Extending OMT for the Specification of Composite Objects

Dunia Ramazani
Gregor v. Bochmann
Département d'informatique et de recherche opérationnelle
Université de Montréal
C.P. 6128, Succursalle Centre-Ville
Montréal, Canada H3C 3J7

Abstract:
In engineering and telecommunication applications, it is common to have composite objects. Existing object-oriented methods propose many approaches for modeling these objects. However, these approaches fail to capture the linkage between the structure and the behavior of composite objects. In [Ramazani 95a], a conceptual framework for the description of composite objects prescribes how this linkage can be established by means of a set of fundamental concepts. In order to make this framework more usable in practice, this paper shows how OMT can be adapted and extended to describe composite objects according to this framework. A great deal of these adaptations and extensions require only minor notational and semantic changes to the method. This work also shows how more requirements in connection with composite objects can be captured, made explicit, and precisely stated using an extended OMT.

Keywords: Composite objects, Object-oriented methods, Object-oriented specifications.

This work was funded by the Ministry of Industry, Commerce, Science and Technology, Quebec, and the Natural Sciences and Engineering Research Council of Canada under the IGLOO project organized by the Centre de Recherche Informatique de Montreal.

1. Introduction

The object-oriented approach describes applications by means of objects which collaborate to provide functionality of the application [Askit 92]. Some of these objects are defined in terms of other objects. Former objects are called *composite objects*, and the latter are *component objects*. Experience in the specification of engineering and telecommunication applications has shown that *the linkage between the structure and the behavior of composite objects plays a key role for understanding the semantics of these applications* [Guttapalle 92, Ramazani 95b].

As a matter of fact, in engineering applications, we find objects consisting of physical assemblies of other objects. Such assemblies form composite objects. The behavior of these assemblies can be expressed in terms of the behavior of their components. Since, the behavior of components depend upon their environment, connections among the components affect their behavior. These connections follow *principles of engineering and physics* which can be complex [Ramazani 95b]. In order to understand the behavior of a given composite object, one has to understand the behavior of each component along with their connections. These two form the structure of composite objects.

As a consequence, in engineering applications, the behavior of physical assemblies depends upon their structure. In telecommunication applications, the reasoning is similar, except that the connections among components follow *protocols*. In addition, applications are explicitly structured using *containment relationships* [Guttapalle 92]. A containment relationship localizes an object inside another object. It allows the description of the containing object behavior in terms of that of the contained object, since the former object has access to the latter object.

Existing object-oriented methodologies propose many approaches to the specification of composite objects. These approaches can be classified into four categories as described in [Ramazani 95b]. For example, the Fusion method [Coleman 94] represents composition as the abstraction of a given relationship among component objects. This way of handling composition shows the interconnections between the component objects. It focuses on the structure and neglects the behavior of the composite objects. In OMT [Rumbaugh 91], composition of objects is described by means of is-part-of (aggregation) relationships relating the composite object to its components. This approach has the advantage of describing, in an hierarchical manner, the structure of an object. In addition, it captures the fact that the composite object has access to its components. In methods such as Booch’s OODA [Booch 94] and HOOD [Robinson 92], composition of objects is represented by attribution, i.e., component objects are represented by attributes of the composite object. Attribution captures the hierarchical organization of composite objects. However, it precludes any distinction between composition and interconnection of objects. There are situations where composition of objects is modeled by multiple inheritance. Such situations are reported in [Cargill 92, Rumbaugh 93, Sakkinen 89]. Modeling composite objects by multiple inheritance is appropriate when the identity of component objects is not important and the focus is on the resulting properties of the composite object. This way of handling composition, while treating both structural and behavioral aspects of composition, can create problems when reusing such a specification. For more details on these problems, the reader is refered to [Cargill 91, Cargill 92, Rumbaugh 93, Sakkinen 89]. All these approaches distinguish themselves by focusing on some aspects and neglecting other aspects of composition. Among these aspects, the contribution of the structure of the composite object to its behavior is ignored. To alleviate this problem, we propose in [Ramazani 95a] a new perspective on composition of objects. In the sequel, we briefly present this approach.

The approach focuses on the linkage between the structure and the behavior of these objects. Among the mechanisms which allow this linkage, we find (1) *visibility* of components, (2) *promotion* of component properties to the status of composite properties (inherent properties), and (3) *aggregation* of component properties in order to form properties of the composite object (aggregate properties). These mechanisms determine *the way requirements placed upon structure and behavior of composite objects relate*. Approaches to the description of composite objects which neglect these aspects will fail to capture the connection between the structure and the behavior of these objects. This failure may have an impact upon *reusability* and *modifiability* of composite object descriptions.

This approach combines features of object-oriented methods and it adds to these the concepts and mechanisms necessary for relating structure and behavior of composite objects. Many of these concepts and mechanisms forming this approach already exist within object-oriented methods or they are allowed by the object paradigm. *It is the way they are organized and utilized in the framework which is novel*. As a consequence, object-oriented methods can be adapted to describe composite objects in accordance with the approach. The amount of changes and extensions required depends upon the selected method. In this paper, we show how the OMT [Rumbaugh 91] can be used and extended in order to describe composite objects according to the approach.

In OMT, structural aspects of an application are described apart from its dynamic aspects. Structural aspects are described using object models. An object model is a diagrammatic representation of classes portraying their attributes, operation signatures, and associations. Dynamic aspects are described by means of dynamic models. A dynamic model describes the temporal evolution of the objects in an application in terms of the changes they undergo in response to interactions with the other objects inside or outside the application. The dynamic model is defined by statecharts, each of which describes the behavior of a class. Features of object and dynamic models allow the description of a great deal of aspects of composite objects in accordance with the framework proposed by our approach as shown in the following table.

OMT models	**Features**	**Aspects of composite objects**
Object model	*Class*	- Component classes - Composite class
	Association	- Associations between components - Aggregation association
Dynamic model	*Statechart*	- Behavior of components - Communication by event sending between the components - Behavior of composite objects using the dominant object - Communication between the composite and its components through the dominant object

Table 1: Coverage of the framework by features of object and dynamic models

Step	**Observation**	**Remarks**

Components and interactions between the components	Could be extended	• *Communication between components is restricted to explicit sending of events between objects. This is inadequate for expressing more abstract interactions like constraints between behaviors.*
Inherent and aggregate properties	Inadequate	• *Not covered, except the is-part-of (aggregation) association* • *Extensions are required for:* *- Visibility/hiding of components* *- Promotion of components properties* *- Aggregation mechanisms for component properties* *- Explicit distinction between inherent and aggregate properties*
Emergent properties	Adequate	• *However, OMT should be extended in order to distinguish between emergent and other properties of the composite.*

Table 2: Evaluation of OMT with respect to the framework

The table 2 summarizes our evaluation of OMT with respect to steps of the conceptual framework. Extensions required can be achieved using appropriate notational and semantic changes to OMT, except for visibility/hiding of components. The notion of visibility/hiding of components requires fundamental changes to the object paradigm itself, since it is related to aliasing, identification, reference, and typing of objects. Such changes to the paradigm are out of the scope of this paper.

The structure of this paper is as it follows. We begin by describing an application example which is used throughout the paper. It consists of a portable cassette player. This device has features which help to illustrate significant aspects of the approach. Next, we present our three step process for the description of composite objects. This is followed by the description, using OMT, of the portable cassette player according to our approach. Throughout the description, we describe how to represent characteristics of composite objects and we also reveal shortcomings of OMT. As a consequence, we propose extensions to OMT. Finally, a summary of our contributions closes the paper.

2. The Portable Cassette Player

We assume that the reader is familiar with cassette players. We omit certain details which do not contribute to the essential points of this paper. A portable cassette player consists of two parts, a case and a earphone, as shown in Figure 1. It offers functions for managing the playing of cassettes. In the context of our presentation, the earphone can be considered as a simple object. The case is a composition consisting of the following components:

- a cassette compartment into which a cassette may be placed;
- a motor which moves the tape of the cassette;
- a head which reads the content of the tape and produces the corresponding sound;
- an amplifier controlled by a volume controller.

It regulates the volume level of the sound produced by the head which is then sent to the earphone;

- a control panel for managing the play function. It consists of five buttons, namely play, forward, rewind, pause and stop.

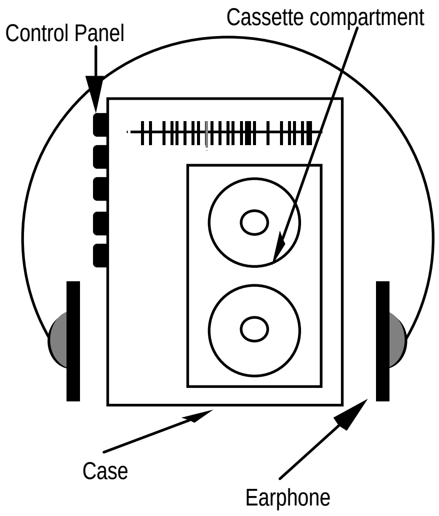


Figure 1: A portable cassette player

The normal operation of the cassette player is as follows. When a cassette is present in the compartment, the user may activate the play function. This is done by pushing the button play. This action turns on the motor and causes the motor to move at normal speed. This action also activates the head. It sets the head to operate in read mode. Then, the sound produced by the head is transmitted to the amplifier which regulates this sound according to the level of volume determined by the volume controller, and transmits it to the earphone. The user may move forward a cassette by pushing the forward button. This action turns on the motor and causes the tape to move forward at fast speed. He may also move backward a cassette by pushing the rewind button. This function is similar to forward except that the tape moves backward. To stop these functions, the user has to press the stop button. The pause button is used to temporarily disable the playing function by turning off the motor. To resume the play function, the user has to deactivate the pause function by pushing one more time on the button.

Each part of the portable cassette player and the player itself are considered as physical objects. As such, they have a weight and a position in space. The portable cassette player has also other properties such as a price which the owner may want to change. The panel consists of five buttons namely, forward, play, rewind, pause, and stop. Each button can only be in two states "On" or "Off". The play, forward, and rewind buttons are mutually exclusive in the sense that two of these buttons can not be in the "On" state at the same time. In addition, when the stop button is "On", forward, play, and rewind buttons must be "Off". We also assume that the control panel has a part number which serves for its identification.

3. Aspects of Composite Objects

Among the aspects which characterize composite objects, our experience has shown that three distinct aspects are important [Ramazani 95a]. These aspects are: (1) its structure; (2) its inherent and aggregate properties; (3) its emergent properties. We may define a composite object as follows.

Definition 1: ***Composite object***

A composite object is an object with an internal structure. It has three kinds of properties, namely inherent, aggregate and emergent properties. A property may denote an attribute, an operation, a behavior, a structural relationship, a behavioral interaction or a sequence of interactions. The structure of a composite object consists of:

- the components (type and number of instances within the composite object);
- the interconnections and/or dynamic interactions between the components of the composite object.

Definition 2: ***Inherent property***
An inherent property is a property of the composite object such that the semantics of this property is given by a property of some component of the composite object. When the component on which depends this property is absent in the composite object, the inherent property is undefined.

Definition 3: ***Aggregate property***
An aggregate property is obtained by a combination of corresponding properties of all components. The aggregation mechanisms used for combining these properties are defined at the composite object level and they depend on the way the components are interconnected.

Definition 4: ***Emergent property***
An emergent property is a property of the composite object which does not directly depend upon the properties of the components.

What really makes a composite object different from a simple object is the possible presence of inherent and aggregate properties. As a consequence, a composite object showing only emergent properties can be treated as a simple object. The requirements in connection with each of these aspects may lead to complex specifications. Separation of concerns is used in the description of composite objects by subdividing the description into three distinct steps, each focusing on a specific aspect. These steps are the following.

Step 1: Structure of composite objects
In the process of specifying composite objects, we first begin by describing the structure of these objects in terms of components and interactions among these components. The step consists of three activities which are as follows.

(1) *Identification of the components* : Here, we specify the number of components and the names or identifiers which shall be used to address these components within the specification of the composite object.

(2) *Description of individual properties of components* : Individual properties of components consist of requirements in terms of attributes, operations, and behavior that these objects must support in order to be components of the composite.

(3) *Description of collective properties of components* : Collective properties of components consist of structural relationships (object relationships or associations) and behavioral interactions among components. We mean by a behavioral interaction a constraint between two or more behaviors. It affects the behavior of the involved objects. This constraint can be described in terms of dependencies between operations, states, and sequences of operations and states of the involved objects.

Step 2: Inherent and aggregate properties
Next, we have to describe how component properties relate to composite properties through inherent and

aggregate properties. This step consists of the following activities:

(1) *Definition of the characteristics of the is-part-of relationship binding components to the composite* : This includes determining visibility and hiding of component objects. Here *visibility* is taken in the sense of controlling access to an object. An object x is visible to an object y if and only if y has a reference to x and y can access x using this reference. An object x is invisible to an object y if and only if (1) y has no reference to x, or (2) y has a reference to x, but y is not allowed to access x using this reference.

(2) *Description of inherent properties* : Note that properties of visible components are automatically considered as inherent properties since they are available to clients of the composite object.

(3) *Definition of aggregate properties* : Care should be taken when defining how component properties compose to form aggregate properties. The composition mechanism applied on these properties needs to be compatible with the composed properties. When composing component properties, the collective properties of the components may conflict with the composition mechanism. As an example, when we use concurrent aggregation of component behaviors, if the components have dependencies, care should be taken to avoid deadlock situations.

Step 3: Emergent properties

Finally, we describe the emergent properties by extending the inherent and aggregate properties through the definition of new properties. This is a case of specialization. These new properties must be compatible with inherent and aggregate properties. Property extension follows subtyping rules, and consistency checking in relation to existing properties is done according to these rules. Emergent properties are described as usual properties of simple objects.

4. Using OMT to describe composite objects

Before illustrating how OMT can be used to describe composite objects in accordance with the approach as described in Section 3, we shall present how concepts of OMT, in connection with composite objects, relate to those of our approach. Among these concepts, notion of composite object, semantics of aggregation relationship, behavior of composite objects, and associations involving composite objects are compared. The material of this section comes from [Rumbaugh 94, Rumbaugh 95a, Rumbaugh 95b, Rumbaugh 95c, Rumbaugh 95d and Rumbaugh 95e] which describe the second generation of OMT.

4.1. OMT versus our approach

What is a composite object?

In OMT, a composite object is an extended form of aggregation where the composite is viewed at a higher level of abstraction than the parts. The whole and its parts are at the same semantic level and can coexist at runtime. According to Rumbaugh [Rumbaugh 94], *Composites have no additional semantics but instead serve to organize your understanding of a model*. An aggregate (or composite object) is a set of objects taken together that is viewed as a single high-level object. The dominant object in a composite object is an object which holds information common to the entire composite. Also, the composite is distinguishable from its dominant object.

We view a composite object as an object formed by the superposition of three kinds of properties namely inherent, aggregate, and emergent. A composite is distinct from its components. But components are indiscernible from the composite, i.e. when the components are involved in interactions which are triggered by objects outside the composite, these interactions also involve the composite. Composite objects represent concrete or abstract things of the world which can be described in terms of other things. Special attention is given to relationships and interactions among components.

The concept of composite objects proposed by OMT is appropriate for describing objects which, when associated with one another may form a conceptual entity for analysis, design, and implementation purposes. These conceptual entities may not necessarily exist in the situation represented. This approach contrasts with our approach, since the concept of composite object in our approach can be used to model both existing composite objects and/or composite objects created by the specifier, like in OMT.

Comments on the is-part-of relationship

In OMT, the terms is-part-of and aggregation are used indistinctly to denote composition of objects. Is-part-of relationships are represented by aggregation which is a special form of association. Aggregation is anti symmetric and transitive. There are two kinds of aggregation namely physical aggregation, i.e. aggregation with multiplicity of one, and catalog aggregation, i.e. aggregation with multiplicity of many. The main distinction between these forms is the possibility of sharing components.

In our opinion, in addition to the properties introduced by OMT, is-part-of relationships are non-reflexive at the instance level. There are many forms of is-part-of relationships. These forms are characterized by seven aspects namely dependency, sharing, physicality, visibility, functionality, homogeneity, and separability. More on these characteristics of is-part-of relationships can be found in [Ramazani 95a]. Special attention is given to visibility of components due to its intricate interaction with aliasing, identification, reference, and typing of objects. In the presence of physical (or concrete) composite objects, some of its clients may have a partial view of the composite by interacting only through its visible components. In order to have an accurate description of such situations, we need to consider the visibility of components. That is why in our approach the emphasis is on this aspect.

Behavior of composite objects

In OMT, the behavior of a composite is similar to the behavior of simple objects. It is described by means of the statechart of its dominant object. There is implicit concurrency between the components. Behavioral interactions are expressed using:

(1) informal text companion to the object, dynamic, and functional models. This can give rise to human interpretation errors during later stages of the development;

(2) operational restrictions by controlling the execution of actions through preconditions and sending of events.

We recognize three kinds of behavior for composite objects namely inherent, aggregate and emergent behaviors. A composite object is logically distributed among its components, therefore there is concurrency in a composite.

Associations involving composite objects

In OMT, there are implicit and explicit associations for composite objects. Implicit associations involve the components, while explicit associations involve only the dominant object. Our approach introduces three kinds of associations. These associations are distinguished by the participating objects which can be the components and/or the composite. Inherent associations are associations which involve some of the components. Aggregate associations involve all the components and the composite. Finally, emergent associations involve only the composite.

To summarize, the main differences between OMT and our approach lie in:

a) *raison d'être of composite objects.* In OMT, composite objects are analysis, design, and implementation artifacts. This contrasts with our approach where the concept is in addition also used to represent composite objects existing in hypothetical and/or real-world situations.

b) *properties that characterize composite objects*. In OMT, composite objects are indistinguishable from non-composite objects. We distinguish between composites from non-composites through the presence of inherent and aggregate properties in composite objects.

c) *linkage between structure and behavior of composite objects*. This aspect is neglected by OMT.

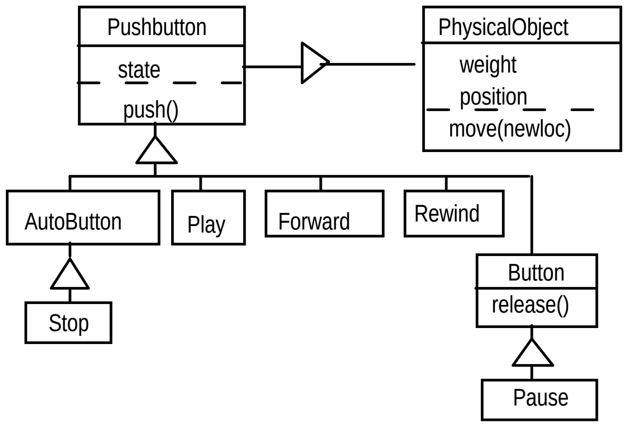
4.2. Description of composite objects

In order to show how OMT can be used to describe composite objects in accordance with our approach, we shall follow the process defined in Section 3. For each step, we describe how OMT can be used and what are the adaptations and extensions required. In addition, we use the portable cassette player to illustrate the ideas.

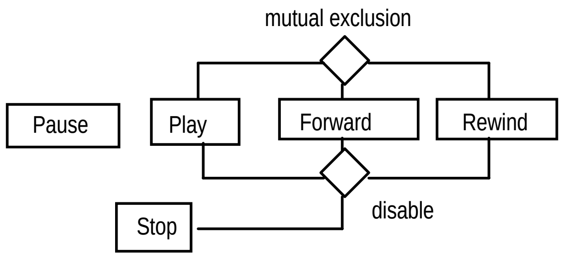
Step 1: Structure of composite objects

Individual properties of components are adequately represented using the concept of class of OMT in conjunction with statecharts associated to classes. The class captures attributes and operations of components, while the statechart captures the behavior of components. Structural properties are represented by means of OMT associations. In OMT, behavioral interactions are represented by explicit communication between the statecharts or by using the state of one class in the guard of transitions of another class as proposed by Rumbaugh [Rumbaugh 95b]. This way of representing behavioral interactions is harmful to reusability and modifiability. The problem is that at analysis phase, behavioral interactions have to be expressed in a more abstract way so that the description does not introduce implementation bias. One way to achieve this is to describe these interactions by constraining the behavior of the involved objects. This is done by using predicates which constrain features of various statecharts. The way the conditions imposed by these predicates are realized are left for other phases of the development.

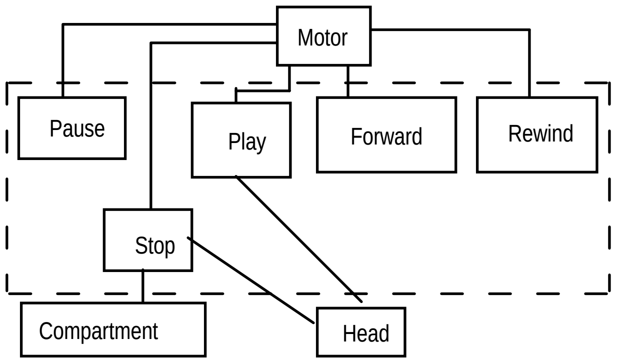
Structures of composite objects are represented by (1) object models describing attributes and operations of components, and structural relationships between components; and by (2) dynamic models describing behavior of components and behavioral interactions among the components. The objects outside the composite object are separated from the components by a dashed rectangle surrounding the latter objects. For illustration, consider a control panel. Its structure is presented in the next figures. Each type of button is represented by a distinct class in order to facilitate the presentation of structural relationships and behavioral interactions among the components. A distinction is made between configurational and external relationships. Configurational relationships are associations among components, while external relationships are associations between components and objects outside the composite.



Attributes and operations of components

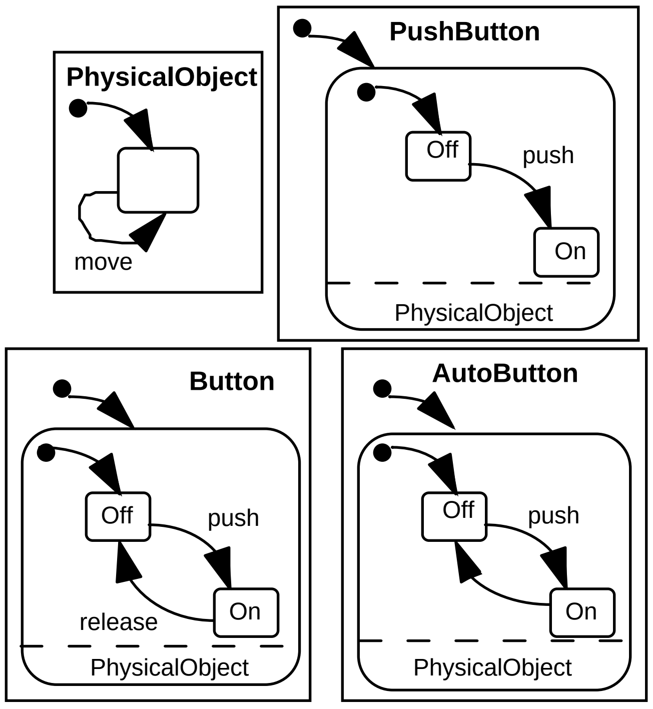


Configurational relationships

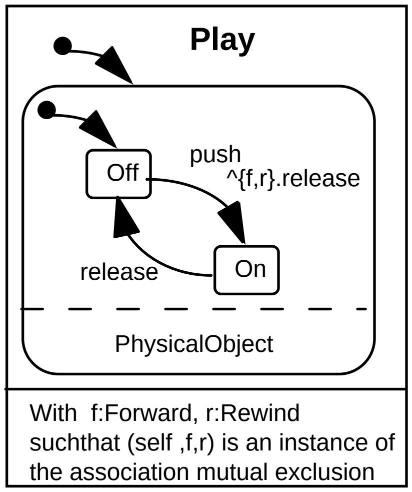


External relationships

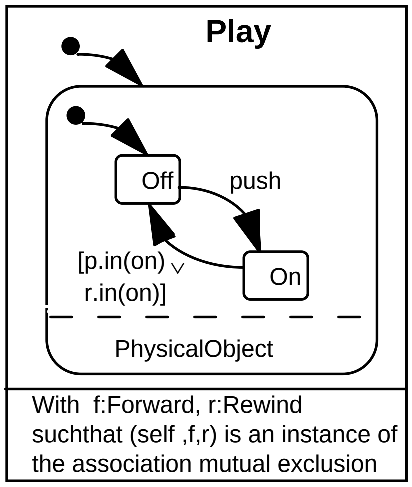
The difference between PushButton and AutoButton is that in an AutoButton, the button automatically comes back to the state "Off" at the completion of the push operation.



In OMT, we have two alternatives for representing the constraints among the buttons. Consider for example the mutual exclusion between Play, Forward, and Rewind buttons. One way is to introduce a *release* operation which will be triggered when another button is *On*. This makes the statechart of the Play button to look like the statechart *Sending* *events*. The other alternative which is illustrated in the statechart *Guards on transition* consists of introducing a spontaneous transition with a guard using the states of the mutual exclusive buttons. The notation *object.in(state)* is used in OMT to denote the predicate which is true when the object is in the state "state" and false otherwise. According to Cook and Daniels [Cook 94], descriptions of message sending confuse specification issues with implementation tactics. It appears that the approach *Sending events* over-specifies the interactions between the buttons since it introduces unnecessary sequencing between the transitions of the buttons.



Sending events



Guards on transition

Instead of using this approach, we propose an extension to the notation which consists to describe interactions between classes by means of contracts. The concept of a contract is similar to the Contract technique introduced by Helm and Holland [Helm 90, Holland 92, Holland 93]. In our approach, we represent contracts using the ad hoc textual convention illustrated below. In this ad hoc notation, <utility> is the name of the association abstracting the interactions between the classes.

contract for <utility>

<role(i): component-type>
...
<role(n): component-type>

<set of assertions>

Contracts in OMT

In a contract, an assertion can be any predicate allowed in OMT specifications. For instance, the contracts describing the associations "mutual exclusion" and "disable" are represented by the following diagrams.

Contract for mutual exclusion

p:Play; f:Forward; r:Rewind

(p,f,r) is a instance of the association mutual exclusion
$p.in(on) \rightarrow f.in(off) \wedge r.in(off)$
$f.in(on) \rightarrow p.in(off) \wedge r.in(off)$
$r.in(on) \rightarrow f.in(off) \wedge p.in(off)$

(I)

Contract for disable

p:Play ; f:Forward; r:Rewind; s:Stop

(p,f,r, s) is a instance of the association disable
$s.in(on) \rightarrow p.in(off) \wedge f.in(off) \wedge r.in(off)$

(II)

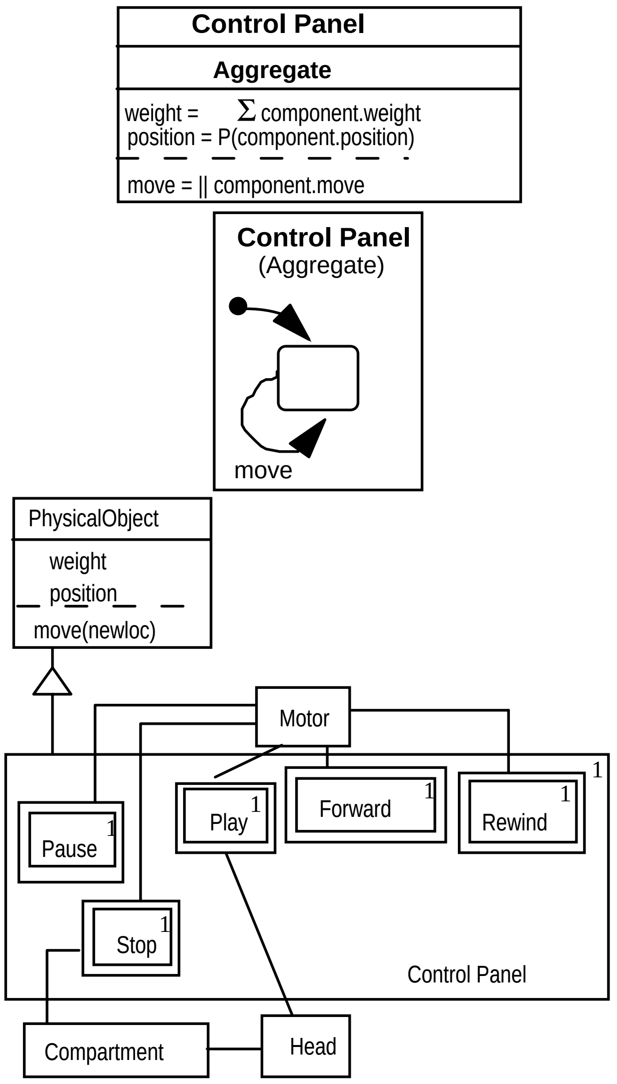
A contract represents the concurrent composition of its participant statecharts such that the behavior of the participants conforms to the constraints explicitly stated in the contract. AND-composition of statecharts corresponds to concurrent composition of statecharts as defined in [Rumbaugh 95b]. The composed objects can interact explicitly by sending events. They can interact implicitly if one object has a guard condition that depends on the state of another object. In our contract notation, the interactions between the statecharts are expressed in an abstract manner using predicate logic as well as OMT constraints. The global statechart represented by the contract may have less states than the statechart obtained using only AND-composition without communication between the member statecharts. This is due to its set of assertions (i.e. the constraints). For instance, the statechart (I) has 4 states while the AND-composition without communication between Play, Forward, and Rewind buttons has 8 states. The statechart (II) has also 4 states compared to the AND-composition of its member statecharts which has 16 states.

Step 2: Inherent and aggregate properties

We found that the notation used in OMT for representing aggregation relationships is adequate. This is done by graphically including the component classes within the composite class. Multiplicity of components is indicated by a number at the right corner of the class. However, the notation does not cover visibility of components. Therefore, we extend OMT as follows. Within a composite, visible components are represented by using double rectangles and hidden components by simple rectangles.

OMT does not distinguish between inherent and aggregate properties of a composite object. Therefore, we need also extensions to the notation. *Concerning attributes, operations, and behavior due to visibility of components, the notation proposed for visible components implicitly conveys the idea that properties of visible components are also properties of the composite*. For attributes, operations, and behavior which originate from the hidden components of the composite, we require a specific box and statechart to represent these. Inherent relationships which involve visible components are represented by lines which cross the composite object boundary and end up at the visible component. Inherent relationships which involve hidden components are represented by lines ending up at the composite boundary. The mapping with the corresponding component is achieved by drawing a dashed line within the composite from the component to the end point of the relationship at composite boundary. An end point is represented by a small rectangle at the composite object boundary. Keep in mind that aggregate properties may imply constraint propagation, as described in [Rumbaugh 88, Mili 90], from the composite to its components. Aggregate attributes, operations, and behavior are represented like inherent properties originating from the hidden components, except that these attributes and operations are annotated with aggregation assertions explaining how the component properties are combined. Aggregate relationships are represented by lines ending up at the composite boundary. At this end point, the annotation {A} indicates that it is an aggregate relationship. In OMT a dominant object can be used to represent aggregate attributes, operations, and behavior. However, the notation does not distinguish between aggregate and inherent properties.

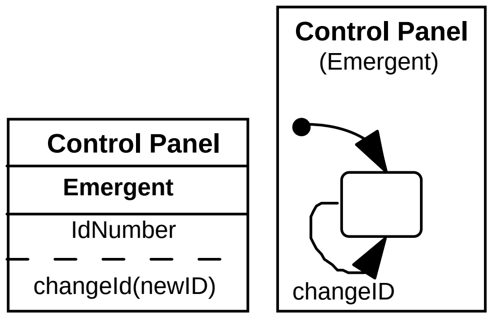
For illustration, consider again the control panel. All its components are visible. Therefore, the properties of the buttons become inherent properties of the control panel. Its weight and position constitute aggregate attributes. It has only one aggregate operation which is "move". In the definition of these aggregate properties "component" stands for any component of the control panel, i.e. the "component" may be Pause, Play, Forward, Rewind or Stop. We may also use contracts to define the semantics of these properties.



Step 3: Emergent properties

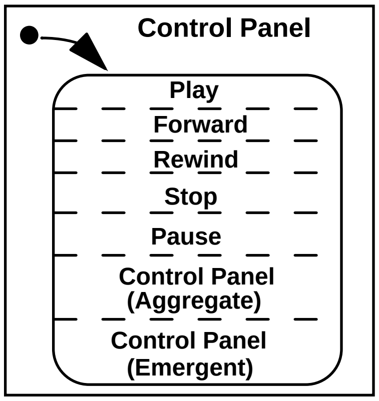
In OMT, emergent attributes, operations, and behavior can be represented by attributes, operations, and behavior of the dominant object. Using this artifact, the distinction between aggregate and emergent properties is not clear. Instead, we propose to extend the method.

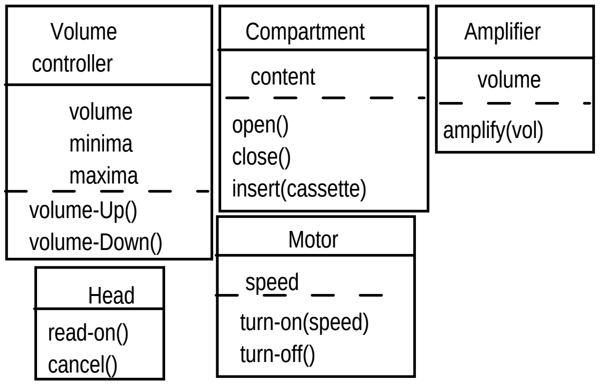
We use a diagrammatic notation similar to the one proposed for aggregate properties with an indication that these properties represent emergent properties. For the control panel, when considering its identification number and the operation used for updating this number, this looks like illustrated below. Unfortunately, in this case, there is no emergent relationship.



Behavior of composite objects

In our approach, the behavior of composite objects is split into three parts namely inherent, aggregate, and emergent behaviors. The behavior of the composite object is represented by the statechart resulting from concurrent-composition of the behavior of its visible components, inherent behavior, aggregate behavior, and emergent behavior. For the control panel, this looks like illustrated below.

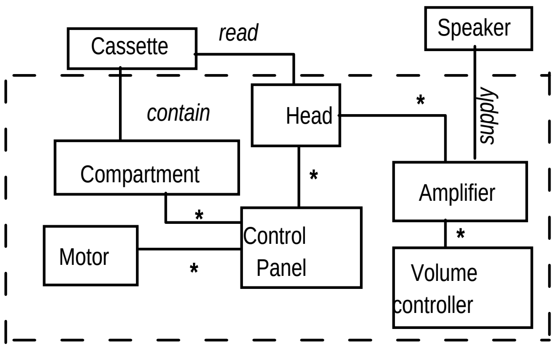


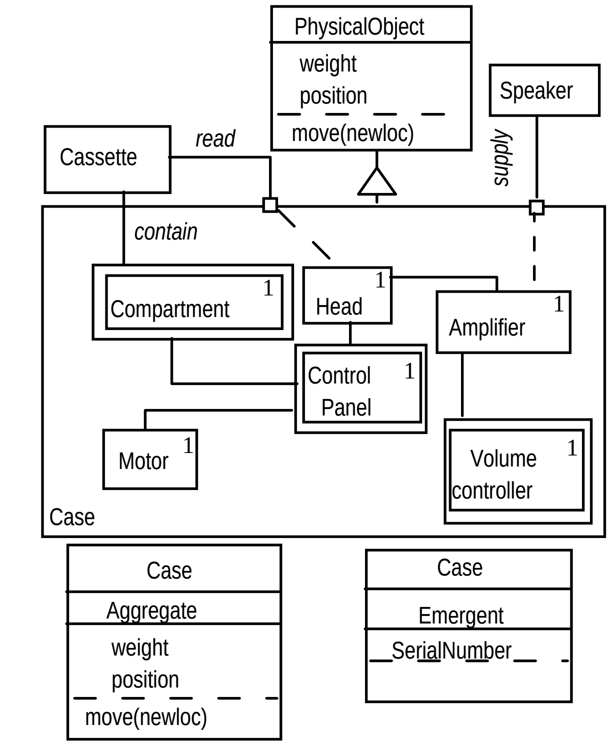


I: Structure of the case

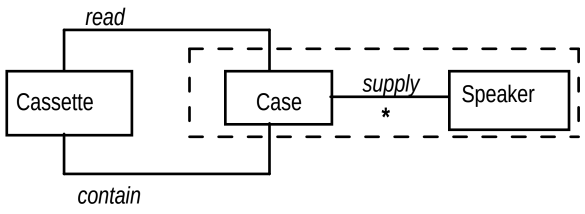
Other composite objects in the context of the portable cassette player

For sake of brevity, we make some simplifications in the presentation. The case has six components, namely the control panel, the motor, the compartment, the head, the amplifier, and the volume controller. There are a lot of behavioral interactions between the components of the composite objects case and cassette player which require an explicit contract. However, due to lack of space, we only indicate in the figures the associations which require a contract. This is done by labeling with a "*" symbol the line representing the association.

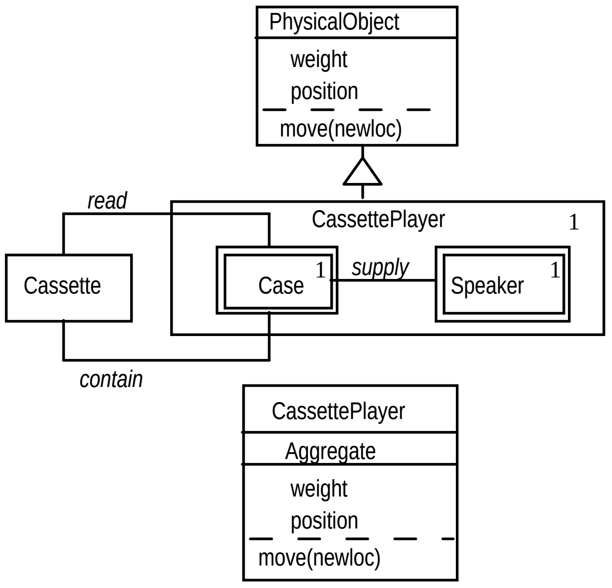




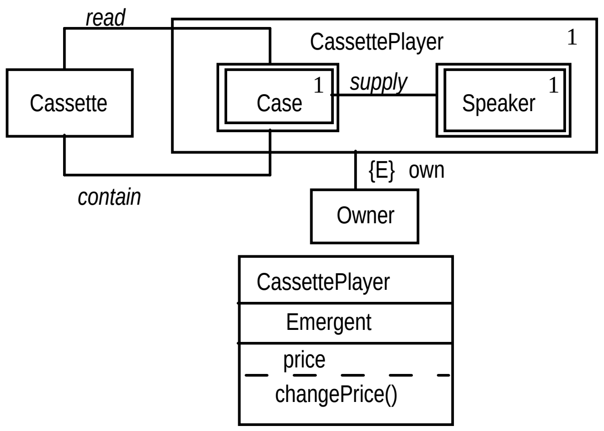
II: Inherent, aggregate and emergent properties of the case



III: Structure of the cassette player



IV: Inherent and aggregate properties of the cassette player



V: Emergent properties of the cassette player

Only the control panel, the compartment, and the volume controller are visible components of the case. As in the case of the control panel description, the properties of these visible components become inherent properties of the case. Therefore, the case has three inherent relationships, namely contain, read, and supply. The association "contain" links the visible component compartment to a cassette. In addition, the head is allowed to read the cassette and the amplifier to supply the earphone with sound. These properties of hidden components are mediated by the case and become inherent properties of the case. Using the consideration that all parts of the cassette player are physical objects, we may define the weight, the position, and its change as aggregate properties of the case. We consider only one emergent attribute for the case, which is its serial number. The overall behavior of the case is defined by superposing the behavior of its visible components (control panel, compartment, and volume controller), its inherent behavior realized using hidden components (head and amplifier), as well as its behavior as a physical object (aggregate properties). We omit the statechart representing this behavior since it is similar to the statechart of the control panel.

The cassette player has two components, the case and the earphone. They are related by the fact that the case provides the earphone with the sound. Both the case and the earphone are visible components of the cassette player, therefore the properties of these visible components become inherent properties of the cassette player. Aggregate properties are similar to those defined for the case. We have chosen to model the ownership of the cassette player as an emergent property. Besides this property, we may consider the price attributed to the cassette player and to allow updating this price when necessary. These constitute additional emergent properties of the cassette player. The overall behavior of the cassette player is defined by superposing the behavior of its visible components (case and earphone), its behavior as a physical object (aggregate properties), as well as the possibility of owning the cassette player and defining its price.

More on visibility of components

The concept of visibility of components, as introduced by our approach, has much to do with subtyping and it should not be confused with visibility of class features proposed for object-oriented programming languages. In our approach, when a component is visible, this means that its substitution by a more specialized one allows the composite to offer more properties. This contrasts with visibility of class features which introduces three kinds of visibility namely public, protected, and private. Public features are accessible to clients and can be used in derived classes. Protected features are inaccessible to clients but can be used in derived classes. Private features are only accessible within the class. Visible components are public features of composite classes while hidden components may be protected or private features of composite classes. Visibility of class features does not convey the idea that the composite shares its interface with its visible components and that the latter objects are substitutable by the former object.

We may draw an analogy between visibility of components and subtyping (inheritance). The composite class is a subtype of all its visible component classes since they have common properties and in most of the cases the composite offers all the properties of its visible components. When a composite class defines more than one visible component, this situation is similar to multiple inheritance. However, a composite object created by multiple inheritance can not have multiple instances of the same class as its components. In addition, the components can not be replaced. As a consequence, we do not recommend the modeling of visibility of components using visibility of class features or subtyping (inheritance). The concept of visibility of components has to be further examined and suitable concepts, mechanisms, and principles have to be defined.

5. Conclusion

Various aspects of composite objects are explicitly captured using an extension of OMT. This allows to envision their usage in the specification of requirements with respect to composite objects. Returning to the purpose of this paper, the contributions are twofold:

(1) On the one hand, OMT is extended with:

- a concept of composite object which encompasses both concrete objects of an application, and abstractions in the form of aggregations created for analysis, design, and implementation purposes;
- an explicit linkage between structure and behavior of composite objects by means of a set of fundamental concepts;
- an understanding of the major differences between composite and non-composite objects according to structural and behavioral characteristics of these objects;
- more abstract forms of communication between objects;
- a new concept of visibility/hiding of component objects; etc.

(2) On the other hand, our approach to the description of composite objects described earlier [Ramazani 95a] is made more usable in practice through its integration within OMT.

In the literature, the work done by Mili and colleagues [Mili 90] is close to the conceptual framework. In [Mili 90], there is an in depth study of how the structure of composite objects may affect or influence their behavior. In particular, they assume that some behavioral and functional relationships between objects are the consequences of the structure relationships. Structure relationships include the aggregation relationship and the connections between the components. We may relate this approach to our conceptual framework as follows. In [Mili 90], the concept of change propagation between related objects is key to the specification of constraints between the composite and its components and it implicitly indicates the connection between the structure and the behavior of composite objects. Considering our conceptual framework, we may classify the properties of a composite object into two classes. The first class involves properties for which change propagation is necessary from the composite to its components. The second class involves properties for which there is no change propagation. This latter class of properties corresponds to emergent properties in our framework. The former class of properties can be further subdivided into two categories. One for which the change propagation are limited to one component, corresponding to inherent properties, and one for which the change propagation affects all components. This latter category corresponds to aggregate properties. While the two approaches, ours and [Mili 90] can be reconciled, there is an important difference between the two approaches. In [Mili 90], the structure of an object is not visible to outside objects such that an outside object can not request operations that directly manipulate the component objects. In our approach, we allow that certain components may be visible; these components may be accessed and manipulated from the objects outside of the composition.

Our experience with engineering and telecommunication applications has shown that extended OMT allows to capture explicitly more requirements in connection with composite objects. It has also shown that the extended OMT improves reusability and modifiability of composite object descriptions due to the application of (1) separation of concerns and (2) superposition through concurrent-composition of component behaviors. In the future, we plan to look at other object-oriented methods, such as the Fusion method [Coleman 94], and Object Oriented Design with Applications [Booch 94]; and also to update an existing tool supporting OMT in order to provide means for using the approach proposed in this paper. We shall also further examine the implications of visibility/hiding of components upon the object paradigm and how these concepts can be fully integrated within object-oriented methods and programming languages.

References

Askit, M., Bergmans, L., Obstacles in Object-Oriented Software Development, Proceedings of OOPSLA'92, SIGPLAN Notices Vol. 27, No. 10, pp. 341-358, October 1992.

Booch, G., Object-Oriented Analysis and Design with Applications, Second Edition, The Benjamin/Cummings Publishing Co. Inc., 1994.

Cargill, T., A Case Against Multiple Inheritance in C++, Proceedings of USENIX Conference, 1991.

Cargill, T., C++ Programming Style, Addison-

Wesley, 1992.

Coleman et al., Object-oriented Development, THE FUSION METHOD, Prentice-Hall, 1994.

Cook, S. and Daniels, J. Object communication, JOOP, September 1994.

Guttapalle, N., Kilov, H., and Morabito, J., The Materials: A Generic Object Class Library for Analysis. Information Modeling Concepts and Guidelines, Science and Technology Series, ST-OPT-002010, Issue 1, October 1992, BellCore.

Helm, R., Holland, I. M., and Gangopadhyay, D. Contracts: Specifying Behavioral Compositions in Object-Oriented Systems, Proceedings of ECOOP/OOPSLA'90, Ottawa 1990, pp. 169-180.

Holland, I.M. Specifying Reusable Components Using Contracts, Proceedings of European Conference on Object-Oriented Programming 1992 (ECOOP'92), LNCS 615 Springer-Verlag, pp. 287-308.

Holland, I. M. The Design and Representation of Object-Oriented Components, Ph.D. Thesis from Northeastern University, Boston, 1993.

Mili, H., Sibert, J., Intrator, Y., An Object-Oriented Model Based on relations, Journal of Systems Software, No 12 pp. 139-155, 1990.

Ramazani, D., Bochmann, G.v., A Conceptual Framework For Object Composition and Dynamic Behavior Description, Publication départementale #949, DIRO, Université de Montréal, Montréal, Canada, 1995.

Ramazani, D., Contribution of Object-Oriented Methodologies to the Specification of Complex Systems, Proceedings of Fifth Complex Systems Engineering Synthesis and Assessment Technology Workshop (CSESAW'95).

Robinson, P., Hierarchical Object-oriented Design, Chapman & Hall, 1992.

Rumbaugh, J., Controlling Propagation of Operations using Attributes on Relations. Proceedings of OOPSLA'88 Conference, September 1988.

Rumbaugh, J. et al. Object-oriented Modeling and design, Prentice Hall, 1991.

Rumbaugh, J., Disinherited! Examples of misuse of inheritance, JOOP Vol. 5, No. 9, pp. 22-24, February 1993.

Rumbaugh, J., Building Boxes: Composite Objects, JOOP Vol. 7, No. 7, pp. 12-22, November/December, 1994.

Rumbaugh, J., OMT: The object model, JOOP Vol. 7, No. 8, pp. 21-27, January, 1995.

Rumbaugh, J., OMT: The dynamic model, JOOP Vol. 7, No. 9, pp. 6-12, February, 1995.

Rumbaugh, J., OMT: The functional model, JOOP Vol. 8, No. 1, pp. 10-14, March/April, 1995.

Rumbaugh, J., OMT: The development process, JOOP Vol. 8, No. 1, pp. 8-16, May, 1995.

Rumbaugh, J., Taking things in context: Using composites to build models, JOOP Vol. 8, No. 7, pp. 6-11, November-December, 1995.

Sakkinen, M., Disciplined Inheritance, Proceedings of ECOOP Conference, 1989, pp. 39-56.