(s)]$ denote a state and response obtained by performing a method *op* on a state s of an object o, respectively. "$op_1 \circ op_2$" shows a serial computation of op_1 and op_2.

Applications obtain service of an object o through methods. Each service is characterized by *quality of service* (*QoS*). A QoS *value* is a tuple of values $\langle v_1, \ldots, v_m\rangle$ where each v_i is a value of parameter like frame rate. A QoS *value* q_1 *dominates* another QoS value q_2 $(q_1 \succeq q_2)$ iff q_1 shows a better level of QoS than q_2. For example, $\langle 160 \times 120[\text{pixels}], 1024[\text{colors}], 15[\text{fps}]\rangle \succeq \langle 120 \times 100, 512, 15\rangle$. $q_1 \cup q_2$ and $q_1 \cap q_2$ show least upper bound and greatest lower bound of q_1 and q_2 on $\succeq$, respectively. Let $Q(s)$ be a QoS value of a state s of an object o. $Q(op(s))$ and $Q([op(s)])$ are QoS values of state and output obtained by performing *op*. An application requires an object o to support some QoS, named *requirement* QoS (*RoS*).

Suppose a class c is composed of component classes $c_1, \ldots, c_m$ $(m \geq 0)$. An application specifies whether each component class c_i is either *mandatory* or *optional*. There are the following relations among a pair of states s_t and s_u of a class c:

- s_t is *state-consistent* with s_u $(s_t - s_u)$ iff $s_t = s_u$.
- s_t is *semantically consistent* with s_u $(s_t \equiv s_u)$ iff $s_t - s_u$ or $c_i(s_t) \equiv c_i(s_u)$ for every mandatory component class c_i of c.
- s_t is *QoS-consistent* with s_u $(s_t \approx s_u)$ iff $s_t - s_u$ or s_t and s_u are obtained by degrading QoS of some state s of c, i.e. $Q(s_t) \cup Q(s_u) \preceq Q(s)$.
- s_t is *semantically QoS-consistent* with s_u $(s_t \simeq s_u)$ iff $s_t \approx s_u$ or $c(s_t) \simeq c(s_u)$ for every mandatory component class c_i of c.
- s_t is *r-consistent* with s_u on RoS r $(s_t \approx_r s_u)$ iff $s_t \approx s_u$ and $Q(s_t) \cap Q(s_u) \succeq r$.
- s_t is *semantically r-consistent* with s_u on RoS r $(s_t \equiv_r s_u)$ iff $s_t \approx_r s_u$ or $c_i(s_t) \equiv_r c_i(s_u)$ for every mandatory class c_i of c.

For example, a *movie* class is composed of mandatory classes *car* and *tree* and an optional class *background*. Each state s_i of the *movie* object is composed of *car* c_i, *tree* t_i, and *background* b_i $(i = 1, 2)$. $s_1 \simeq s_2$ if c_1 and c_2 show a same car with different QoS and t_1 and t_2 indicate a same tree with different QoS.

Let $\square_\alpha$ show an α-consistent relation where α shows some consistent relation. For example, $\square_{QoS}$ (or $\square_\approx$) shows "$\approx$". *State, Sem, QoS, R, Sem-QoS,* and *Sem-R* stand for sets of possible *state, semantically, QoS, R, semantically QoS*, and *semantically R* consistent relations on states of a class c, respectively. Here, R is $\{ \square_r \mid r$ is a possible QoS$\}$, and *Sem-R* is $\{ \square_{\equiv_r} \mid r$ is a possible QoS value$\}$. Let C be a family of the sets *state, Sem, QoS, R, Sem-QoS*, and *Sem-R* of consistent relations. A relation "$a \rightarrow b$" for a pair of sets a and b shows that b is a subset of a. That is, $s_t \square_b s_u$ if $s_t \square_a s_u$ for every pair of states s_t and s_u. *State* $\rightarrow$ *Sem*, *State* $\rightarrow$ *R*, *R* $\rightarrow$ *Sem-R*

*QoS に基づくマルチメディアオブジェクトの補償演算
†横山 基一, 田中 勝也, 滝沢 誠
‡東京電機大学

$R \rightarrow QoS$, $QoS \rightarrow$ *Sem-QoS*, *Sem-R* $\rightarrow$ *Sem-QoS* are primitive relations, i.e. not transitive.

Let op_t and op_u be a pair of methods of a class c. "$op_t \sqsubset_\alpha op_u$" shows that $op_t(s) \sqsubset_\alpha op_u(s)$ for every state s of the class c. ϕ shows an empty sequence of methods. $op \sqsubset_\alpha \phi$ iff $op(s) \sqsubset_\alpha s$ for every state s of c. For example, $display - \phi$. Let r_1 and r_2 be a pair of QoS values where $r_1 \succeq r_2$. Here, $\sqsubset_{r_1} \rightarrow \sqsubset_{r_2}$ if $r_1 \succeq r_2$. For example, $s_t \approx_{r_1} s_u$ if $s_t \approx_{r_2} s_u$.

In the traditional theories, a method op_t is *compatible* with another method op_u on a class c iff the result obtained by performing op_t and op_u is independent of the computation order. Otherwise, op_t *conflicts* with op_u.

[**Definition**] For every pair of methods op_t and op_u of a class c, op_t is *α-compatible* with op_u ($op_t \diamond_\alpha op_u$) iff $(op_t \circ op_u) \sqsubset_\alpha (op_u \circ op_t)$ where $\alpha \in C$. □

For example, op_t is *semantically compatible* with op_u ($op_t \,|||\, op_u$) iff $(op_t \circ op_u) \equiv (op_u \circ op_t)$. The "*R-compatible* relation" $\diamond_R$ shows a set $\{\diamond_r | r \in R\}$ where R is a set of possible QoS values. op_t *α-conflicts* with op_u ($op_t \not\diamond_\alpha op_u$) unless $op_t \diamond_\alpha op_u$. Let *State, Sem, QoS, R, Sem-QoS,* and *Sem-R* be sets of possible *state, semantically, QoS, R, semantically QoS,* and *semantically R-compatible* relations on methods of a class c, respectively. $\diamond_\alpha$ is symmetric and transitive.

3 Compensating Methods

In traditional systems, if the system is faulty, the state stored in the log is restored in the system and then the system is restarted. Suppose *paint* is performed on a *background* object. If *erase* is performed, the *background* object can be restored. *erase* is a compensating method of *paint*. Traditionally, a method op_u is a *compensating* method of another method op_t on a class c if $op_t \circ op_u(s) = s$ for every state s of the class c. We extend the compensation concept to multimedia objects.

[**Definition**] A method op_u *α-compensates* another method op_t on an object ($op_u \rhd_\alpha op_t$) with respect to a consistent relation α in C iff $(op_t \circ op_u) \sqsubset_\alpha \phi$. □

Let $(\sim_\alpha op)$ denote an *α-compensating* method of a method op, $op \circ (\sim_\alpha op) \sqsubset_\alpha \phi$.

Let *State, Sem, QoS, R, Sem-QoS,* and *Sem-R* denote sets of possible *state, semantically, QoS, R, semantically QoS*, and *semantically R* compensating relations of methods of a class c. Let CR be a family of these compensating relations, $CR = \{\rhd_\alpha | \alpha \in C\}$.

Suppose $\alpha_1 \rightarrow \alpha_2$ for $\alpha_1, \alpha_2 \in CR$. For example, *Sem* $\rightarrow$ *Sem-R*. This means that op_t *Sem-r*-compensates op_u for RoS r in R ($op_t \rhd_{\equiv_r} op_u$) if $op_t \rhd_\equiv op_u$.

[**Theorem**] If $\alpha_1 \rightarrow \alpha_2$, $op_t \rhd_{\alpha_2} op_u$ if $op_t \rhd_{\alpha_1} op_u$. □

After performing op on a state s of a class c, a state s' is obtained by performing the compensating method $(\sim_{Sem} op)$. $s' \equiv s$. From the theorem, op can be α_2-compensated by $(\sim_{\alpha_1} op)$ instead of $(\sim_{\alpha_2} op)$ if $\alpha_1 \rightarrow \alpha_2$. For example, *add-bg* is ($\sim_\equiv$*del-car-bg*). Suppose that *add-car-bg* is a method by which *car* and *background* objects are added. *add-car-bg* is ($\sim_{state}$*del-car-bg*). A state obtained by performing *add-car-bg* is semantically consistent with one obtained by performing *add-bg*.

[**Theorem**] $(\sim_\alpha op) \sqsubset_\beta (\sim_\beta op)$ iff $\alpha \rightarrow \beta$. □

4 Reduced Compensating Sequence

Let r show RoS "application is not interested in colors". A method *add-car* is r-compatible with *grayscale* ($add - car \diamond_r grayscale$). Suppose *add-car* is performed before *grayscale*, i.e. $add - car \circ grayscale$. This sequence is *r-compensated* by $(\sim_r grayscale) \circ (\sim_r add - car)$. However, it takes a shorter time to perform $(\sim_r grayscale)$ after removing a car which is added by *add-car*, i.e. $(\sim_r add - car)$, because the number of objects whose colors to be changed are decreased. Hence, $add - car \circ grayscale$ can be more efficiently compensated by $(\sim_r add - car) \circ (\sim_r grayscale)$ with respect to RoS r. The method *del-car* is an r-compensating method of *add-car*, i.e. *del-car* = ($\sim_r$*add-car*) = ($\sim_{state}$*add-car*). Since the application is not interested in color, $(\sim_r grayscale)$ can be omitted, i.e. ϕ is $(\sim_r grayscale)$.

Next, let us consider how to reduce the number of compensating methods to compensate a sequence of methods. Suppose a *car* object c is deleted after added, i.e. *add-car* $\circ$ *del-car*. Since (*add-car* $\circ$ *del-car*) $-\phi$ holds, ($\sim_{State}$*del-car*) $\circ$ ($\sim_{State}$*add-car*) is not required to be performed. Next, suppose a method $paint_1$ which paints an object *red* is performed after painting *yellow* by $paint_2$. $paint_2 \circ paint_1$ brings the same result obtained by performing only $paint_1$, i.e. $(paint_2 \circ paint_1) - paint_1$. In order to compensate $paint_1 \circ paint_2$, only $(\sim_\alpha paint_1)$ can be performed. The following relations are defined for methods op_t and op_u and a consistent relation α:

- op_t is an *α-identity* method iff $op_t \sqsubset_\alpha \phi$.
- op_t *α-absorbs* op_u iff $(op_t \circ op_u) \sqsubset_\alpha op_t$.

5 Concluding Remarks

In this paper, we discussed how the QoS of the object is manipulated by methods. We defined semantically, QoS, RoS, semantically QoS, and semantically RoS conflicting relations among methods of multimedia objects. By using the relations, we defined compensating methods to undo the works done by the methods. We need further study to obtain an optimized sequence of methods.

References

[1] Bernstein, P. A., Hadzilacos, V., and Goodman, N., "Concurrency Control and Recovery in Database Systems," *Addison-Wesley Publishing Company*, 1987.

[2] Yokoyama, M., Tanaka, K., and Takizawa, M.,"QoS-Based Recovery of Multimedia Objects," *Proc. of IEEE Int'l Conf. on Parallel and Distributed Systems (ICPADS-00) Workshops*, 2000, pp.43–48.

[3] Korth, H. F., Levy, E., and Silberschalz, A., "A Formal Approach to Recovery by Compensating transactions," *Proc. of VLDB*, 1990, pp.95–106.

[4] MPEG Requirements Group, "MPEG-4 Requirements," ISO/IEC JTC1/SC29/WG11 N2321,1998.

[5] Yokoyama, M., Nemoto, N., Tanaka, K., and Takizawa, M., "Quality-Based Approach to Manipulating Multimedia Objects," *Proc. of 2000 Int'l Conf. on Information Society in the 21 Century: Emerging Technologies and New Challenges (IS2000)*, 2000, pp.380–387.