Conceptual Patterns – a Consolidation of Coad's and Wohed's Approaches

Petia Wohed

Department of Information and Systems Sciences
Stockholm University/Royal Institute of Technology
Electrum 230, 164 40 Kista, Sweden
petia@dsv.su.se

Abstract: The information system analysis process is considered as a difficult phase during the information systems development. The difficulty lies in gathering relevant information from the domain experts. Different techniques for supporting this process have been developed. One of them is provided by Coad, who defines a number of patterns aimed for use during the analysis of an information system. Another one is the work provided by Wohed on automating the information gathering process, which so far is implemented for one domain only. In this paper these two approaches are consolidated in order to continue the work provided by Wohed and extend it into a domain independent effort.

1 Introduction

One of the main activities during the development of an information system is the information analysis, which gives input for the design of the database behind a system. To support information analysis, the conceptual modelling technique has widely been used. The goal of conceptual modelling is to derive a model covering the relevant aspects of the underlying universe of discourse (UoD). The difficulty for the designers of an information system lies in capturing and extracting the relevant information from people working within and knowledgeable about the UoD that is analysed.

To support this process different techniques, supporting the communication between domain experts and information systems developers, have been developed. One of them is the representation of a conceptual schema produced by system developers in a natural language [4],[2] in order to make the schema readable and understandable to domain experts, who will then be able to validate it.

Another approach is the verbalization technique as described by Hofstede in [1]. This technique builds on the assumption that a verbalization of the samples from the UoD gives the structure and rules of the UoD. The point of the departure is a description of the communication in the UoD to be modeled, which builds the set of sample sentences, and the output is a conceptual schema.

Similarly, a description of a system is required for the technique developed by Purao and Storey [8],[9], which provides intelligent support for retrieval and synthesis of patterns for object-oriented design. The required input for this technique is a natural language sentence describing the aim of the system. Then, after applying techniques for natural language processing, automated reasoning, learning heuristics, pattern retrieval from a patterns library, and pattern synthesis a conceptual model for the UoD is received.

The differences between the two approaches producing conceptual schemas described above are: firstly, the access to a pattern library in Purao and Storey's approach; and secondly the kind of the description, which is used as input: a set of sample sentences in the first case, and a sentence describing the aim of the system in the second one. However, both these approaches start with a description. If such description is not available the approaches can not be applied. An alternative, for such situations, should be the approach proposed by Wohed [10], which does not require any particularly prepared input. Instead, information for the modeled domain is gathered by posing a number of predefined questions to a domain expert. In order to automate the modeling process, similar to Purao and Storey's approach, Wohed's approach builds on the access to a pattern library from which the suggested solutions are selected.

The work provided by Wohed has so far been concentrated on one domain only. The work provided here is a continuation of it. It attempts to generalize the predefined questions in order to make them domain independent. Such a generalization is suggested after consolidation of the approach with the work provided by Coad [3], who extracts and defines a number of generic conceptual patterns.

The paper is organized as followed. Section 2 gives a brief description of the notation used in the paper. In Sections 3 and 4 the approaches proposed by Wohed and Coad are described. Section 5 presents the consolidation of the approaches. In section 6 a suggestion for the generalization of the questions is presented and some loss of information is outlined. Finally, Section 7 summarizes the paper and gives directions for further work.

2 Conceptual Schemas

In this section we briefly introduce the modeling language which is used. A formal definition of it, may be found in [7].

The basic construct in conceptual modeling approaches is the object. Objects that are similar to each other are grouped together into object types, such as Person and Country. Objects have different properties, which are modeled by attributes, and they can be related to each other, which is represented by relationships. In our graphical notation (see Figure 1) object types are represented by rectangles, attributes by lists inside the object types, and relationships by labeled lines. A direction of each line is given in its label. The object type initiating a relationship is called the domain of that relationship and the object type in its end is called the range. Generalization constraints are shown by arrows where the head of an arrow points to the super-type. For each relationship, the mapping constraints, represented by a quadruple <1m,1m,tp,tp>, specify whether it is single-valued, injective, total or surjective. A relationship is single-valued, denoted by 1 in the first position of the quadruple, when each instance of its domain is connected to at most one instance of its range. A relationship that is not single-valued is multi-valued, denoted by m. A relationship is total, denoted by t on the third position, when each instance of its domain is connected to at least one instance in its range. A relationship that is not total is partial, denoted by p. A relationship is injective (surjective) when its inverse is single valued (total). The second and fourth positions in the quadruple are reserved for the injective and surjective properties, correspondingly.

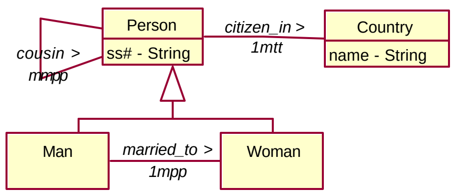


Figure 1 An example of a conceptual schema

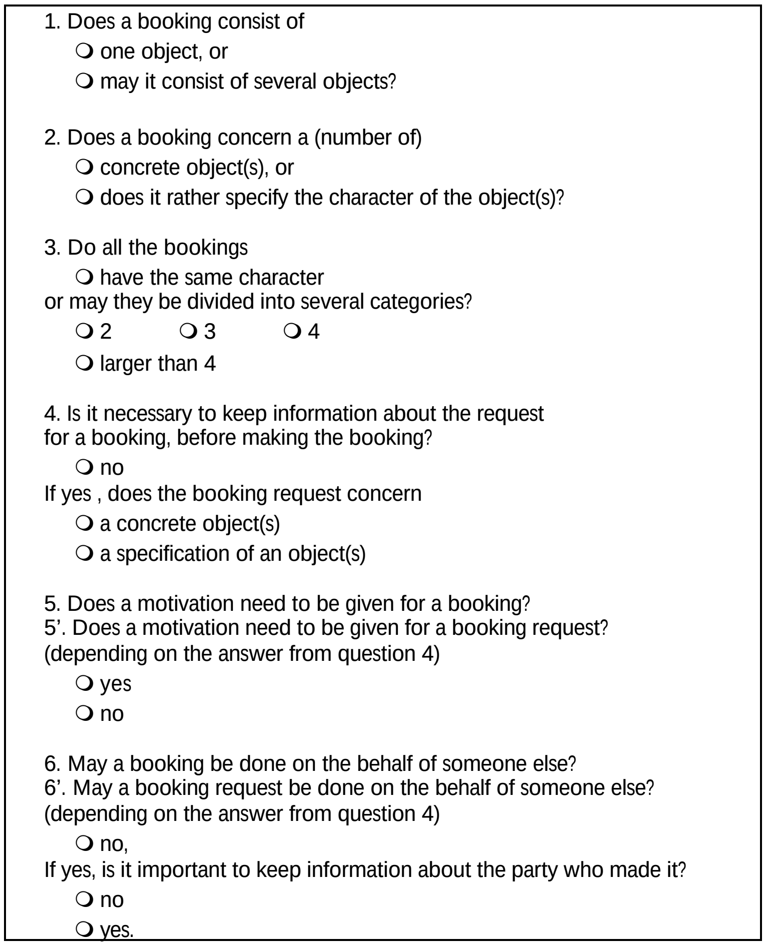
The description of the domain represented by the model in Figure 1 is as follows: a Person is a citizen_in a Country and (s)he may have several cousins. The distinction between Man and Woman is kept. A person may only marry_to someone with the opposite sex, and polygamy is only allowed for women.

3 The Modeling Wizard Tool

The conceptual modeling wizard is based, as can be seen from its name, on the concept of a wizard tool, i.e., a tool gathering information from the user by asking her (him) a number of questions and suggesting a solution tailored to the set of the received answers. Applied to the area of conceptual modeling, this wizard poses questions about a domain and suggests a conceptual schema for this domain (a detailed description of the tool can be found in [10]. So far the wizard supports the booking domain only.

Six questions were collected and implemented in the tool (see Figure 2). The questions were identified after a number of different solutions, collected from Hay [6] and Fowler [5], and completed by other relevant solutions for the booking domain, were analyzed. They were aimed to cover the differences between different booking situations so that a satisfactory solution could be identified. A brief description of the questions is given below.

The first question is about the cardinality of a booking, i.e., where a booking may consist of several booking lines (e.g. when booking a trip, both tickets and hotel rooms may be reserved), or not (usually when booking a rental car). The second question is about whether a booking concerns a (number of) concrete objects (like when booking a time to dentist) or not (e.g. when booking a book in a library, not a particular exemplar of a book, but rather a title is booked). The third question investigates whether the bookings made within a system have the same character (as is the case for cinema tickets bookings) or whether they may differ. (In the trip booking example above, tickets bookings require departure and destination places and time for departure, whereas hotel room bookings require arrival and departure times and number of beds.) The fourth question is aimed to clarify whether or not it is necessary to keep information about the request for a booking. (For instance when scheduling room bookings for university courses, booking requests are collected from the head teachers for each course and used as input in the scheduling work.). The fifth



1. Does a booking consist of
   - ❍ one object, or
   - ❍ may it consist of several objects?

2. Does a booking concern a (number of)
   - ❍ concrete object(s), or
   - ❍ does it rather specify the character of the object(s)?

3. Do all the bookings
   - ❍ have the same character

or may they be divided into several categories?

   ❍ 2 ❍ 3 ❍ 4

   ❍ larger than 4

4. Is it necessary to keep information about the request for a booking, before making the booking?
   - ❍ no

If yes , does the booking request concern
   - ❍ a concrete object(s)
   - ❍ a specification of an object(s)

5. Does a motivation need to be given for a booking?
5'. Does a motivation need to be given for a booking request?
(depending on the answer from question 4)
   - ❍ yes
   - ❍ no

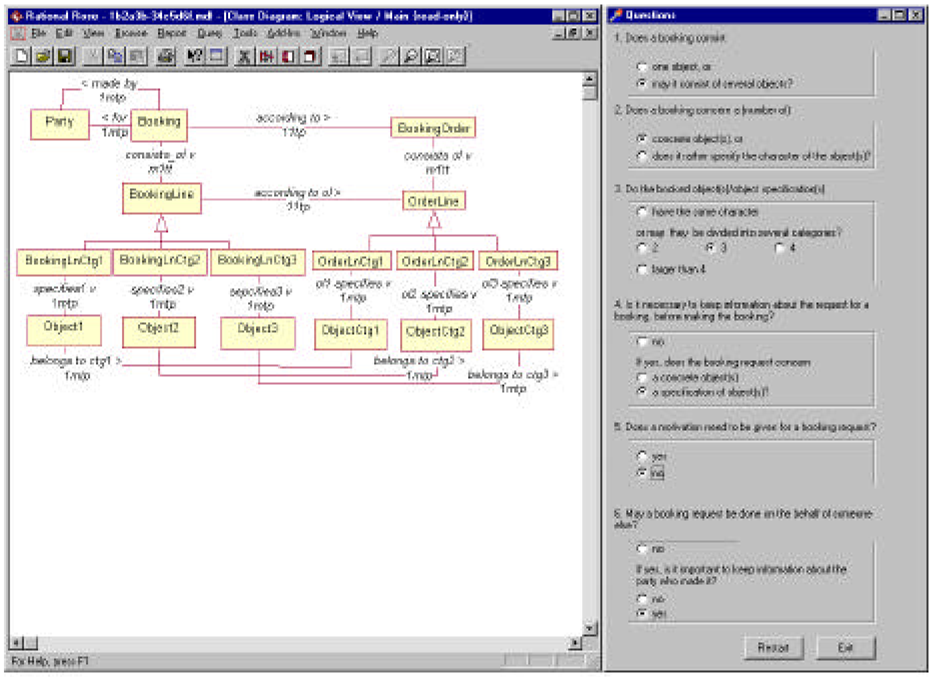
6. May a booking be done on the behalf of someone else?
6'. May a booking request be done on the behalf of someone else?
(depending on the answer from question 4)
   - ❍ no,

If yes, is it important to keep information about the party who made it?
   - ❍ no
   - ❍ yes.

Figure 2 The questions implemented in the tool

and sixth questions depend of the answer to the fourth question. If booking requests are necessary to keep information about, the fifth and the sixth questions are posed for the booking requests, otherwise the questions are posed for the bookings. The fifth question asks if a motivation (a purpose) for a booking/booking request is necessary. (For the university scheduling work each head teacher gives the course he/she will make the bookings for.) The sixth question asks whether a booking/booking request may be done on behalf of someone else (e.g., a secretary may book the business trips for her/his boss).

For the sake of user-friendliness the questions are placed in sequence by showing a new question only when the previous one has been answered. A conceptual schema solution is gradually built and refined, and graphically presented after each new answer. Figure 3 shows a solution suggested by the tool according to the answers in the right hand side of the figure.



1. Does a booking consist

( ) one object, or
[✓ may it consist of several objects?]

2. Does a booking concern a (number of)

[✓ concrete object(s), or]
( ) does it rather specify the character of the object(s)?

3. Do the booked object(s)/object specification(s)

( ) have the same character

or may they be divided into several categories?
( ) 2 [✓ 3] ( ) 4
( ) larger than 4

4. Is it necessary to keep information about the request for a booking, before making the booking?

( ) no

If yes, does the booking request concern
( ) a concrete object(s)
[✓ a specification of object(s)?]

5. Does a motivation need to be given for a booking request?

( ) yes
[✓ no]

6. May a booking request be done on the behalf of someone else?

( ) no

If yes, is it important to keep information about the party who made it?
( ) no
[✓ yes]

Figure 3 A snap shot from the tool

Initially, the object types Booking and Parties and the relationship for are introduced. This is done because of the totality of this relationship. The answer from the first question, results in the introduction of the object types Booking and BookingLine and the relationship between them. The answer from the second question causes the introduction of an object type called Object, which can not be seen in the final solution, due to changes performed when answering the rest of the questions. The answer from the third question results in specializing BookingLine into BookingLnCtg1, BookingLnCtg2 and BookingLnCtg3. The object type Object, introduced after the previous answer, is divided into Object1, Object2, and Object3. The first part of the fourth question results in the introduction of the object type BookingOrder. The second part results in the introduction of the object types for different object categories (ObjectCtg1, ObjectCtg2 and ObjectCtg3). Beside, the structure of a Booking is mirrored in the structure of the BookingOrder. If the answer from the fifth question is positive a further object type called Motivation should be introduced, but since it is negative, no changes are done. Finally, according to the answer from the last question the relationship made_by between Booking and Party is introduced.

4 Coad’s transaction patterns

One of the earliest contributions on patterns in information systems analysis was provided by Coad, who in 1995 published “Object Models: Strategies, Patterns, and Applications” [3]. The book presents 148 strategy steps for building systems and 31 object-model patterns for building object models and demonstrates their applicability by using them when building five different systems.

The characteristic of Coad’s 31 patterns is that they are very small and generic, usually consisting of no more than two object types and a relationship between them. They do not address a particular problem, but are rather general and may be considered as building bloks of a schema.

Coad divides 30 of his patterns in the following four categories: transaction patterns; aggregate patterns; plan patterns; and interaction patterns. All these patterns follow a template, defined through the first pattern called the fundamental pattern (see Figure 4)

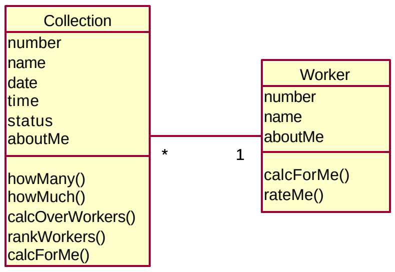
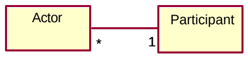


Figure 4 The fundamental pattern - Coad’s first pattern (pattern #1) -

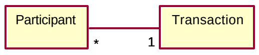
Only the patterns classified as transaction patterns are relevant for this work. An extract of Coad’s own description of them, including the object types’ names, the relationships between them and some of the examples, is presented in Figure 5. Coad’s notation is close to UML. The main difference is the placing of mapping constraints on the opposite side of a relation.

#2 Actor-Participant pattern



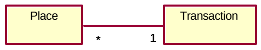
Examples:
Actor: person, organization
Participant: agent, applicant, buyer, client, owner customer, dealer, order clerk, recipient, etc.

#3 Participant-Transaction pattern



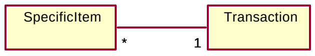
Examples:
Participant: agent, applicant, buyer, client, owner customer, dealer, order clerk, recipient, etc
Transaction: agreement, contract, delivery, inquiry, order, purchase, etc

#4 Place-Transaction pattern



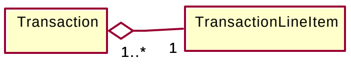
Examples:
Place: geographic entity, store, shelf, etc.
Transaction: agreement, assignment, contract, delivery, inquiry, order, purchase, etc

#5 Specific Item-Transaction pattern



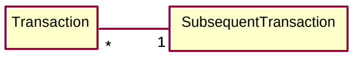
Examples:
Specific item: specific aircraft, specific ship, etc.
Transaction: agreement, assignment, contract, delivery, inquiry, order, purchase, etc

#6 Transaction-Transaction Line Item pattern



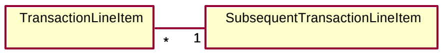
Examples:
order-order line item, sale-sale line item, etc

#7 Transaction – Subsequent Transaction pattern



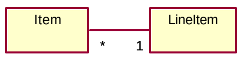
Examples:
order- shipment, reservation-sale, purchase-payment, etc

#8 Transaction Line Item – Subsequent Transaction Line Item pattern



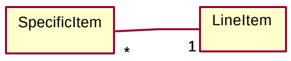
Examples:
order line item – shipment line item, reservation line item –sale line item, etc

#9 Item – Line Item pattern



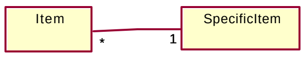
Examples:
item - order line item, item – shipment line item, item – rental line item, etc

#10 Specific Item – Line Item pattern



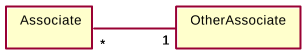
Examples:
videotape - rental line item, etc

#11 Item – Specific Item pattern



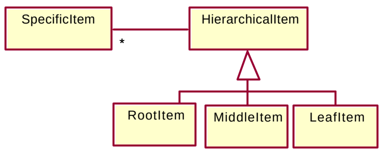
Examples:
job description – specific job, aircraft – specific aircraft, etc.

#12 Associate – Other Associate pattern



Examples:
building-sensor, diver-vehicle, etc.

#13 Specific Item – Hierarchical Item pattern



Examples:
organization – organization description hierarchy

Figure 5 The transaction patterns defined by Coad

5 Relationships between the wizard's questions and Coad's patterns

In the previous two sections the patterns defined by Coad and a modeling wizard tool built partly on patterns presented by Hay and Fowler and partly on additional patterns relevant for the booking domain, have been presented. In this section, the attention is turned into incorporating these two approaches by analyzing the questions implemented in the tool trough Coad's patterns.

Starting with the first question "Does a booking consist of one object or may it consist of several objects", it may be noticed that an answer, claiming that a booking consists of several object, results in the introduction of the Transaction-Transaction Line Item pattern (pattern # 6 from Figure 5) into the solution (instantiated by Booking - BookingLine in the example from Figure 3). An answer claiming the opposite should not give any change. In this way the first question investigates whether pattern #6 shall, or shall not, be included in the solution.

Going further to the second question, investigating whether a booking concerns a concrete object, or rather the description of an object, can be related to Item-Specific Item pattern (#11). Not the whole pattern is applied at this moment, but the question still focus on the distinction made by this pattern. Furthermore, depending on the answers from the first and the second questions together, one of the following patterns are used for extending the solution after answering the second question: Item-Line Item pattern (#9); Specific Item–Line Item pattern (#10) (the one used in Figure 3 where it is instantiated by Object and BookingLine), Specific Item–Transaction pattern (#5); and finally a pattern not defined by Coad, but using his terminology we should refer to it as Item–Transaction pattern.

The third question, asking whether the bookings in the analyzed domain have the same, or different, characters, is related to the Specific Item–Hierarchical Item pattern (#13). It results in specializing the booking object type into sub types when there are different (but no more than four) kinds of bookings.

Continuing analyzing the fourth question, asking whether information about the requests for a booking shall be kept or not, a relation to the Coad's Transaction–Subsequent Transaction pattern (#7) can clearly be outlined. Certainly, the name Transaction – Preceding Transaction pattern should be more suitable, in this particular case, but since the semantic should be the same, no new patterns extending the Coad's set are necessary. In our particular example Transaction-Subsequent Transaction is instantiated by BookingOrder and Booking, correspondingly. Moreover, the second part of the question is a repetition of the second question for the BookingOrder and the analysis provided under the second question may be repeated here. The suggested solution after answering the fourth question depends also on the first and the third question. If the Transaction–Transaction Line Item pattern (#6) has been applied, then the Transaction Line Item–Subsequent Transaction Line Item pattern (#8) is applied now (which resulted in Figure 3 in the OrderLine-BookingLine construction). Similarly the Specific Item-Hierarchical Item pattern (#13) is used depending on whether it has been used in the solution resulting from the third question.

The fifth question asking whether the motivation for a booking (booking request) is necessary or not is related to Associate–Other Associate pattern (#12). Finally the last question investigating the different participants in a booking is related to Participant–Transaction pattern (#3).

Summarizing, we would like to note that all patterns from Coad's library, except Actor-Participant pattern (#2), and Place-Transaction pattern (#4) were covered by the questions (see Figure 6). One reason for not having a question related to pattern #4 could be that the place where a booking is done was not identified as potential important information. However there may exist situations where even the physical place for a booking is necessary to keep information about, which should make it necessary to extend the questions in order to cover this aspect as well. Neither is the pattern #2 covered by the questions in their present state. This can be explained by the fact that, within the booking domain, an actor usually has one role only. In the cases where an actor may have several roles, the connection between the different roles is not considered. In domains where it is usual for the actors to be shared between different roles, and in the cases where the connections between the different roles are important, a question covering this aspect should be necessary to include.

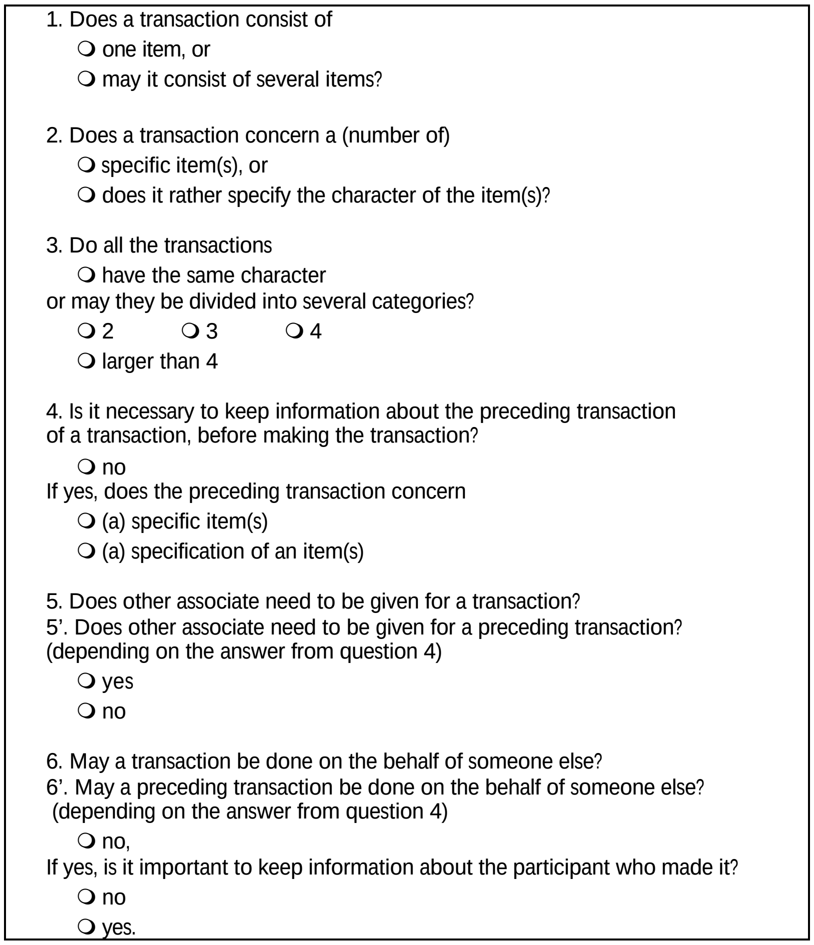
Question number	**Pattern #**
Q1	#6
Q2	#11
Q1 & Q2	#9, #10, #5, and an additional pattern
Q3	#13
Q4 (the second part repeats Q2)	#7 (and the same patterns as for Q2 and Q1 & Q2)
Q1 & Q3 & Q4	#8, #13
Q5	#12
Q6	# 3

Figure 6 The patterns related to each question

Even if the analysis provided above was partly made to reason about the completeness of the questions, i.e., whether all patterns were covered by the questions, it is still not enough for making any conclusion whether the set of questions is complete or not. This is due to the varying number of times a pattern can be applied for building different solutions. To be able to reason about the completeness of the questions according to the iteration of some patterns in a particular solution, some empirical studies have to be provided.

6 Generalization of the Questions

After mapping the wizard's questions into the Coad's patterns successfully, the next step, in the work of making the wizard domain independent, is generalizing the questions. This is done by the support of the terminology defined by Coad, during the construction of the patterns. In Figure 7 the result of this generalization is presented. The following substitutions have been performed: booking $\rightarrow$ transaction (booking is substituted with transaction); object $\rightarrow$ item; concrete object $\rightarrow$ specific item; request and booking request → preceding transaction; motivation → other associate; and finally, party → participant.


1. Does a transaction consist of

❍ one item, or

❍ may it consist of several items?

2. Does a transaction concern a (number of)

❍ specific item(s), or

❍ does it rather specify the character of the item(s)?

3. Do all the transactions

❍ have the same character

or may they be divided into several categories?

❍ 2 ❍ 3 ❍ 4

❍ larger than 4

4. Is it necessary to keep information about the preceding transaction of a transaction, before making the transaction?

❍ no

If yes, does the preceding transaction concern

❍ (a) specific item(s)

❍ (a) specification of an item(s)

5. Does other associate need to be given for a transaction?
5'. Does other associate need to be given for a preceding transaction? (depending on the answer from question 4)

❍ yes

❍ no

6. May a transaction be done on the behalf of someone else?
6'. May a preceding transaction be done on the behalf of someone else? (depending on the answer from question 4)

❍ no,

If yes, is it important to keep information about the participant who made it?

❍ no

❍ yes.

Figure 7 The questions after the substitution

Even if this set of general questions has not yet been empirically tested, it can be observed that some of the semantics we had for the booking domain is lost. For instance, when asking whether the booking requests were necessary to keep information about, or not, it was supposed that the structure for booking requests is similar to the structure of the bookings, i.e., if the bookings consisted of several booking lines, then the requests should also consist of several order lines. It was also supposed that there should be a connection from each booking line to each order line. However, when asking generally if it is necessary to keep information about some preceding transactions, we cannot just presuppose such semantic. Instead, it is necessary to gather information whether such semantics exists by particularly asking about it. Besides, it should also be necessary to ask if it is necessary to keep information about any subsequent, and not only preceding, transactions. These observations indicate that the questions need to be extended in order to make them work for any domain where a transaction is involved. So far the questions work only on domains with the presupposed properties described for the booking domain, e.g., Order-Shipment, Purchase-Payment etc.

7 Conclusions and Further Research

This paper summarizes and consolidates two approaches supporting the information analyses process. The first approach, suggested by Coad, consists of a set of generic patterns to be used during the conceptual modeling process and a set of guidelines for how and when to use these patterns. The second approach, suggested by Wohed, propagates the automation of the modeling process by suggesting a modeling wizard, aimed to gather information about a domain by posing questions to the domain experts and suggesting a solution according to the received answers. The wizard is so far implemented for one domain only, the booking domain. The questions implemented in the wizard were matched to the patterns defined by Coad. This matching was done both to reason about the completeness of the questions and to support their generalization into domain independent questions. During the generalization process some of the semantics relating the questions with each other were lost. Introducing complementary questions for retrieving this semantics is a way to solve this loss of information.

The wizard can in this way be extended to even support other domains where a transaction is involved. Each time the wizard is used it should be tailored to the particular domain it is going to be used for. Such tailoring can be performed by instantiation of the general questions into domain specific ones, going in the opposite direction of the one performed in this paper. It is then necessary to define a set of relevant substitutions for each domain.

After implementing and testing the extensions discussed here, the next step is to further extend the wizard to support other domains then just transaction domains, e.g., to support the modeling process of different product structures. The aggregate patterns defined by Coad could then be used as a point of departure.

Acknowledgments

I would like to thank my advisor Docent Paul Johannesson, my colleague Maria Bergholtz, and the anonymous reviewers for many valuable comments on earlier drafts of this paper.

References

1. A.H.M. ter Hofstede, H.A.Proper, T.P. van der Weide, “Exploiting Fact Verbalisation in Conceptual Information Modelling”, *Information Systems*, vol. 22, no. 6/7, pp. 349-385, 1997
2. M. Bergholtz and P. Johannesson, “Validating conceptual models – utilising analysis patterns as an instrument for explanation generation”, to be presented at the *5th International Conference on Applications of Natural Language to Information Systems*, NLDB’00, Versailles, 2000
3. P. Coad, D. North, M. Mayfield, *Object Models: Strategies, Patterns, and Applications*, Prentice Hall, 1995.
4. H. Dalianis. “A Method for Validating a Conceptual Model by Natural Language Discourse Generation” in Loucopoulos, P., (Ed.), *CAISE-92 International Conference on Advanced Information Systems Engineering*, Lecture Notes in Computer Science, No. 593, pp. 425-444, Springer Verlag, 1992
5. M. Fowler, *Analysis Patterns: Reusable Object Models*, Addison-Wesley, 1997.
6. D.C. Hay*, Data Model Patterns: Conventions of Thought*, Dorset House Publishing, 1996.
7. P. Johannesson, *Schema Integration, Schema Translation, and Interoperability in Federated Information Systems*, Dissertation at Department of Computer and Systems Sciences, Stockholm University and Royal Institute of Technology, Sweden, 1993.
8. S. Purao and V.C. Storey, “Intelligent Support for Retrieval and Synthesis of Patterns for Object-Oriented Design”, in, D.W Embley and R.C. Goldstein., (Eds.), *CAISE-97 International Conference on Advanced Information Systems Engineering, Lecture Notes in Computer Science*, No. 1331, pp.30-42 , Springer Verlag, 1997.
9. S. Purao, “APSARA: A Tool to Automate Systems design via Intelligent Pattern Retrieval and Synthesis”, *The Data Base for Advances in Information Systems*, vol. 29, no. 4, 1998.
10. P. Wohed, “Conceptual Patterns for Reuse of Information Systems Design” to be presented at *CAiSE’00, International Conference on Advanced Information Systems Engineering*, Stockholm, 2000.