Object-Oriented Requirements Elicitation

Donald G. Firesmith

The developer of the Firesmith Method and the OOSDL specification and design language, Donald G. Firesmith has recently joined Knowledge Systems Corporation as a senior member of the technical staff and can be reached at 4001 Weston Parkway, Cary, NC 27513; 919.481.4000; dfiresmith@ksccary.com, or 73664.3513@compuserve.com.

DURING THE LAST seven years, many books and articles have been published documenting object-oriented requirements analysis (OORA) methods and techniques. And almost all of these methods assume the prior existence of well-documented formal customer requirements to be separately analyzed during OORA. Unfortunately, this assumption is often false and it has serious potential negative consequences even when it is true.

Such customer requirements, if they exist, are almost always incomplete, inconsistent, vague, and ambiguous. The current separation of requirements generation and requirements analysis into disjoint customer and developer activities is not only ineffective, but it is also inconsistent with the incremental and iterative nature of the object-oriented (O-O) development paradigm. Incremental development is preferable because many customers do not accurately know what they want until presented with functioning prototypes to verify and validate their desires, needs, and requirements. For historical reasons, customer requirements are typically organized by functional decomposition, event partitioning, or information engineering approaches that separate data and processing requirements. This organization is inappropriate for OORA and makes the tracing of customer requirements to the resulting objects and classes very difficult. Because the customer had to partially analyze the requirements before documenting them for the developer, requirements analysis essentially must be performed twice, which is very inefficient, especially when the results of this first analysis are inappropriate for the second object-oriented analysis.

The following analogy may clarify the problem. Think of Humpty Dumpty as an object containing a yoke of properties surrounded by the whites of operations and protected by a shell of messages. Once these older approaches to analysis have shoved Humpty (together with dozens of other eggs) off the wall and have scrambled their requirements, all of the project's analysts and designers will find it difficult indeed to put Humpty Dumpty back together again. On large projects, Humpty's requirements are mixed with those of many other eggs destined to make an immense omelet (i.e., the application), and the kitchen often contains many cooks, each working on different parts of the omelet. Once objects and classes have been scrambled and distributed in this manner, it is little wonder that developers constantly reinvent different versions of the same wheel and object technology's potential for reuse is rarely realized.*

Finally, what should be done when the customer and the developer belong to the same organization or when formal customer requirements do not exist? With few exceptions,[2] little guidance has been available for those developers who have recognized these problems and who have sought appropriate methods for eliciting high-quality, O-O requirements from the customer for further analysis.

HOW TO ELICIT O-O REQUIREMENTS

Figure 1 illustrates the O-O requirements elicitation and analysis process recommended by the Firesmith Method.† This process combines both requirements elicitation and analysis to maximize productivity and quality by being compatible with the incremental, iterative, and recursive object-oriented development cycle. First, management must staff and train the requirements elicitation and analysis team consisting of both developers and customer representatives. Management and the team must then identify and document the scope of the project in order to determine the objectives and limits of the application to be developed. The team must then identify and obtain sources of information to be used during the requirements elicitation and analysis process. The fourth step can be divided into four parallel subactivities that typically inter-

* Even O-O methods are not necessarily immune to this problem. Objectory[1] is based on use cases (i.e., classes of usage scenarios). Because the same use case involves multiple objects and classes and because the same object or class may take part in multiple use cases, decomposition by use case on large projects and the allocation of different use cases to different development teams may result in the scattering of the requirements for the same object or class to different teams, which may then develop different partial variants of the same object or class.

† Although part of a specific O-O development method, this approach may be used with any O-O development method.

act and often require iterating back to one or both of the previous activities. If appropriate domain analysis has already been performed, then the results of this domain analysis should be reviewed for relevant requirements for the current project. If previous applications in the same domain have been completed, organizational reuse repositories should be reviewed for relevant requirements and frameworks that may be reused on the current project. Any existing customer documents should also be reviewed for relevant requirements. Most importantly, individual interviews and joint application development (JAD) workshops with the customer, user representatives, and domain experts should be scheduled and performed. The team, possibly working with additional customer representatives, then develops O-O requirements models during OORA using the results of the previous step. This step should be iteratively and simultaneously performed with the preceding step and may also be combined with object-oriented design, coding, and testing activities. Finally, the team and the customer representatives evaluate the results of OORA. The preceding requirements elicitation steps can be repeated using iteration to correct any errors found and recursion to elicit and analyze additional requirements missed the previous time(s) through.

Staff and Train the Team

The first step of the proposed requirements elicitation and analysis process is for management to staff the requirements elicitation and analysis team: a small group of customers, users, and developers with the mandate and resources required to identify and analyze the project requirements. This team should be small enough to be efficient and avoid producing "requirements by committee." This team should also be large enough to include expertise in the application domain and object technology including object-oriented analysis. Due to the current state of the industry, team members often require training in one or both of these areas.

Scope the Project

Based on initial guidance from management and marketing, the second step is for the requirements elicitation and analysis team to identify and document the scope of the project. This includes determining the overall objectives of the project such as to develop a totally new system or to correct, enhance, or replace an existing system. This step should identify and document the high-level desires, needs, requirements, and constraints of the customer. The initial product of this step is then used to limit all following activities and especially to limit the scope of the following third step. The requirements elicitation and analysis team need not strive for completeness when they first perform this step because the iterative and recursive nature of the O-O development cycle will ensure that they may revisit this step as many times as is necessary.

Identify Sources of Requirements

The requirements elicitation and analysis team must identify and obtain access to all sources of requirements for elicitation and analysis. These requirements may be obtained from the following four sources: the results of domain analysis, the contents of reuse repositories, documentation, and experts. Documentation and experts are typically the primary sources of information because all too often (1) O-O domain analysis has not been used and (2) the reuse repositories are either incomplete or missing.

Typical customer documents include contractual documents such as requests for proposals (RFPs), statements of work (SOWs), operational concept documents (OCDs), initial system and software requirements specifications, and user manuals for the current system. Application domain and system training materials are often useful. Technical books, encyclopedias, and college texts may also be useful for learning about the application domain of the project.

One of the most valuable and underutilized sources of information is the human expert. The requirements elicitation and analysis team should identify application domain experts, users, and customers who may provide, explain, describe, clarify, justify, and validate requirements for analysis. Because the requirements elicitation and analysis team must eventually reach a consensus on the approved requirements, lead users and customers should also be identified to resolve disputes when consensus proves difficult or impossible to achieve.

Review Domain Analysis

If appropriate domain analysis has previously been performed, then its results should be reviewed for relevant requirements for the current project. If an O-O domain analysis has been performed, then relevant essential objects and classes will probably have been identified and analyzed. In addition, essential

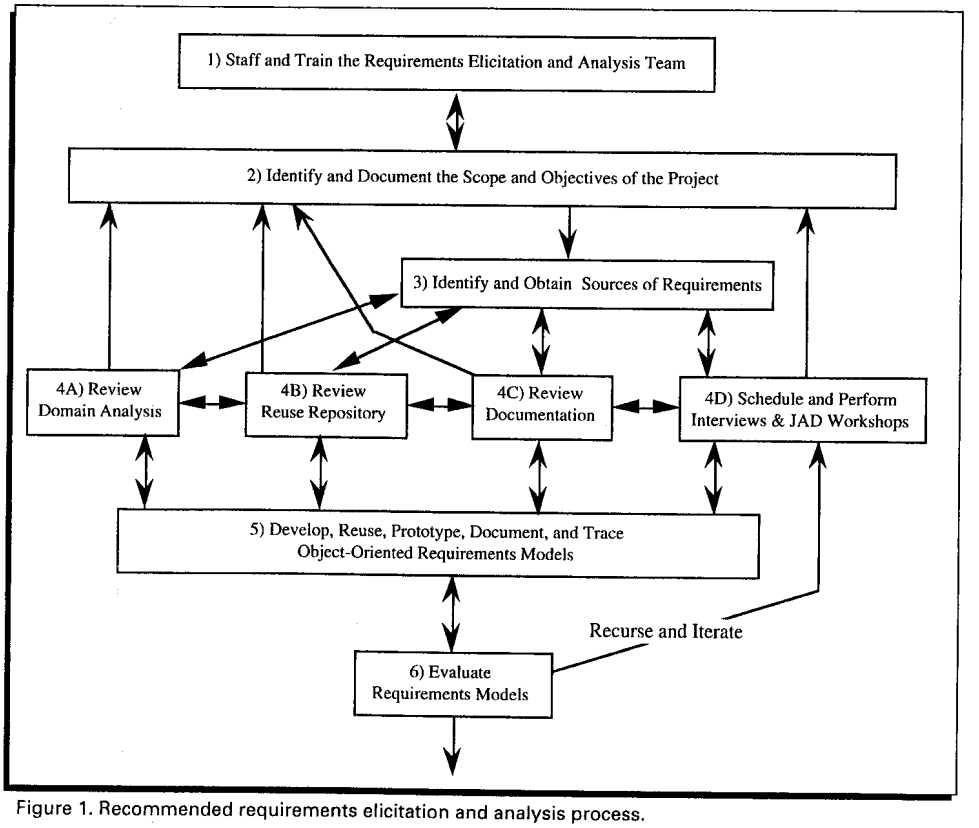

Figure 1. Recommended requirements elicitation and analysis process.

patterns, mechanisms, and subassemblies of objects and classes will hopefully have been identified and analyzed. The previously identified scope and objectives of the project as well as the review of project documentation, interviews, and workshops may then be used to determine which of the requirements identified by domain analysis are relevant to the current project. In this way, the results of domain analysis may be reused as part of the requirements elicitation and analysis of the current project.

Review Reuse Repositories

If previous applications have already been developed, then organizational reuse repositories should be reviewed for relevant requirements for the current project. Such reuse repositories may contain properly documented essential objects and classes as well as essential patterns, mechanisms, and subassemblies of objects and classes. The previously identified scope and objectives of the project as well as the review of project documentation, interviews, and workshops may then be used to determine which of the requirements identified in the organizational reuse repositories are relevant to the current project. These repositories may even contain application frameworks that supply the majority of the essential objects and classes needed by the current project. In this way, organizational reuse repositories may also be reused as part of the requirements elicitation and analysis of the current project.

Review Documentation

Customer documents have traditionally been considered a major source of requirements for the current project. They should be reviewed to identify these requirements and also in preparation for the interviews and workshops. As stated before, typical customer documents include contractual documents such as RFPs, SOWs, OCDs, initial system and software requirements specifications, and user manuals for the current system. Unfortunately, such documents still tend to be incomplete, inconsistent, vague, ambiguous, and organized in a manner (e.g., by functional decomposition and the separation of data and processing requirements) that is inappropriate for OORA. Tracing of requirements from these documents to the resulting objects and classes is difficult. Because they often consist of little more than narrative English text, little modeling and analysis of the raw requirements has occurred. For these reasons, such documents should be considered as mere sources of raw requirements rather than the official repository of approved requirements.

Schedule and Perform Interviews and JAD Workshops

After an initial review of the documentation, the requirements elicitation and analysis team schedules and holds both (1) interviews with individual application and domain experts and (2) JAD workshops with small numbers of application and domain experts. These interviews and workshops should further refine and prioritize the scope and objectives of the project. During these interviews and workshops, questions are asked resulting in the identification and documentation of essential objects and classes. Directed by members of the requirements elicitation and analysis team, various O-O models and their notations are jointly developed and evaluated. This interactive building of models during interviews and workshops produces models that are easier for customers and users to understand as well as models that are more likely to meet the needs and desires of the customers and users. Object-oriented diagrams (e.g., semantic nets, interaction diagrams, inheritance and aggregation diagrams, state transition diagrams, timing diagrams) are incrementally developed. A very effective way to develop such diagrams is to use large (e.g., 5" by 7") cards representing objects, classes, and other O-O entities (e.g., users, hardware terminators). Paper can also be printed with two or three appropriate icons, reproduced on a standard office copier, and cut into individual cards.‡ Labeled with the proper identifier, each card is taped onto *large* whiteboards and arcs representing appropriate relationships (e.g., collaboration, inheritance, aggregation) are drawn and labeled. Having the nodes on paper makes it easy to move them around the whiteboard as the diagrams evolve. Because there is little investment involved in creating diagrams this way, people feel free to make improvements and iterate until the diagrams stabilize and quality requirements and designs result. The interviews and workshops should also be used to elicit scenarios of how the system should behave and to determine the essential interactions among the objects and classes. These scenarios should cover both normal and exceptional behavior.§ The workshops are also good venues for role playing and the building of consensus.

A major problem with many requirements specifications is that they do not adequately specify how the system is to behave when it cannot behave as normally specified. One should remember that as much as 80% of safety-critical and business-critical systems is exception-handling code

Interviews should involve asking questions designed to elicit O-O requirements. Early on, the emphasis should be on objects rather than functions or even data. Objects and classes can be identified by asking "What *things* are important?" and "What *things* exist in the application domain?" Attributes can be elicited by asking "What are their required *properties*?" Both objects and attributes can be identified by listening for nouns. Operations can be elicited by asking "How do these things *behave*?" and by listening for verbs. Associations and links can be elicited by asking "How are these things *related* to one another?" Inheritance can be elicited by asking "What things are 'a kind of' other thing?" Aggregation can be elicited by asking "What things *contain* other things or are *parts* of

‡ I originally recommended Post-It™ notes, but they often did not stick well to whiteboards, and after lunch one could find part of one's model lying on the floor.

§ A major problem with many requirements specifications is that they do not adequately specify how the system is to behave when it cannot behave as normally specified. One should remember that as much as 80% of safety-critical and business-critical systems is exception-handling code. If the requirements do not adequately specify such software, individual developers will often guess the intent, sometimes with disastrous results. Assertions and exceptions are therefore as much as a part of an object as its attributes and operations.

other things?" Scenarios can be used to tie these objects and classes together to obtain required high-level behaviors. As more is learned, interviews and workshops can be repeated iteratively and recursively to identify new requirements and verify existing ones.

Develop, Reuse, Prototype, Document, and Trace Object-Oriented Requirements Models

Different O-O development methods create different O-O models documented using different notations according to different development procedures. These models are best developed by experts in object technology working together with experts in the application domain during JAD workshops. The traditional approach of having developers develop models in isolation using only customer documentation as input is very inefficient for several reasons. It minimizes communication between customers and developers and suffers due to the incompleteness and inappropriateness of most customer documentation. The requirements elicitation and analysis team should iteratively and recursively identify requirements and develop the requirements models. Where practical, they should reuse requirements models created by domain analysis and stored in organizational reuse repositories. They should trace the identified and analyzed requirements back to their sources: domain analysis, reuse repository, document, interviewed person, and JAD workshop. They should document these identified and analyzed requirements in the appropriate sections of an object-oriented requirements specification or an object-oriented requirements specification and design document[ii] whose contents and format are consistent with the specific O-O development method used.

Evaluate Requirements Models

The requirements elicitation and analysis team, together with the customer, should evaluate the documented requirements models for correctness, completeness, consistency, etc. Scenarios should be evaluated and role playing should occur to ensure that the required architecture and behavior is captured by the requirements. They should prioritize these requirements and attempt to allocate them to the different builds and releases of the application. Finally, a consensus must be reached as to which requirements are approved and which are relegated to the status of mere desired feature.

If the requirements are being elicited and analyzed in very small increments as part of a recursive "analyze a little, design a little, code a little, test a little" development process, then each increment of requirements may be informally evaluated as part of a peer-level inspection by the team. Under such circumstances, customer involvement in the reviews may be postponed until the next formal in-process review (IPR) that evaluates all increments (e.g., subassemblies of objects and classes) developed since the previous IPR.

CONCLUSION

The topic of O-O requirements elicitation has been largely ignored as most analysis methods have assumed the prior existence of well-documented requirements to be analyzed. Unfortunately, this assumption is often incorrect, and any existing documentation is often a poor and inadequate source of O-O requirements. The separation of customer requirements identification from developer requirements analysis is also inconsistent with the iterative and recursive nature of the incremental, parallel O-O development cycle. In this column I have proposed an O-O requirements elicitation process for incorporation into existing O-O development methods. I have recommended specific activities that are to be performed both iteratively and recursively as part of an O-O development cycle. The proposed process can be of immediate benefit, and it is hoped that this article will stimulate discussion and further research into O-O requirements elicitation.

References

1. Jacobson, I., M. Christerson, P. Jonsson, and G. Oevergaard. Object-Oriented Software Engineering: A Use Case Driven Approach, Addison-Wesley, Reading, MA, 1992.
2. Graham, I. Object-oriented requirements capture, ObjectExpo Europe Conference Tutorial Proceedings, July 1993.

[ii] Because an incremental, iterative "analyze a little, design a little" development process should be used, it is typically preferable to localize all of the requirements and design into a single object-oriented requirements specification and design document instead of in two separate documents as was common when using the classic waterfall development cycle. This also greatly simplifies requirements traceability and maintenance, resulting in decreased project schedule and cost.

continued from page 13

References

1. Henderson-Sellers, B. and J.M. Edwards. Book Two of Object-Oriented Knowledge: The Working Object, Prentice Hall, Sydney, 1994.
2. Monarchi, D.E. and G.I. Puhr. A research typology for object-oriented analysis and design, Communications of the ACM 35(9):35–47, 1992.
3. Wand, Y. and R. Weber. An ontological evaluation of systems analysis and design methods, Information Systems Concepts: An In-Depth Analysis, Falkenberg, E. and P. Lindgreen, Eds., Elsevier Science Publications/North-Holland, Amsterdam, 1989.
4. de Champeaux, D. and P. Faure. A comparative study of object-oriented analysis methods, Journal of Object-Oriented Programming 5(1):21–33, 1992.
5. Coad, P. and E. Yourdon. Object-Oriented Design, Prentice Hall, Englewood Cliffs, NJ, 1991.
6. Wirfs-Brock, R.J., B. Wilkerson, and L. Wiener. Designing Object-Oriented Software, Prentice Hall, Englewood Cliffs, NJ, 1990.
7. Rubin, K.S. and A. Goldberg. Object behavioral analysis, Communications of the ACM 35(9):48–62, 1992.
8. Meyer, B. Applying "design by contract", IEEE Computer 25(10):40–52, 1992.
9. Jacobson, I., et al. Object-Oriented Software Engineering: A Use Case Driven Approach, Addison-Wesley, Reading, MA, 1992.
10. Drake, J., et al. Object-oriented analysis: Criteria and case study, International Journal of Software Engineering and Knowledge Engineering 3(3):319–350, 1993.
11. Firesmith, D.G. Object-Oriented Software Requirements Analysis and Logical Design: A Software Engineering Approach, Wiley, New York, 1993.
12. Unhelkar, B. and G. Mamdapur. Practical aspects of using a methodology: A road map approach, Report on Object Analysis and Design 2(2), 1995.
13. Walden, K. and J.M. Nerson. Seamless Object-Oriented Architecture, Prentice Hall, Englewood Cliffs, NJ, 1994.
14. McGregor, J.D. and T.D. Korson. Integrated object-oriented testing and development processes, Communications of the ACM 37(9):59–77, 1994.
15. Henderson-Sellers, B. Towards a process metamodel architecture: I. Basic rationale, Report on Object Analysis and Design 2(2), 1995.