Discovering Objects: Which Identification and Refinement Strategies Do Analysts Really Use?

Christopher G. Jones
Utah Valley State College

Thomas S. E. Hilton and Charles M. Lutz
Utah State University

In the near future industry experts predict the object paradigm will be adopted as the dominant approach to systems analysis, design, and implementation. The promise of the object-oriented approach to systems analysis hinges on correctly partitioning the problem domain into essential classes and objects. Most developers agree that this is no easy task. Limited research has been done to date regarding class and object identification and refinement. This paper presents the findings of an empirical study that (a) assesses the frequency of use of various strategies for identifying and refining classes and objects during systems analysis and (b) tests the hypothesized relationships between strategy usage and several respondent characteristics. Results suggest analysts should learn a base strategic repertoire complemented by selected strategies targeted to the application type and the analyst’s previous background.

Software industry pundits predict that soon the object paradigm will emerge as the dominant approach to analyzing, designing, and constructing complex information systems (Orfali, Harkey, & Edwards, 1996). Instead of crafting applications from the traditional building blocks of data and procedures, the new unit of system construction will become the “object”—an integrated package of data and procedures (Cox & Novobilski, 1991; Taylor, 1992).

The object-oriented approach to systems development hinges on correctly partitioning the problem domain into the essential classes and objects (Booch, 1994; Graham, 1994). Methodologists and educators alike recognize that such partitioning is not easy (Booch, 1994; Jean & Strohmeier, 1990). Notable design experts, Gamma, Helm, Johnson, and Vlissides (1995), assert: “The hard part about object-oriented design is decomposing a system into objects” (p. 11). Leading object-oriented methodologist, Grady Booch (1994), contends that “identification of classes and objects is the fundamental issue in object-oriented design” (p. 167).

Firesmith (1993) posits a direct link between class and object identification and software quality:

One of the most important and difficult steps in any object-oriented software development method is that of identifying objects and classes of objects. It is vitally important because the requirements specification, design, and code will consist of interacting objects and classes, and any mistakes in proper identification will have a significant impact on the quality of the resulting software. (p. 169)

Although several strategies exist for finding objects (Firesmith, 1993), including abstraction heuristics (Coad & Yourdon, 1991; Shlaer & Mellor, 1988), behavior analysis (Rubin & Goldberg, 1992), domain analysis (Booch, 1994), extended structured analysis and data modeling (Seidewitz & Stark, 1987; Ward, 1989), and textual analysis (Abbott, 1983), object identification is very much an art with each strategy
producing different lists of objects from a single domain (Budd, 1991). Moreover, not all class and object identification strategies appear to be equally effective in helping analysts identify essential objects (Firesmith, 1993). For today’s analyst, identifying classes and objects is more art than science (Ramackers & Laffra, 1992).

**Problem Statement**

Identification of essential problem domain classes and objects is seen as a major bottleneck to widespread deployment of object technology (Graham, 1994). Although several strategies for finding objects have been reported (Booch, 1994; Firesmith, 1993; Graham, 1994; Potts & Richter, 1993), to date little research has been done regarding (a) how often these strategies are actually used or (b) whether there is a relationship between usage and various developer characteristics.

**Purpose of the Study**

The purpose of this study was to gather information from attendees of a major conference on object-oriented development and research in order to address the following questions derived from the literature:

1. **Initial Identification Approaches**—How frequently are various identification approaches being used to discover the initial classes and objects during object-oriented systems analysis?

2. **Refinement Approaches**—How frequently are various identification approaches being used to refine the initial set of classes and objects into the final set of classes and objects of the system under development?

3. **Hypothesized Relationships**—Is there a statistically significant relationship between the reported frequency of use of an identification or refinement approach and key developer characteristics?

4. **Usage Model**—Is there a combination of developer characteristics that explains a statistically significant portion of the variance in reported use of a particular identification or refinement approach?

Descriptive analysis was used to address research questions one and two. Inferential analysis of six sets of null hypotheses was used to address questions three and four.

**Review of Related Literature**

**Object Identification Strategies**

A review of the literature revealed references to several strategies for identifying and refining the potential classes and objects in an application domain. Table 1 presents synopses of those strategies used in the early stages of object-oriented systems analysis to identify the initial classes and objects that form the vocabulary of the problem domain. Table 2 provides synopses of the various strategies used to refine and refactor the initial set of classes and objects into the final set of classes and objects used to model the system under development. A complete discussion of the various strategies is found in

<table>
<thead>
<tr>
<th>Identification approach</th>
<th>Synopsis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstraction heuristics</td>
<td>Use rules of thumb such as, “Objects are often derived from tangible things, concepts, and events.”</td>
</tr>
<tr>
<td>Behavior analysis</td>
<td>Identify critical system responsibilities or behaviors. Assign responsibilities to system participants that exhibit the desired behavior.</td>
</tr>
<tr>
<td>Browsing of previous work and/or class libraries</td>
<td>Review previous object-oriented analysis and design efforts and/or scan available class libraries and application frameworks.</td>
</tr>
<tr>
<td>Domain analysis</td>
<td>Identify classes and objects that are common to all applications within a given domain by conferring with a domain expert.</td>
</tr>
<tr>
<td>Object-oriented joint application development (OO JAD)</td>
<td>Using a team of domain experts, users, and systems analysts supported by rapid application development tools, brainstorm the key classes and objects that make up the problem domain.</td>
</tr>
<tr>
<td>Reverse engineering</td>
<td>Identify classes and objects by analyzing the forms, reports, and screen displays used in the existing system.</td>
</tr>
<tr>
<td>System state analysis</td>
<td>Identify classes and objects by analyzing the life-cycle and corresponding state transitions of each object in the system.</td>
</tr>
<tr>
<td>Textual analysis</td>
<td>Identify nouns and noun phrases in the narrative description of the proposed system. Common nouns represent classes; proper nouns and direct references represent objects.</td>
</tr>
<tr>
<td>Transforming data modeling/Structured analysis</td>
<td>For Entity-Relationship Diagrams (ER), transform entities into classes and objects. For Data Flow Diagrams (DFD), group external entities, data stores, control stores, and control transformations into objects. Transform data flows and control flows into classes.</td>
</tr>
</tbody>
</table>

Table 1: Synopses of Approaches to Initial Identification
Related Content

Context Management of ERP Processes in Virtual Communities
www.igi-global.com/chapter/context-management-erp-processes-virtual/18558?camid=4v1a

Meronymic Relationships
www.igi-global.com/article/meronymic-relationships/51092?camid=4v1a

Towards a Normal Form and a Query Language for Extended Relations Defined by Regular Expressions
www.igi-global.com/article/towards-a-normal-form-and-a-query-language-for-extended-relations-defined-by-regular-expressions/165161?camid=4v1a

A Meta-Analysis Comparing Relational and Semantic Models
Keng Siau, Fiona F.H. Nah and Qing Cao (2011). *Journal of Database Management* (pp. 57-72).
www.igi-global.com/article/meta-analysis-comparing-relational-semantic/61341?camid=4v1a