Historically, two conceptual modeling constructs have been used within the database literature to represent composites: entity classes or object classes, and relationship classes or associations. Artale et al. [1] call the former approach the explicit modeling approach—the composite is represented explicitly in the conceptual model. They call the latter approach the implicit modeling approach—overall knowledge about the composite is represented implicitly in the conceptual model via the attribute and relationship classes belonging to its components.

For example, Figure 1(a) illustrates the concept of marriage, as represented in an entity-relationship model proposed by Chen [4]. He models marriage, a composite comprising two partners, via a relationship class construct. Alternatively, Figure 1(b) shows marriage modeled via an entity class. Similarly, Rum-baugh et al. [7] model a team that comprises performers and backups as an association. Alternatively, the team could be modeled as an object class. Within the knowledge-repre-

Composites are important phenomena that occur frequently in the real world [8]. A composite is a thing that is composed of other things. For example, a bicycle is composed of wheels and a frame, an orchestra is composed of a conductor and various musicians, and a team is composed of a leader and members. Because conceptual models are models of reality and composites are a part of that reality, we must be able to represent composites faithfully as a basis for building databases, knowledge bases, and information systems in general. Otherwise, our ability to understand and reason about composites and their components will be fundamentally impaired [1].

By Graeme Shanks, Elizabeth Tansley, and Ron Weber
sentation literature, similar approaches have been used to represent composites and components [1].

In this article we argue that a composite should always be modeled as an entity or object class and never as a relationship class or association. In this regard, Figure 1(a) is an appropriate model if we are concerned only about a person and a mutual property (marriage) the person possesses with another person. The relationship class of marriage rightly models this mutual property of the two persons. We consider marriage as a thing in its own right having its own intrinsic properties (properties possessed only by the composite) and mutual properties (properties possessed by the composite and another thing to which it is related), however, we argue that Figure 1(a) is an inappropriate model. Instead, the model in Figure 1(b) should be used.

**Fundamental Concepts**

When component things are combined into a larger composite thing, the composite always gains at least one property that did not exist previously, called an emergent property [3, 11]. Emergent properties may be intrinsic or mutual properties. For example, when employees are pooled together to form a team, the composite (the team) has a new property—namely, number of members. This property cannot be attached to any individual team member. It is this emergent property that distinguishes composite things from simple things and makes them of particular interest to conceptual modelers. If a composite has no emergent property of interest to the modeler, the composite itself is of no interest. If the composite has one or more emergent properties of interest, however, then modeling these properties correctly is critical.

Certain ontological theories (theories dealing with the structure and dynamics of the real world) tell us that representing composites as relationship classes or associations is inferior to representing them as entity classes or object classes [11]. These theories do not permit properties to be attached to a relationship class or association (see [11] for a full explanation). The reason is they ascribe “ownership” of properties only to things in the world. Properties themselves cannot possess properties. Thus, important real-world phenomena associated with composites (emergent properties) cannot be modeled if the composites themselves are represented as properties (see also [1, 6]).

Unless the composite is modeled as an entity or object class, mutual properties that components have with the composite also cannot be modeled easily. For instance, a person may be a stepparent in one or multiple marriages (but not other marriages). This fact cannot be represented easily in Figure 1(a) where marriage is shown as a relationship class. One approach would be to attach an optional attribute called “stepparent” to the “person” entity class. Problems arise with this solution, however, if a person is a stepparent in one marriage but not another marriage. (What is the status of the optional attribute?)

The fact of being a stepparent in a marriage can be represented easily in Figure 1(b), however, simply by showing a stepparent relationship class between the person entity class and the marriage entity class. The semantics underlying the fact are clear. We can identify the specific marriage or marriages in which a person is a stepparent.

In short, two problems arise when composites are represented as relationship classes or associations and not entity classes or object classes: emergent properties of the composite cannot be represented, and mutual properties between the composite and its components cannot be represented.

To illustrate these issues, consider the example of a team composed of one or more employees. Assume some employees are team leaders, and some are team members. Figure 2(a) illustrates the team composite using an association construct. Figure 2(b) illustrates the team composite using an object class construct. The two models shown in the figures are intended to represent the same team/leader/member reality. Both
figures have been depicted according to the UML class diagram syntax described in [7]. Carefully consider the following questions:

**Question 1: Can a team have no leader?** Consider, first, the model in Figure 2(a). The answer to this question depends on how we interpret “team.” Does team refer to one particular association link between a single leader and a single member? Does it refer instead to the set of association links between leader and member employees who make up a team in real life? Presumably, the latter interpretation is more appropriate, because the word team refers to all involved leaders and members. The question then becomes, “Can we have a set of association links with no leader instance?” The answer to this question must be no, because an association link must have an object instance at each end. Note how the cardinalities suggest a different answer.

Now consider the same question using the model in Figure 2(b). The answer is clearly yes. The cardinalities indicate a team may have no leader. This answer contradicts that obtained Figure 2(a), even though the two models are intended to represent the same reality.

**Question 2: Can a team have no members?** The model in Figure 2(a) does not permit us to have an association link with an object instance at one end but not the other. Based on Figure 2(a), therefore, the answer to this question is no. Figure 2(b) clearly indicates, however, that a team can have no members. As with Question 1, the two models lead to contradictory answers.

**Question 3: Can a team have no leader and no members?** This question might seem strange, but consider a team given a name, security access rights, project aims, and so on, before any employees are assigned to it. According to Figure 2(a), if we consider a team to be represented via a set of association links between a leader and several members, the answer to this question is no. Each association link must have an object instance at each end. Figure 2(b) clearly indicates, however, that a team may have no leader and no members. Figures 2(a) and (b) once again contradict each other, even though they are intended to represent the same reality.

**Question 4: Can an employee lead more than one team?** This question is also difficult to answer using the model in Figure 2(a). The cardinalities indicate a leader may be teamed with zero or many members, but are all these employees on the same team? Figure 2(a) does not prohibit a leader being teamed with 15 different members divided into three teams of five members each. Nor does the model explicitly allow it. The answer is indeterminate. On the other hand, Figure 2(b) clearly indicates that a leader may lead more than one team.

**Question 5: Can an employee be a member of more than one team?** If an employee were a member of more than one team, that employee would be teamed with more than one leader. Figure 2(a) allows this possibility. Is this because an employee can be in more than one team, however, or because a team can have more than one leader? The model does not differentiate, and once again the answer is indeterminate. On the other hand, Figure 2(b) clearly indicates that an employee may be a member of more than one team.

**Question 6: Must a leader be assigned to a team?** This question might seem incongruous, but consider the possibility that employees might be required to undertake a course in project management before they are permitted to be team leaders. An employee who has successfully completed the project management course is then considered a qualified leader, but may not yet be assigned to a team.

This question is difficult to answer based on the model in Figure 2(a), which indicates that an employee who plays the role of leader may be teamed with zero or many members. Therefore, a leader may not be associated with any members. Is this because the leader has not been assigned to a team? Or is it because the leader has been assigned to a team with no members? Based on the model in Figure 2(a), recall that a team instance cannot exist without a leader instance and a member instance. Thus, it seems likely that the answer to Question 6 is no, because a leader cannot be assigned to a team unless at least one member of the team exists. In short, the only way employees can be leaders yet not be associated with any members is for them not to be assigned to a team. The answer to the question based on Figure 2(b), however, is clear. A leader does not have to be assigned to a team.

**Question 7: Must a member be assigned to a team?** As per the analysis in Question 6, the answer to this question appears to be no based on the model in Figure 2(a), and is definitely no based on the model in Figure 2(b).

**Question 8: Can a team have more than one leader?** Based on the model in Figure 2(a), the answer is indeterminate. Even if we examine association links and find that two employees are team leaders for the same set of members, the answer is still indeterminate. We cannot tell whether we have identified an instance of two different teams with the same members but different leaders, or a single team with more than one leader. Figure 2(b) clearly indicates, however, that a team can have a maximum of one leader.

**Question 9: What is a team?** For the model shown in Figure 2(a), a team exists only when at least one instance of an association exists. The association link
must have an employee playing the member role and an employee playing the leader role. Teams do not have an existence independent of employees. For the model shown in Figure 2(b), however, a team can exist without having either members or a leader. Teams can have a separate existence independent of employees.

**Straightforward Answers**

We have illustrated the difficulties involved in interpreting composites when represented as relationship classes or associations. Even when an answer can be derived to the questions asked about a domain, the logic is often lengthy and involved. Two different readers may reach different answers. The use of an object/entity class to represent a composite, however, provides clear, straightforward answers.

This outcome raises implications for the use of relationship classes or associations to represent composites. One role of a conceptual model is to facilitate communication between users and modelers. Users may be asked to read and validate the model. They may even have to sign off on the model’s validity. If composites are modeled as relationship classes or associations, we predict that the likelihood of users correctly reading and interpreting a model will be low. As a result, errors and omissions may occur during the early stages of information systems development. These types of errors and omissions are the most difficult and expensive to remedy [2].

Another role of a conceptual model is to provide a basis for design. For instance, database designers may use a conceptual model to design the table structure of a new database. Unless they fully understand the semantics of the real world that the conceptual model represents, they might unwittingly misinterpret or even completely miss business rules regarding the composite. As a result, the database design might be incomplete or erroneous. Errors that occur during the database design phase are also expensive to fix if they are not detected quickly.

Conceptual models may also be used during database maintenance activities. A database administrator may refer to the model to gain a holistic understanding of the real-world phenomena that are represented in the database. Models that use relationship classes or associations to represent composites will undermine the database maintainer’s ability to understand the database. They complicate the maintenance process and make it more error prone.

End users of a database may also employ conceptual models to help them formulate queries or updates against a database. If they cannot understand the real-world semantics of composites because they have been modeled as relationship classes or associations, however, they may interrogate or update the database incorrectly. Moreover, they may not realize their actions are erroneous.

Experienced conceptual modelers might argue that common sense suggests that relationship or association constructs should not be used to represent composites. For pedagogical reasons, however, we ought to have theoretical, empirical, and practical evidence to justify this conclusion. Moreover, disciplines advance when commonsense conclusions are codified and provided with a substantive rationale.

Ontologists and other researchers have also shown the semantics of composites and components can be far more complex than the semantics we have considered in our previous examples (see [5, 9, 10, 12]). Eventually we must be able to model faithfully the full range of semantics that can arise with composites and components. Otherwise, we will undermine our ability to implement them successfully in databases, knowledge bases, and information systems. In our view, this need makes the case for modeling composites as entity classes or object classes rather than relationship classes or associations even more compelling.

**References**


This research was supported by a grant from the Australian Research Council.

**Graeme Shanks** (Graeme.Shanks@infotech.monash.edu.au) is a professor in the Faculty of Information Technology, Monash University, Melbourne, Australia.

**Elizabeth Tansley** (e.tansley@cqu.edu.au) is a lecturer in the Faculty of Informatics & Communication, Central Queensland University, Rockhampton, Australia.

**Ron Weber** (Ron.Weber@infotech.monash.edu.au) is Dean of the Faculty of Information Technology, Monash University, Melbourne, Australia.

© 2004 ACM 0001-0782/04/0700 $5.00