Domain modelling pattern: 'Three-party pattern'[1]

Lotte De Rore, Monique Snoeck, Guido Dedene
MIS Group, Dept. Applied Economic Sciences, K.U.Leuven,
Naamsestraat 69, 3000 Leuven, Belgium
{lotte.derore, monique.snoeck, guido.dedene}@econ.kuleuven.be
Tel: +3216/32.68.81 Fax: +3216/32.67.32

Introduction

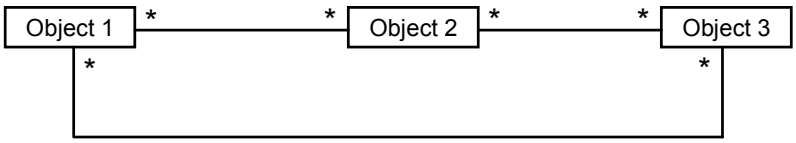
Conceptual domain modelling (or business modelling) is concerned with capturing requirements from the real world and converting these into a model that can be used as part of an information system model. Domain modelling patterns can be domain specific or domain independent. In the latter case, the pattern is formulated in an abstract way in terms of generic objects. When applying the pattern, those generic objects are then mapped to specific objects of the domain at hand. In this paper we present a domain independent pattern that applies to a situation where you have three business objects which can be tangible or intangible objects from the real world and that need to be related with each other. The three business objects are potentially related by three binary associations, but a ternary association relating the three business objects at once could be considered as well. Your problem is to decide on which associations to include in your model and to capture potential interferences between the different associations.

Name

Three-party pattern.

Context

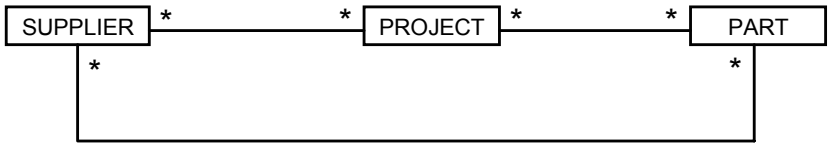
Suppose you start from three business objects which can be tangible or intangible objects from the real world. These entities are potentially related by three binary associations. Examples of such three related entities could be [PROJECT, PART, SUPPLIER], or [EMPLOYEE, PROJECT, DEPARTMENT] or [PERSON, TASK, TOOL/RESOURCE] or [PLAYER, TEAM, TOURNAMENT]. The binary associations can represent factual relationships among two of the basic objects (e.g. PLAYER *is-part-of* TEAM), can keep track of history and duration of the relationship (e.g. EMPLOYEE *has been assigned to* DEPARTMENT), or keep track of "authorised" relationships (e.g. TOOL *can be used for* TASK, PRODUCT *can be acquired from* SUPPLIER, EMPLOYEE *is capable for* TASK, EMPLOYEE *is authorized for* TASK).
Each of the three business objects has potentially a life cycle during which the business object evolves through a number of states. The state in which a business object is, determines which association instances can be created, modified or deleted.



[1]©KULeuven. *Paper Submitted for EuroPlop 2005. This material is presented to ensure timely dissemination of scholarly and technical work. Copyright and all rights therein are retained by authors or by other copyright holders. All persons copying this information are expected to adhere to the terms and constraints invoked by each author's copyright. In most cases, these works may not be reposted without the explicit permission of the copyright holder.*

Example

Assume you have the Business Objects PROJECT, PART and SUPPLIER. The association between PART and SUPPLIER tells you which parts are supplied by which supplier. The association between PART and PROJECT tells you which part has been used in which project. And the association between PROJECT and SUPPLIER tells you which supplier has delivered something for which project.



PROJECTs can be *planned*, *in progress* or *closed*. Only when a project is in the state *planned* or *in progress*, new parts and suppliers can be assigned to a project.

Forces

The main force behind this pattern is 'conceptual modeling quality'. According to Lindland & Sindre [2], this quality can be split up in syntactic quality, semantic quality and pragmatic quality. Syntactic quality relates to the modeling language, which is UML in this paper. Semantic quality relates the model to the domain by considering not only syntax, but also relations among statements and their meaning. Pragmatic quality relates the model to audience participation by considering not only syntax and semantics, but also how the audience (anyone involved in modeling) will interpret them. In this pattern, the focus is mostly on semantic quality. In [2] semantic quality is subdivided in following less abstract attributes:

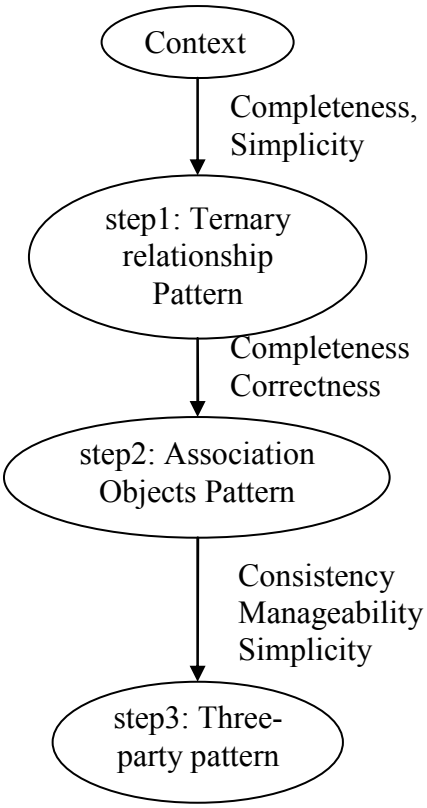
- Completeness: the model contains all the statements about the domain that are correct and relevant.
- Consistency: the model contains no contradicting statements.
- Semantic correctness: each statement made by the model is true in the real world.

In addition the following forces can be considered as well:

- Minimality or simplicity: Although the model should be complete, it should not contain more classes and associations than really necessary. In particular, redundant associations between classes have to be avoided as they can compromise the consistency of the model.
- Instance manipulation (insertion, update and deletion of class and association instances): When inserting, modifying or deleting an instance of a class or association, the model should provide the necessary information to validate these instance manipulation.

A Step by Step Solution

The solution to the problem of modelling the associations between the three business objects is presented in a set of consecutive steps. Starting from the initial context, the different forces will guide us step by step to the final solution.



Step 1a: the need for a ternary association

Problem

Suppose you have 3 business objects. You wish to capture the relationship between the three objects. The binary associations between each pair of business objects gives you part of the information you need, but the combination of this partial information does not satisfy your information needs.

Example

Assume you have the business objects PROJECT, PART and SUPPLIER with the binary associations as above. If one part X can be supplied by many suppliers (say A and B) and be used in several projects (say P1 and P2), and both suppliers supplied some parts for both projects, then the three individual associations do not tell you which supplier supplied part X for a particular project.

Forces

The main force here is the completeness of the conceptual model: not all the statements of the real world are yet captured in the model with only 3 binary associations.

Solution

The only way to have this information is by having a ternary association relating the three business objects at once.

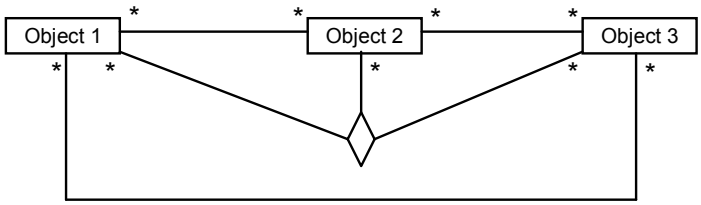


Figure 1: Ternary Relationship pattern

Step 1b: needing the binary associations as well

Problem

Suppose you have the situation as above: 3 business objects, 3 binary relationships and one ternary relationship. You can ask now the question whether the 3 binary associations are still necessary. Maybe the ternary association is enough to capture all the information.

Forces

You now face a trade-off between completeness and simplicity:

- Minimality or Simplicity dictates to keep the model as simple as possible. If the ternary association gives you all the information you need, the binary associations should be removed from your model.
- Completeness: On the other hand, all statements about the real world should be reflected in the model and removing the binary association may lead to an incomplete model.

Additionally, the model should provide the necessary support for instance manipulation:

- Instance insertion: when inserting a new instance of the ternary relationship, you face the problem that not *any* triple of business object instances represent a valid instance of the ternary association. Are there then enough validation possibilities to see whether this is a valid triple in the real world? Are the binary relationships necessary or not to validate these insertions?

Solution

In order to keep information consistent at any time, you need to keep the binary associations that define the valid instances of the ternary association. The instances of the binary associations can now be used to validate the creation of instances of the ternary association.

Example

The triple (project: P1, Part: X, Supplier: A) is only valid if part X is required by project P1, supplier A is capable of supplying part X and Supplier A is an authorized supplier for project P1. Before creating this triple, you can now verify whether (PROJECT: P1, PART:X) is a valid/existing instance of the association between PROJECT and PART and whether (PROJECT:P1, SUPPLIER: A) is a valid/existing instance of the association between PROJECT and SUPPLIER and whether (SUPPLIER:A, PART:X) is a valid/existing instance of the association between SUPPLIER and PART.

Step 2: Managing history, evolution and life cycles of associations.

Problem

At the end of Step 1, our model contains the three business objects, three binary associations and one ternary association. Each association instance represents a link between two or three

business objects. But those associations instances change over time and the history of each association is in some cases considered to be valuable information. In addition, the nature or cardinality of a relationship might change when it is considered 'over time' instead of at one point in time. Finally, associations are a kind of intangible objects that may have behaviour of their own which cannot be modelled as part of the life cycle of the business objects and can neither be included in the concept of association itself. In summary, the associations give valuable information, but there are potential additional requirements for history, evolution over time and life cycle information.

Example

Suppose you have the 3 business objects PROJECT, PART and SUPPLIER related by a ternary association and two-by-two related by three binary associations as described above. If you examine the current state of an association, this association is often very simple: e.g. a supplier delivers parts, parts are used for projects and suppliers delivers for projects. Over time, however, a supplier that used to deliver a part may not deliver this part anymore. And a supplier which used to be authorized for a project may not be authorized anymore. As another example, suppose that each part has *one* supplier at one point in time. Over time, this supplier can change and thus a part has *many* suppliers.
Finally, the use of a part from a supplier for a project can go through a complete acquisition process resulting in a life cycle with several consecutive states: planned, delivered, invoiced, paid, used. The stages of this part usage cannot be modelled as stages in the life cycle of the individual project or supplier or part.. Similarly the supplier-part relationship can have a life cycle as well: parts can for example be temporarily out of stock. This being out of stock is neither a property of the supplier or the part. It is a property of the association between Supplier and Part.

Forces

The main forces here are completeness and correctness:

- *History*: If a relationship changes, for example a project changes from supplier, you loose the information about the former supplier. For reasons of completeness of the model, you may want to keep track of history. Similarly, if you want to check whether a supplier was authorized for a project at some point in time, but that relationship has already changed in the mean time, the model will give you the wrong information. Here also, keeping track of history should give a solution.
- *Association life cycle*: for reasons of completeness of the model, you may want to provide the association with extra attributes that allow keeping track of the state an association is in.
- *Changing nature of associations and correctness of the model*: If an association end has a cardinality of *one* at one point in time, this cardinality may become *many* if you decide to keep track of the history of associations. You then face the problem that you nevertheless need to ensure that at one point in time there can be only one active association instance.

Solution

First, define with the help of the Association Pattern [1] three (binary) association objects, one for each of the binary associations, capturing date and time of the associations. These binary association objects each collaborate with the two business objects related by the association. Determine attributes and operations unique to these associations.
Secondly, define one ternary association object that captures the information, date and time of the ternary association. This ternary association object collaborates with the three business objects.
Now you can keep track of the history of the associations.

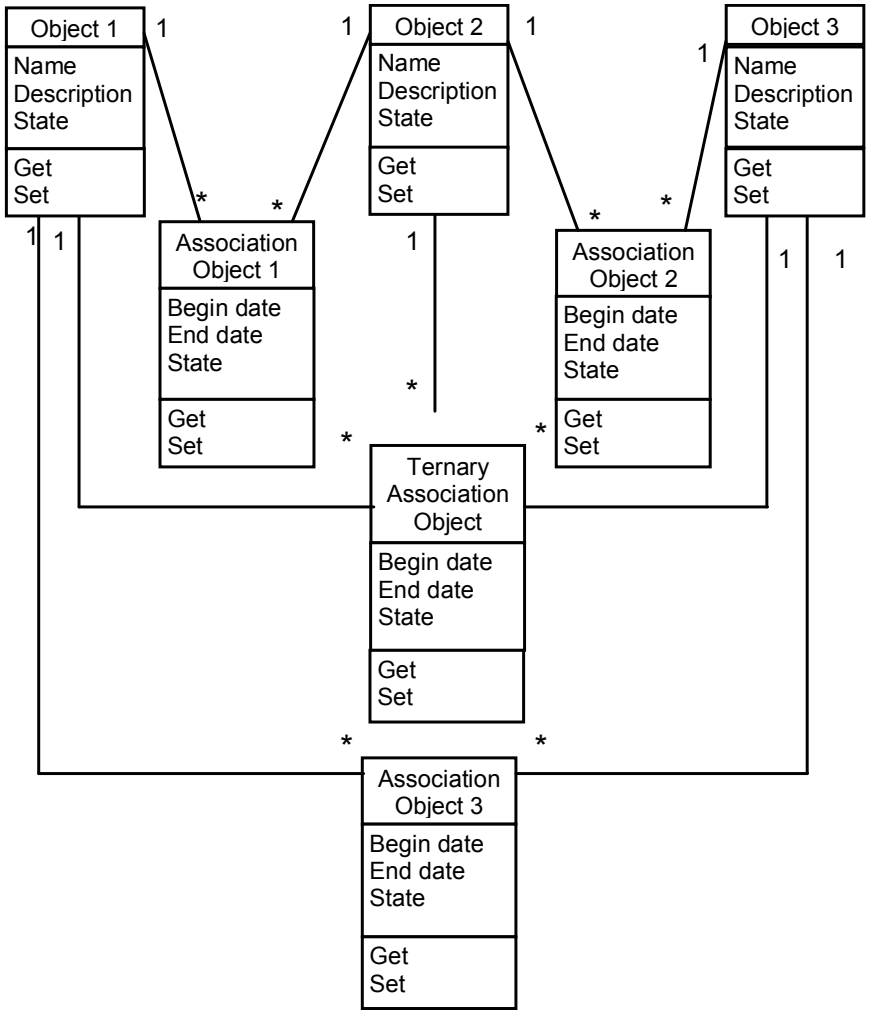
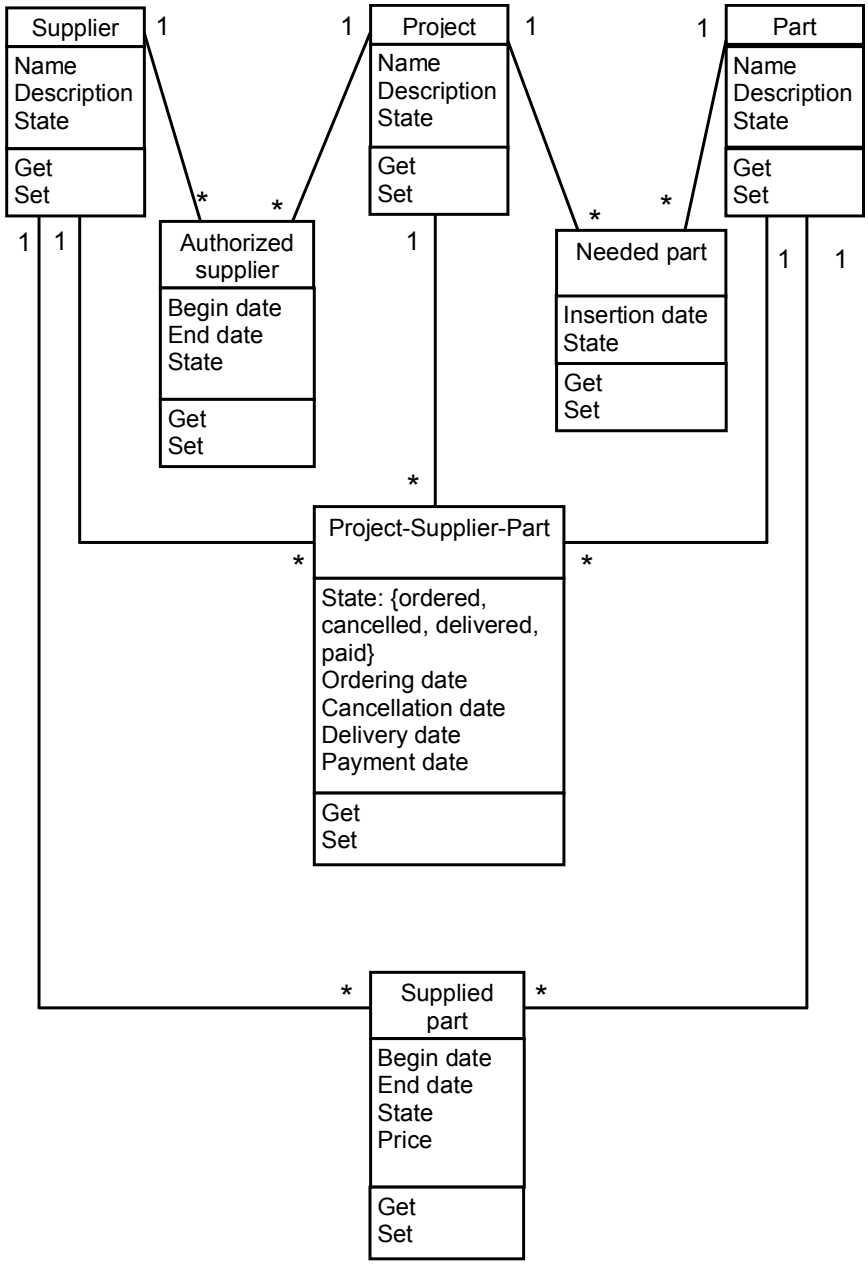


Figure 2: Association Objects Pattern

The key point is that keeping the dates allows discerning between active and historical instances of the associations: if the end-date is in the past, the association instance is a historical one. If the begin-date is in the past and the end-date is in the future, the instance is the active one. If at one point in time there should be only one active association instance, this can be enforced by setting the end-date of the last active instance to a past date every time a new instance is inserted. Possibly, the begin-date can be in the future as well, indicating a planned association. By adding a state attribute as well, the association object can also account for association life cycles. Thus, the association object integrates history and life cycle in one object and is able to deal with changing nature of associations while keeping the model correct.

Solution example

If the 3 binary associations and the ternary associations are replaced by the association objects (AUTHORIZED SUPPLIER, NEEDED PART, SUPPLIED PART and PROJECT-SUPPLIER-PART) you can keep track of the history of these associations.



The system keeps track of a set of authorized suppliers per Project. The authorization is the result of an evaluation on a per project basis and is only valid from the given authorization date onwards. No parts can be ordered from this supplier prior to this authorization date. In certain cases, the authorisation can be withdrawn prior to the end of the project. For each Project a list of needed parts is registered. Apart from the date of adding a part to the list, no historical information is required. The dates in the SUPPLIED PART object allow registering price evolution over time: an instance with a past end-date refers to a previous price and the instance with a void end date refers to the current price of the part with this supplier. Per supplier and part there should be at most one current price association instance.
The Project Supplier Part object: keeps track of the ordering of parts from a particular supplier for a specific project. The status attributes and dates allow keeping track of the successive steps in the ordering process.

Step 3: Three-party pattern

Context & Problem

Given the model as described above with 3 business objects, 3 binary association objects and a ternary association object between the 3 business objects, we still face the problem that as the information contained in the binary associations constrains the valid instances of the ternary associations, history and life cycle of the binary associations should be kept consistent with the history and life cycle of the ternary association and vice versa. As an example, relevant pieces of history of the binary associations should not be destroyed as long as it is required for the correct interpretation of the information contained in the ternary association.

Although there is some strong relationship between the information contained in the binary association objects and the ternary association object, these objects are only indirectly related through the business objects. The question to ask here is whether it is advisable to add associations between the ternary association object and the binary association objects.

Example

In our example, the information contained in the binary association is strongly related to the information contained in the ternary association.

- If the use of a part is planned for a project, then this part should be provided by and not be out of stock at the supplier.
- Price information in the SuppliedPart object should not be destroyed as long as instances of the Project Supplier Part association object exist that refer to that piece of price-history.
- Parts should not be removed from the needed-part list if such part has been ordered for the project.
- For a project, a part can only be ordered from a supplier on a date *after* the authorisation date of the supplier for that project. Authorisation information must be kept as long as information on ordered part is kept.
- …

Forces

The main forces here are consistency, manageability, and ease of instance manipulation:

- Consistency and manageability: If the ternary association is collaborating with the three business objects, the ternary association object has no direct access to the information that constrains the validity of its instances. A direct link between the binary and ternary association objects would facilitate the management of the consistency.
- Ease of instance insertion, deletion and update: A direct link would for example allow for referential integrity checks between instances of the ternary association and relevant instances of the binary associations. Also the validation of state changes and insertion of new instances would be simplified. Now more complex code is required for validating creation or evolution of ternary association instances.
- On the other hand, adding additional associations on top of the current model increases the complexity of the model significantly, with the risk of making the whole model unmanageable.
- The forces Simplicity and minimality require us to use as little associations and objects as possible, while keeping the model complete and consistent.

Solution

Rather than adding additional associations, the existing relationships between all objects are reorganised. Instead of letting the ternary association object collaborating with the business

objects directly, it should collaborate with each of the three binary association objects, as shown in the diagram below.

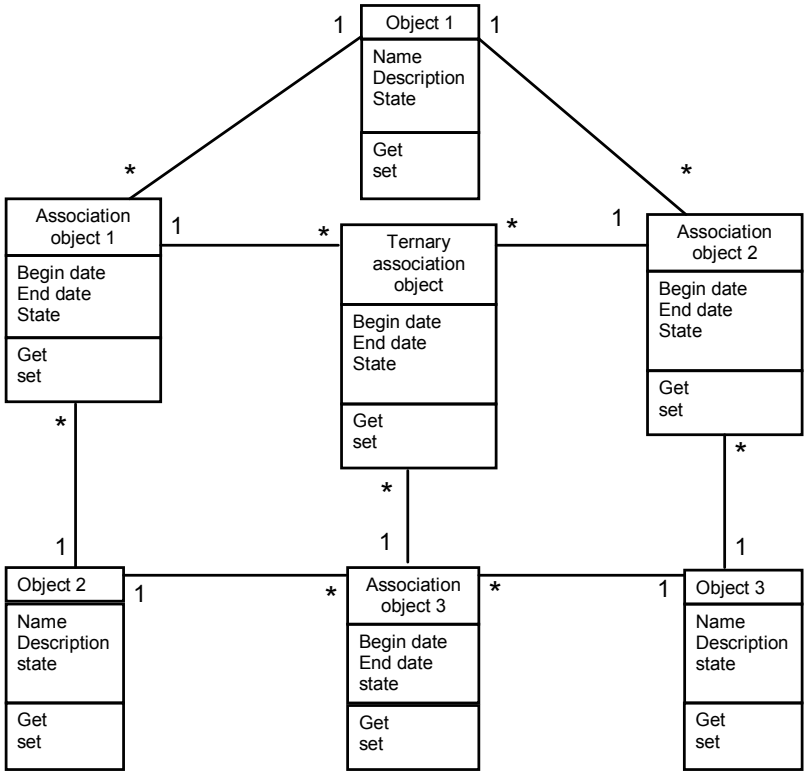


Figure 3: Three-party pattern

The final model has now the following Participants & Responsibilities:

- 3 Business Objects:
  - responsible of managing information and life cycles about the Business Objects
- 3 Binary Associations:
  - responsible of managing factual relationships among 2 of the Business Objects
  - keeping track of history and duration of the binary relationship
  - keeping track of information on 'allowed' ternary links
  - keeping life cycle (status) information on the binary associations.
  - ensuring that transition from one state to another of association instances is validated against the life cycle status of the participating business objects.

- Ternary Association:
  - responsible of managing the factual relationships where the three Business Objects collaborate together.
  - keeping track of the history and duration of the ternary relationship
  - keeping life cycle (status) information on the ternary association.
  - collaborating with the three Binary Association Objects to take into account the existing factual relationships and allowed relationships between 2 parties to validate insertions, updates and deletion of instances.
  - ensuring that transition from one state to another of ternary association instances is validated against the life cycle status of participating the binary association instances.

Pattern Instantiation and Variants

When applying the pattern to a given situation, the three business objects have to be mapped to the objects of the domain under consideration. If the need for a ternary association is conformed (step 1a), then variations can occur in the application of the remaining steps.

Variations can occur by considering if all three binary associations are required or only one or two of them, and by considering for which association object history and life cycle information must be kept.

A further variation point is the fact whether or not the binary associations are constraining the allowed instances of the ternary association. If you consider the situation of [PERSON, UNIT, TASK TYPE] as shown in Figure 4 and suppose that the unit who asks for the task does not need to be the same unit as the unit whereto a person belongs, then the binary association PERSON-UNIT is not constraining the ternary association. So in this case, all three binary associations are kept, but only two of them act as constraining towards the ternary association:

___: When a task assignment is registered, the unit of the person who has to execute the task may or may not be the same as the unit which is responsible of the task.
… : When a task assignment is registered for a Task Responsibility, the task type should match the task type of the task capability.
---: When a task assignment is registered, the unit of the person who has to execute the task should have the capability to execute this task.

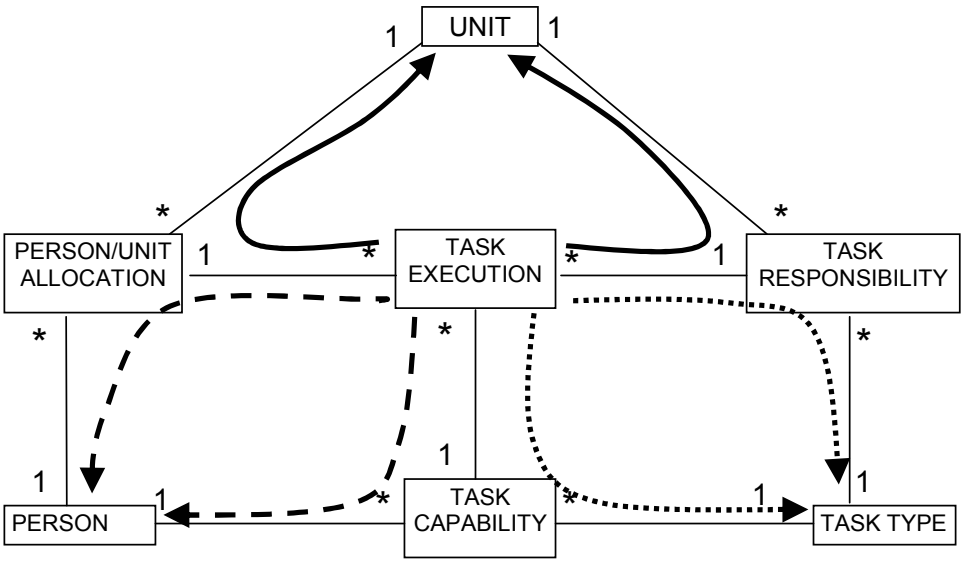


Figure 4: [PERSON, UNIT, TASK TYPE]

Variants of the pattern are also created by dropping one or two of the binary associations, by merging one of the binary associations with the basic entity or by adding extra binary associations.
Considering the situation of [PLAYER, TEAM, TOURNAMENT-DAY] as in Figure 5, one can see that the binary association between PLAYER and TOURNAMENT-DAY has been dropped and that the binary association between TEAM and MATCH is doubled because two teams are involved in a match, namely the home players and the visitors.

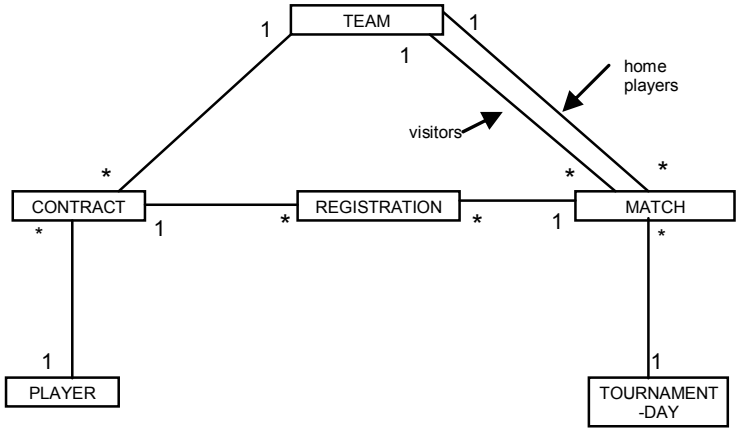


Figure 5: [PLAYER, TEAM, TOURNAMENT-SLOT]

An other example is the situation of an alumnus-organization, [ALUMNUS, GROUP, ACTIVITY], see Figure 6. There are two kinds of memberships with different properties: committee membership and regular alumnus membership. So you get two association objects between GROUP and ALUMNUS. As each group organises its own activities, as can be seen in Figure 6, there is a 1-1 mapping between ACTIVITY ORGANISATION and ACTIVITY. The binary association object can therefore be merged with the basic business object as shown in Figure 7.

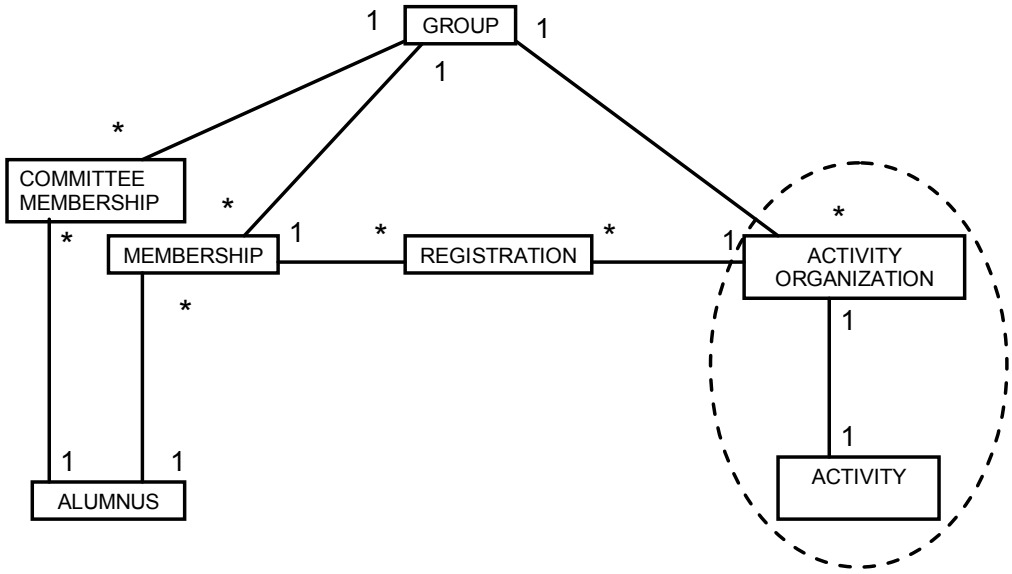


Figure 6: [ALUMNUS, GROUP, ACTIVITY TYPE]

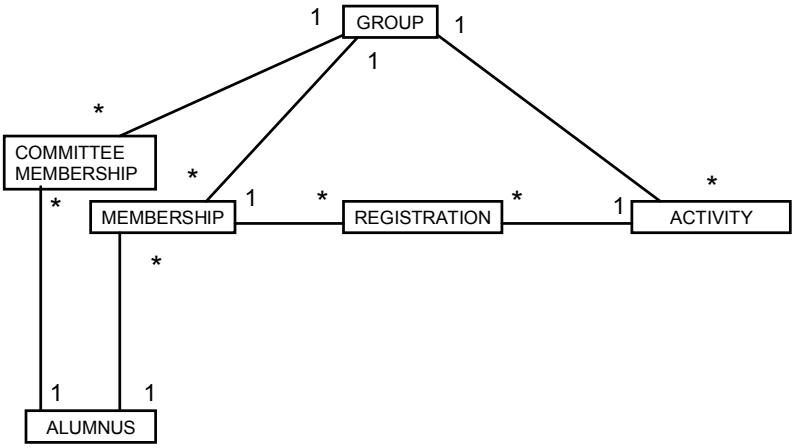


Figure 7: [ALUMNUS, GROUP, ACTIVITY]

Conclusion

The art of conceptual modelling is a difficult one. A conceptual model needs to capture all relevant statements from the universe of discourse in a correct way, using a minimum of constructs. At the same time its pragmatic quality must be kept as it is often used as vehicle for communicating with end users. The Three-Party Pattern meets these quality goal by arranging basic business objects, binary and one ternary association objects in such a way that interactions between the history and life cycles of all those objects can more easily be managed.

Related Pattern

Lorraine L. Boyd's Business Pattern of Association Objects [1]

Acknowledgements

Many thanks to Jeroen Geleyn who validated this pattern against several real life cases in the context of his term paper and to Klaus Marquardt who shepherd this paper from a rough idea to its current form.

References

[1] LL. Boyd: Business Patterns of Association Objects. Pattern Languages of Program Design 3, Chapter 23 (1998)

[2] O.I.Lindland, G. Sindre, A. Solvberg: Understaning Quality in Conceptual Modeling. IEEE (March 1994)