

Re-engineering Legacy Applications

Deriving Class Specifications from Structured Analysis Specifications

insight

Naveen Gunti | June 2006



Object-Oriented Analysis (OOA) has made steady progress as a requirements analysis method, in its own right, and as a complement to other analysis methods. Instead of examining a problem using the classic input-processing-output (information flow) model or a model derived exclusively from hierarchical information structures, OOA introduces a number of new concepts [COA90], [BAI89].

Although the object-oriented (OO) development promotes exploratory programming and rapid prototyping, and exploits reuse via inheritance, one can still question why the stake holders of legacy applications hesitate to re-engineer those applications using new OO paradigms. The reason is the analysis phase and the “Object” that is identified in the problem domain during the OOA [LOY89]. It is in the identification of objects and their classes, their attributes (instance variables), their structural organization, and their categorization into subjects, that most of the difficulties in object-oriented analysis arise. This, when coupled with the significant shift in systems thinking, i.e., from process to object-oriented, is the reason why most analysts prefer structured techniques to object-oriented techniques.

A Possible Solution to the OOA Problem: The Synthesist Approach

A panel of members was asked to consider the following question: What is the relationship between Structured Analysis (SA) and OOA? [OOP90].

More specifically, the questions asked were:

- Can SA be used effectively to produce the requirements for a system that will be designed and implemented in an OO fashion?
- If not, is it possible to adjust SA, what needs to be added? If SA cannot be used at all, what is the key obstacle?
- In case SA and OOA have different applicability ranges, how do we circumscribe – positively and negatively - these ranges? Any overlap?

The response of Paul Ward [OOP90] (quoted on next page) provides a possible answer to the questions posed.

...With the addition of some OO modeling guidelines, SA can be used effectively to express a specification in terms of classes of objects with inheritance properties. Such a specification can then be translated in a straightforward way to an OO design. The model partitioning guidelines of SA originally confined to functional process partitioning and adhoc data partitioning, have evolved to include as alternatives event-response process partitioning and object-based data partitioning. If responses are decomposed into components that access single data objects, an event-response model can be partitioned in terms of interacting objects. If the Entity-Relationship model identifies superclass objects, corresponding superclass process (operations) can be embedded in subclasses in the dataflow model.

What Paul Ward describes is that a possible solution to the OOA problem lies in a synthesisist approach, in which the products of structured analysis are used to identify classes of objects and their specifications. In fact, this is an approach that has been adopted by many researchers [BOO86], [SEI87], [GOR90], [RUM91], [KURO94], and [CHE94]. Paul Ward [WAR89] also makes it clear, that structured analysis can be integrated with OO analysis and design.

In all the proposed methodologies there are two very important issues that are highlighted:

- Integration of SA and OOA is possible, and it is important to be able to derive OOA specifications from SA specifications.
- The process is heuristically driven and requires considerable expertise on the part of the analyst.

However, none of the methodologies are comprehensive or state the heuristics clearly.

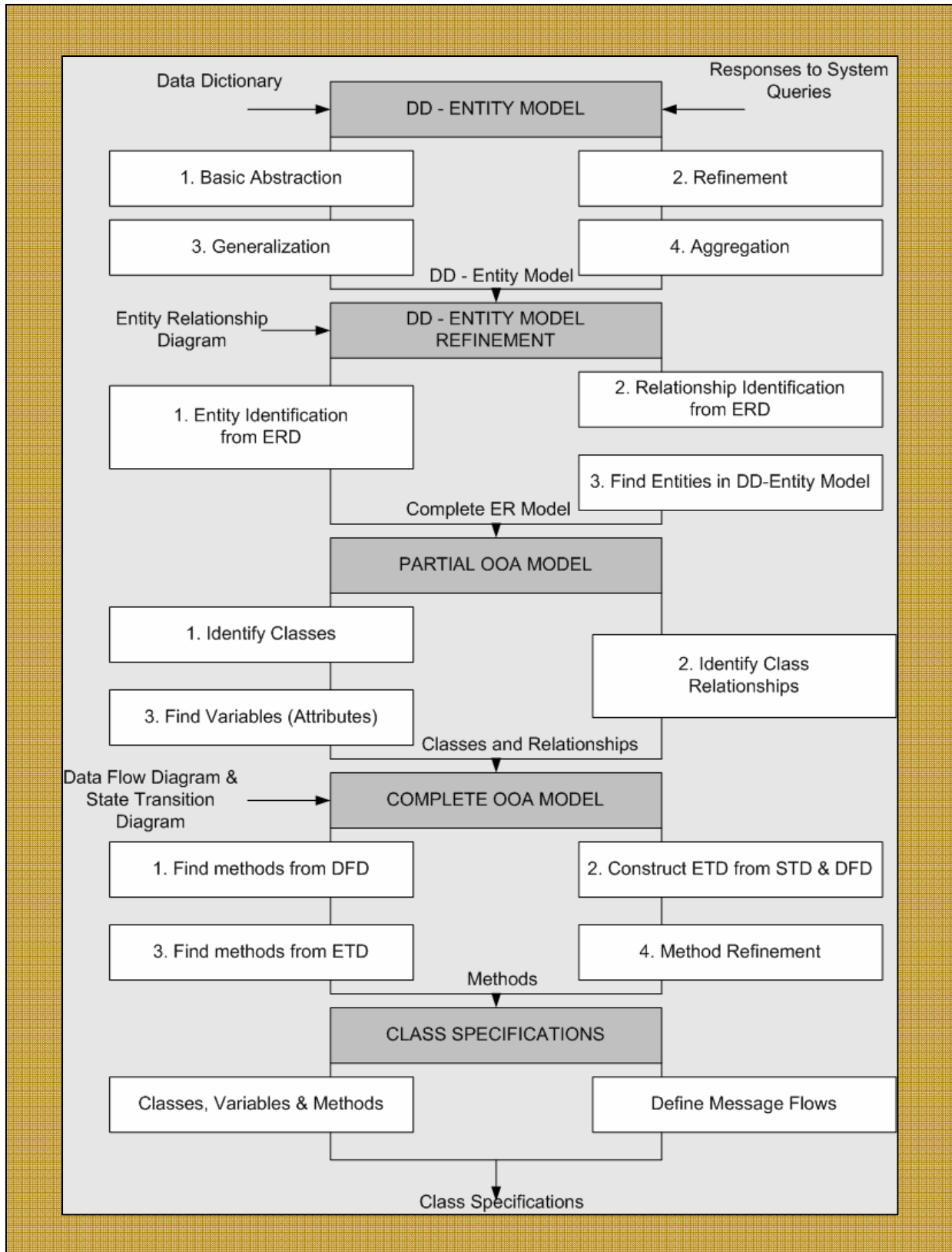
The objective of this paper is to define a methodology for deriving OOA specifications from SA specifications (possibly through an amalgamation of existing methodologies) and give a gist of heuristics.

The Methodology

The proposed methodology, shown in Figure 1: The methodology can be decomposed into 5 stages. These are:

1. Derivation of Data Dictionary-Entity (DD-Entity) model.
2. Refinement of DD-Entity model.
3. Derivation of Partial Object-Oriented Analysis (OOA) model.
4. Derivation of Complete OOA model.
5. Generation of Class Specifications.

Figure 1



In steps (1) and (2), the Entity-Relationship (ER) model is constructed and refined using the data-dictionary and the ER model, which are defined in the SA requirements specifications. During steps (3) and (4), the OOA model is established and step (5) advances the class specifications.

Step-1: This step begins with the collection of data from requirement specifications. The Data Dictionary (with resolved synonyms, homonyms, and instances) together with the expert's experience (obtained through queries posed to the expert) is used to derive a DD-Entity model. This process has 4 sub-steps: basic abstractions, refinement, generalizations, and aggregation development.

By identifying basic abstractions, it can be ensured that the derived ER model is generic in nature and does not contain references to instances of a particular entity set. Further refinements make the DD-Entity model more and more foreign to the user as the model becomes more and more precise for ER model derivations. Lastly, the generalizations and aggregation relationships are identified.

Step-2: The "Entity" is the fundamental unit in modeling the real world. From an object-oriented viewpoint, the data and operation are encapsulated into an entity whereas the traditional entity represents only the data. This represents the difference between a traditional entity and an entity in OOA.

In this step, the entities from an Entity Relationship Diagram (ERD) are listed and are used to crosscheck with the list of entities identified in DD-Entity model. This helps achieve completeness. Next, the relationship defines how an entity is linked with another entity. This relationship will map to some messages and/or class relationships connecting one class to another. The relationship also suggests services an object is expected to provide. By the end of this step, a comprehensive list of entities in the problem domain is formed.

Step-3: The ER model developed from previous steps is used to derive a partial OOA model in which classes, class relationships, class and instance variables are defined.

An object encapsulates the data (attributes' values) and a description of an object's manipulation, i.e., methods that operate on the data [BOO94]. It must reflect the system's responsibilities, real world entities and user's requirements. On the other hand, a class is a set of objects with common attributes and operations. The Entities found in ER model become initial classes and represent most of the system classes. These initial classes may be refined during the identification of class relationships, attributes and operations. The correctness of the classes identified depends entirely on the correctness of the requirement specifications, from which the derivation takes place. However, it may still be difficult, if not impossible, to judge that identified classes are the right ones. What can be done is to refine them and then let the expert decide whether they are appropriate for the system or not?

Classes constitute the static element of a system, while class relationships represent the dynamic behavior of the related classes. As Booch [BOO94] puts it, “the relationship between any two classes encompasses the assumptions that classes make about each other, including what operations can be performed and what behavior results”. Most OO methodologies, however, emphasize how to identify classes, how to find instance variables and methods only. Class relationships are, more or less, neglected. In this case, it is not clear how an object of a class behaves in a system, how it communicates with external objects, and what data are sent by it in the communication. More importantly, it is uncertain if these classes with their instance variables and methods can completely provide the needed services to the users. The relationships identified in this methodology are as follows:

- Generalization-Specialization relationship
- Use relationship
- Aggregation relationship

An instance variable should own the following properties [SEI89]:

- It must describe the characteristics of data that meet the demand of a system.
- It may define some data that makes unique each object instantiated from a class.
- It can represent the common property of objects within same class.
- It describes the value kept within a class, to be exclusively manipulated by the operations of that class, i.e., as class variables.

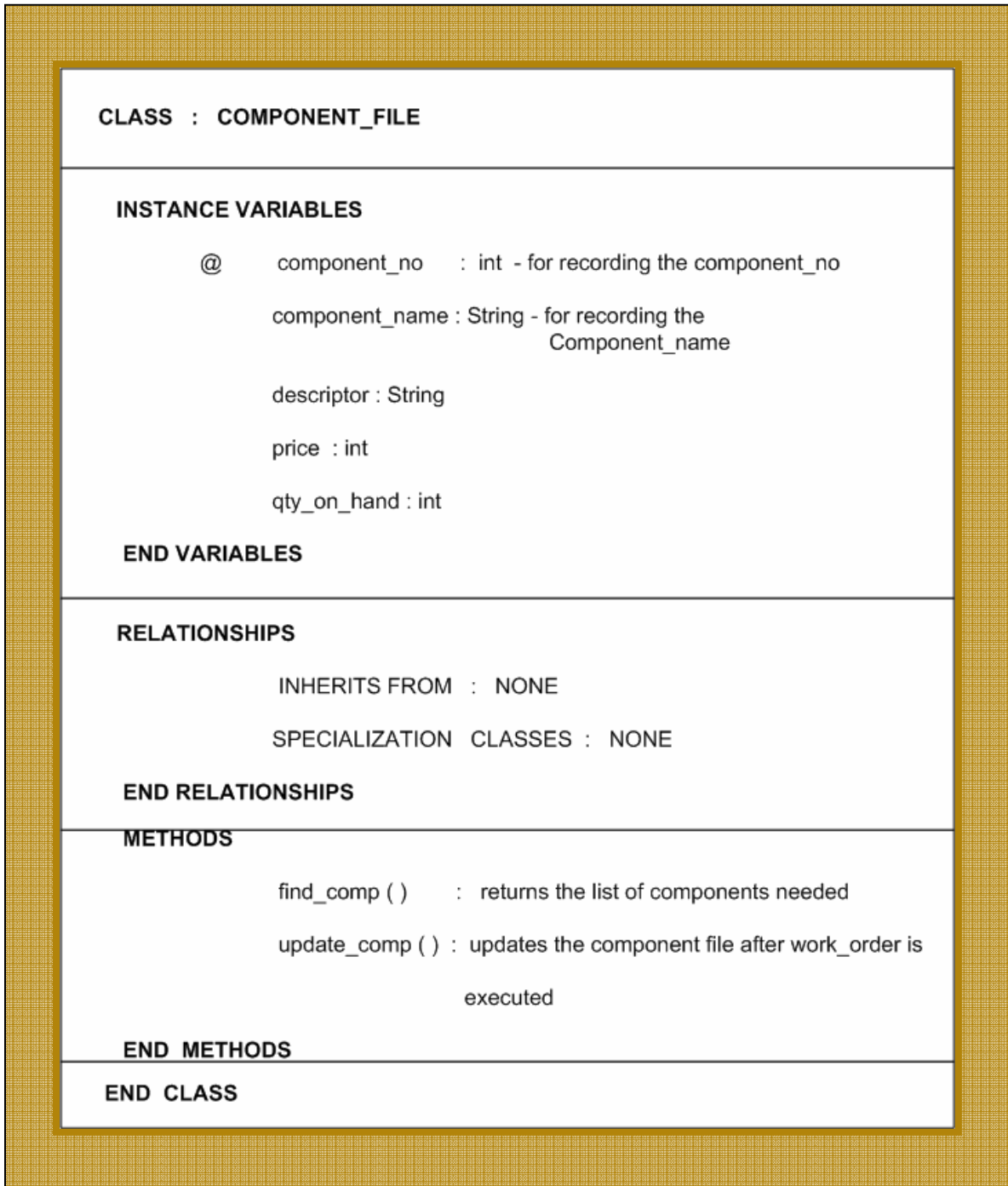
Ordinary instance variables indicate information a class must process or keep. Because they are ordinary, these variables presumably will help understanding the semantics of a class. If variables cannot be allocated to appropriate existing classes, some classes will have to be added. This provides a useful check for class identification. Furthermore, as all input data must be manipulated by methods and then transformed to output data, identification of instance variables can also help to find methods.

Step-4: After the identification of the classes, the Data Flow Diagrams (DFD) and State Transition Diagrams (STD), from the requirement specification, are used to find methods. They are then refined further. Methods of a class provide the capability of accessing/ updating attributes to the class and to communicate through messages with the object instances of the classes. Finding appropriate methods for each class clarifies the system’s behavior and further details the abstraction of the reality being modeled [COA90].

Methods are identified from DFD and Event Trace Diagram (ETD) constructed with help of STD and DFD. An event is an action by an object, which occurs whenever some information is exchanged between two objects in a system. The parameters of the event are the information exchanged. Generally, these parameters are instance variables of the class. This process of constructing ETD requires considerable expertise. ETD can also be used to identify interacting methods. Once these methods are identified, the process of refinement reduces the number of methods of a class without losing any important class characteristics; however, the increase in complexity by grouping some operations into one larger operation should be considered. Thus, there is a trade-off between grouping and not grouping operations.

Step-5: The last step in this methodology is generating class specifications. This is a simple process and involves a mere compilation of the information generated in the previous steps. A typical class specification is shown in the Figure 2: Sample Class Specification.

Figure 2



Conclusion

In this article, a comprehensive methodology, for deriving class specifications from structured specifications has been proposed. The use of this proposal overcomes most of the problems and difficulties in OOA. From structured analysis, which is relatively easy to execute, given the expertise now available, class specifications can be derived. Author has also implemented a prototype Expert System with the help of heuristics derived in each step in the methodology.

References	
[BAI89]	Bailin, S.C., An Object-Oriented Requirements Specification Method, Commns. ACM, Vol. 32, No.5, May 1989, pp.608-623.
[BOO86]	Booch, G., Object-Oriented Development, IEEE Trans. Software Engg., Vol. SE-12, No.2, Feb.1986, pp. 211-221.
[BOO94]	Booch, G., Object-Oriented Analysis and Design, Benjamin-Cummings, 1994.
[CHE94]	Chen, J., and Yu-Shiang H., An Integrated Object-Oriented Analysis and Design method emphasizing Entity/Class Relationship and Operation Finding, J. of Systems Software, Vol.24, No.1, Jan 1994, pp.31-47.
[COA90]	Coad, P., and Yourdon, E., Object-Oriented Analysis, Prentice Hall, 1990
[GOR90]	Gorman, K., Choobineh, J., An Overview of the Object-Oriented Entity-Relationship Model (OOERM), IEEE, 1990, 336-345.
[GUN96]	Gunti, Naveen, Towards an Expert System to integrate Structured Analysis Specifications with Class Specifications in Object-Oriented Analysis, Dissertation report, 1996.
[KUO94]	Feng-Yang Kuo, A methodology for deriving an Entity Relationship Model based on a Data Flow Diagram, J. of Systems Software, Vol. 24, No.1, Jan 1994, pp.139-154.
[LOY89]	Loy, H.P., A Comparison of Object-Oriented and Structured Software Development Methods, Pacific Northwest Software Quality Conference, 1989, pp. 290-303.
[OOP90]	Panel Discussion: Structured Analysis and Object Oriented Analysis, ECOOP/OOPSLA Proceedings, Oct 1990, pp. 135 - 139.
[RUM91]	Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorenson, W., Object-Oriented Modeling and Design, Prentice Hall, Englewood Cliffs, NJ, 1993.
[SEI87]	Seidwitz, E. and Stark, M., Towards a General Object-Oriented Software Development Methodology, Ada Letters, 7, Jul/Aug 1987, pp. 54 - 67.
[SEI89]	Seidwitz, E. and Stark, M., General Object-Oriented Software Development: Background and Experience, J. Systems and Software, Vol 9, 1989, pp. 95 - 108.
[WAR89]	Ward, P.T., How to Integrate Object-Oriented with Structured Analysis and Design, IEEE software, March 1989, pp. 74 - 82.

About the Author



Naveen Gunti, an IT Consultant for last 13 years using Object Oriented technologies last 10 years, has received his Bachelors in Engineering in Electronics and Telecommunications in 1993, Masters in Systems Engineering with Honors in 1995 and Post-Graduate Diploma in Business and Industrial Management in 1996. His interests include developing Artificial Intelligence tools for Software Engineering, Expert Systems and Neural Networks, Distributed Computing and Messaging Systems. He has provided services to fortune 500 companies in Financial, Pharmaceutical and Telecommunication industries in Asia, Europe and North America.

Please share your insights and thoughts with Naveen at naveen.gunti@avenuea-razorfish.com

About Avenue A | Razorfish

Avenue A | Razorfish (avenuea-razorfish.com) is the largest interactive marketing and technology services firm in the U.S., and an operating unit of Seattle-based aQuantive, Inc. (NASDAQ: AQNT). Avenue A | Razorfish solutions are entrenched in deep technology, rigorous analytics and a rich understanding of customer needs, including award-winning web media & creative, search marketing services, email marketing/eCRM, and world-class creative, design and implementation of customer websites and intranets/extranets. Avenue A | Razorfish operates three regions in the U.S. – East, West and Central – with offices located in major markets. In addition, the firm's first international presence was established through the acquisition of U.K.-based full-service interactive agency DNA. Clients include AstraZeneca, Best Buy, Ford Motor Company, Kraft, Microsoft, and Nike. aQuantive, Inc. and all of its operating units are committed to Internet privacy.

Avenue A | Razorfish
821 2nd Avenue, Suite 1800
Seattle, WA 98104
Phone: 206.816.8800
Fax: 206.816.8808

For more information please visit: avenuea-razorfish.com.