Active Design Reviews: Principles and Practices

ARTICLE *in* JOURNAL OF SYSTEMS AND SOFTWARE · DECEMBER 1987

Impact Factor: 1.35 · DOI: 10.1016/0164-1212(87)90025-2 · Source: DBLP

2 AUTHORS:

David Parnas
Middle Road Software
297 PUBLICATIONS 14,912 CITATIONS
SEE PROFILE

David Weiss
Iowa State University
77 PUBLICATIONS 3,639 CITATIONS
SEE PROFILE

Active Design Reviews: Principles and Practices

David L. Parnas* and David M. Weiss

Naval Research Laboratory
Washington, D.C.

* Now also Lansdowne Professor of Computer Science at University of Victoria, Victoria, B.C.

ABSTRACT

Although many new software design techniques have emerged in the past 15 years, there have been few changes to the procedures for reviewing the designs produced using these techniques. This paper describes an improved technique, based on the following ideas, for reviewing designs.

(1) The efforts of each reviewer should be focussed on those aspects of the design that suit his experience and expertise.

(2) The characteristics of the reviewers needed should be explicitly specified before reviewers are selected.

(3) Reviewers should be asked to make positive assertions about the design rather than simply allowed to point out defects.

(4) The designers pose questions to the reviewers, rather than vice versa. These questions are posed on a set of questionnaires that requires careful study of some aspect of the design.

(5) Interaction between designers and reviewers occurs in small meetings involving 2 - 4 people rather than meetings of large groups.

Illustrations of these ideas drawn from the application of active design reviews to the Naval Research Laboratory's Software Cost Reduction Project are included.

KEYWORDS

SOFTWARE ENGINEERING DESIGN REVIEWS
INFORMATION HIDING SOFTWARE DESIGN PROCESS

1. Introduction

Although many new software design techniques have emerged from research in software engineering, there have been few changes to the procedures for reviewing the design before writing the code. This paper presents a new approach to reviewing a software design. We illustrate our ideas in terms of designs based on the information hiding principle, but the ideas can be applied to other designs as well. The principal innovation of our approach is the use of questionnaires to give the reviewers better defined responsibilities and to make them play a more active role. The technique has been used in the Naval Research Laboratory's Software Cost Reduction (SCR) project for several years with good results. The SCR project involves the experimental redevelopment of the operational flight program (OFP) for the Navy's A-7E aircraft. The examples used in this paper are taken from that project.

2. Objectives Of Design Reviews

The purpose of all design reviews is to find errors in the design and its representation. The review should be designed to make it easy for the reviewers to find errors. If errors are present but escape the reviewers' attention, the review has failed. In theory, the errors found should be errors made in producing the documents that are being reviewed. In fact, errors are often found that were made earlier and were not caught. Such errors include unstated requirements, unnecessary requirements, obsolete requirements, and design and implementation decisions stated as requirements. Reviews should be designed to find both the errors made in specifying the latest design decisions and the errors made earlier.

Getting maximum benefit from a review requires that the objectives of the review be made explicit. To achieve the objectives of the review, a systematic review method is needed. This method must ensure that the design is covered completely and in detail by the review. The method should take maximum advantage of the skills and knowledge of the available reviewers.

Error Classification

To focus reviewers' attention properly, and to take advantage of their different skills and knowledge, we find it useful to classify design errors as follows.

(1) Inconsistencies, i.e., places where the design won't work. For example, if two design statements make different assumptions about the "sense" of an angle, the design simply will not work.

(2) Inefficiencies, i.e., places where the design imposes a barrier to efficient programming or use. For example, a programmer who is only interested in obtaining the latitude of a location from another programmer's module, but whose program is only able to request both latitude and longitude at the same time, is being forced to waste both time and space.

(3) Ambiguities, i.e., places where the design specification may be interpreted in several different ways, or is not clear enough.

(4) Inflexibilities, i.e. places where the design does not accommodate change well. For example, if the design assumes that there are exactly two sources of altitude information, it may be difficult to deal with an additional altimeter.[1]

The purpose of this categorization is not to ensure that each error found falls into one and only one category, but rather to guide us in designing reviews that will find as many errors as possible.

Obtaining Detailed Coverage Of The Design

When a set of reviewers that can provide complete design coverage is assembled, it is important to ensure that each reviewer focuses on areas relevant to him and that he can identify the decisions made in arriving at the design. It is also important to make the reviewer think hard about what he is reading rather than skim it for obvious errors.

To accomplish a detailed review, we believe it is necessary to force the reviewers to take an active role whether they agree or disagree with the design decisions that have been made. The reviewers must be asked to provide justification for accepting or rejecting design decisions.

[1]Note that some inflexibilities may actually be requirements misunderstandings, particularly if the requirements specify changes that are likely to occur during the lifetime of the system. (See [1] for an example of a requirements specification containing likely changes.)

Because using a document is often the best way to find its problems, we try to make the reviewers use the document that they are reviewing. We do this by asking them to answer a set of questions about the document - questions that can only be answered by careful study of the document. Some questions may ask them to write programs that would use the programs specified by the design.

Performing a good job as a reviewer is difficult work. It is important that the review be designed to give the reviewers a sense of participation and accomplishment. Lacking such a feeling they are unlikely to participate in future reviews.

3. Conventional Design Reviews

Conventional design reviews usually proceed as follows.

(1) A massive quantity of highly detailed design documentation is delivered to the reviewers three to four weeks before the review.

(2) The designated reviewers, many of them administrators who are not trained in software development, read as much of the documentation as is possible in the time allowed. Often this is very little.

(3) During the review, a tutorial presentation of the design is given by the design team; during this presentation, the reviewers ask any questions they feel may help them to understand the design better.

(4) After the design tutorial, a round-table discussion of the design is held. The result is a list of suggestions for the designers.

The design documentation for such reviews is usually an English prose description of the design in a prescribed format. Pseudo-code representations of many of the algorithms to be implemented are sometimes included.

Those attending the design tutorial include managers who are paying for the development, potential users, programmers who maintain similar systems, and those who will build or maintain the system under development.

Problems With Conventional Reviews

Conventional design reviews tend to be incomplete and shallow. They fail to uncover many of the errors and weaknesses in the design. The following circumstances contribute substantially to this failure.

(1) The reviewers are swamped with information, much of which is not necessary to understand the design. Structural decisions are hidden in a mass of implementation details. Finding the relevant information is difficult and tedious. Even given sufficient time, reviewers are unlikely to extract the information they need. They rarely have the time.

(2) Most reviewers are not familiar with all of the goals of the design and the constraints placed on it.

(3) Responsibility for reviewing the design may rest with the review team as a whole, without individual reviewers having clear individual responsibilities. All reviewers may try to look at all of the documentation, with no part of the design receiving a concentrated examination.

(4) The design team expects the reviewers to initiate action to uncover errors and weaknesses. Reviewers who have a vague idea of their responsibilities or the design goals, or who feel intimidated by the review process, can avoid potential embarrassment by saying nothing.

(5) Interaction between individual reviewers and the design team is limited because the review is conducted as a large meeting. Detailed discussions of specific design issues become hard to pursue in this situation.

(6) Often the wrong people are present. People who are mainly interested in learning the status of the project, or who are interested in learning about the purpose of the system may turn the review into a tutorial.

(7) Reviewers are often asked to examine issues beyond their competence. They may be specialists in one aspect of the system, but they are asked to review the entire system.

(8) There is no systematic review procedure and no prepared set of questions to be asked about the design. In the rush to prepare and read the mass of documentation, little thought is given to the conduct of the review.

(9) As a result of unstated assumptions, subtle design errors may be implicit in the design documentation and go unnoticed.

There are many variations on conventional reviews, each of which changes some aspect of the approach in the hope of improving the process. Some of the currently popular variations are well-described in [2] and [3]. Despite the popularity of these approaches, we believe that none of them solves the major problems of conventional reviews.

4. A More Effective Review Process

To avoid the problems just stated and to achieve the objectives stated in section 2, we redesigned the design review process. Rather than conducting one large meeting, we identify several types of reviews, each designed to find different types of design errors. Since each review requires its own expertise, we identify different types of reviewers needed to perform the reviews. Instead of conducting the reviews as discussions, we devise questionnaires for the reviewers to answer using the design documentation. Responding to the questionnaire requires the reviewer to study the documentation carefully, and to make assertions about design decisions. The issues raised by the reviewers as a result of answering the questionnaires are discussed in small meetings between designers and each reviewer. There are no protracted group discussions involving all designers and all reviewers. Because our reviews are very tightly focused, we also ask a few people for overall reviews to decrease the chance that we overlook problems because we neglected to pose a question.

The remainder of this paper contains a more detailed description of the design properties that we believe a good review should cover, the properties of a reviewable design representation, the structure needed in the design documentation to facilitate the review process, the identification of the individual review types, the classification of reviewers, the design of the questionnaires, and the conduct of the reviews.

4.1. Making The Design Reviewable

Producing a reviewable design requires concern with both the design and the design documentation. We start with a discussion of design properties, and then discuss documentation issues.

4.1.1. Design Properties

Regardless of the object being designed, it should be possible to identify the desirable properties of a design. Our list of such properties follows.

(1) Well structured. The design should be consistent with chosen principles, such as the principle of information hiding.

(2) Simple. To paraphrase a statement often attributed to A. Einstein, the design should be as simple as possible, but no simpler.

(3) Efficient. The functions provided by the design should be computable with the available computing resources and the design should not interfere with meeting space and time requirements.

(4) Adequate. The design should satisfy present requirements.

(5) Flexible. The design should make it as easy as possible to respond to requirements changes.

(6) Practical. Module interfaces should be sufficient for the job, neither providing unneeded capability nor requiring the user to perform functions that properly belong to the module being used.

(7) Implementable. The functions offered by the design should be theoretically computable with the information available.

(8) Standardized. The design should be represented using a standard organization for design documentation.

4.1.2. Making The Design Representation Reviewable

The design representation should make the design assumptions as explicit as possible. Including explicitly stated assumptions is a form of redundancy that makes the design easier to review, but also means that additional consistency checks must be performed.

Making Assumptions Explicit

The interface between two modules is the set of assumptions that the programmers writing one module make about the behavior of the other module. Such assumptions are embodied in the description of the module's access programs. Using the terminology and notation of [4], a module that offers to its users a function that returns a device's current mode of operation embodies the assumption that the software can detect the mode of that device. This assumption is implicit in the statement that such a program is included in the interface; it may go unnoticed and unquestioned by a reviewer. To make the assumption explicit, it must be included in the design documentation; to help the reviewer, it should be included in a section devoted to basic assumptions about the module.

Other types of assumptions that may be implicit are assumptions about operations that can be performed by the programs of the module, types of data provided to and returned by those programs, the effects of the programs on each other, the time at which information is available (compile time, load time, run time, when the module is in a particular state, etc.), the variability of data, and undesired events [5].

For example, the mode determination submodule of the OFP contains an access program, named IN_MODE, that takes as input the name of a mode of the system and returns a Boolean variable whose value indicates whether or not the system is currently in that mode. (Modes are classes of system states.) Included as basic assumptions about the module are the following.

(1) The system is always in at least one mode. Changes in current modes can always be signaled to user programs.

(2) This module can always determine if the system is in a particular mode.

(3) Mode transitions may be considered to be instantaneous.

These assumptions are implicit in the specification of IN_MODE, but are explicitly stated in the design documentation for the mode determination submodule.

We do not advocate that every assumption that users can make about the module be explicitly specified. Assumptions that apply to all designs need not be included. An example is the assumption that it is possible to pass information to other modules using parameters or the assumption that a parameter designated as "Input", will not be assigned a new value.

Including Redundant Information In The Design Representation

A basic principle used in engineering is that redundant information is needed to perform error checking. The lists of assumptions included with the design documentation for each module are a form of redundancy; they represent the designer's intentions. They can be used to check that the designer stayed true to his intentions, and they can be read by non-programmers to check the adequacy and validity of his intentions.

Two lists of assumptions are included for each module. One is a list of basic assumptions about the module, i.e., information that designers of the module assume will never change. The second is a list of assumptions that describe incorrect usage of the module, i.e., usage that the designers assume should not occur.

The list of basic assumptions is redundant with the description of access programs and their externally-visible effects. The list of incorrect-usage assumptions is redundant with the description of undesired events. For an example of an abstract interface representation see [6].

Omitted access programs, parameters, undesired events, and program effects are examples of missing information that we can detect by including redundancies. Contradictory assumptions and descriptions of program effects are examples of inconsistencies similarly detectable. Detection is accomplished by comparing the lists of assumptions, which appear in one section of the interface document, with the rest of the document. In the previous example, a reader of the interface can compare the definition of IN_MODE and its effects with the basic assumptions about the module. Were the third assumption omitted, the reviewer should question the use and meaning of invoking IN_MODE when the system is in transition from one mode to another.

4.1.3. Organizing Design Documentation For Review

Most of the SCR design documentation consists of abstract interface specifications for the modules that constitute the design. Each such specification is the subject of a separate review. For example, the module hierarchy for the A-7E OFP contains a device interface module (DIM) whose secret is those characteristics of the peripheral devices connected to the computer that are likely to change. This module comprises 23 submodules, each of which was specified in one section of [6]. During the review of the DIM, each of the submodule interfaces was reviewed separately.

Because there is often information common to all the submodules of a higher level module, the abstract interfaces for the submodules are combined in a single design document, with the common information included only once. A set of cross-references and indices are supplied with the document. These aids permit the reviewers (and users) of these documents to find quickly modules, access programs, locally-defined data types, terms, undesired events, and symbols.

We also provide a set of instructions for reviewing the document. Discussion of these instructions follows.

4.2. Identifying Review Types

To give individual reviewers a focus and a clear area of responsibility, a set of specialized reviews should be designed. These reviews are categorized according to the properties of the design that the reviewer should be verifying. Each review must have a specified purpose and be intended for use by a reviewer with a particular expertise. The following briefly characterizes some of the reviews used for the Device Interface Modules.

(1) Review A: Assumption Validity. For each device, check that all assumptions made are valid for any device that can reasonably be expected to replace the current device.

(2) Review B: Assumption Sufficiency. For each device, check that the assumption lists contain all the assumptions needed by the user programs in order to make effective and efficient use of the device.

(3) Review C: Consistency Between Assumptions And Functions. For each module, compare the assumptions to the function and event descriptions to detect whether (a) they are consistent, and (b) the assumptions contain enough information to ensure that the functions can be implemented, the events can be detected, and the module can be used as intended.

(4) Review D: Access Function Adequacy. For each device, check that user programs can use the device efficiently and meet all requirements using only the access functions provided in the abstract interface.

Figure 1 shows some of the questions comprising review C.

4.3. Classifying Reviewers

Just as several reviews each with different purpose are needed, several reviewers each with different expertise and perspective are needed. Among them are the following.

(1) Specialists, such as a person with detailed knowledge of a particular class of device or of a particular application, such as avionics. These specialists should be capable of assessing the performance and feasibility aspects of the design.

(2) Potential users of the system.

(3) Those who are familiar with the design methodology used, even if they are not familiar with the application.

(4) Those who are skilled at and enjoy finding logical inconsistencies and who may be used for performing systematic consistency checks, despite not being specialists in a particular area.

The DIM reviews required 4 kinds of reviewers, as described in figure 2.

To assure complete coverage of the design document, reviewers are assigned so that each kind of review is performed for each section of the document. In scheduling our reviews we try not to require any one reviewer to perform one kind of review for all sections of the document. We try to keep the time expended by an individual reviewer from being more than what is usually available for a design review. It is rare that one reviewer is able or asked to perform more than one kind of review.

4.4. Designing The Questionnaires

The instructions shown in figure 1 describe the properties for which the reviewer should check, the sections of the abstract interface to be studied, and include a questionnaire to be completed by the reviewer.

The questionnaires are designed to make the reviewers take an active stand on issues and to use the documentation. An example is question C1, which requires the reviewer to use the document to find assumptions that justify the inclusion of each access program in the interface. The question is phrased in an active way ("Which assumptions tell you ..."), rather than "Is there an assumption that justifies the implementation of this function?" The passive phrasing would give the designer little assurance that the reviewer read, understood, and thought about the document. Using the passive phrasing makes it too easy to say "yes."

4.5. Conducting The Review

The review schedule must include time for each reviewer to complete his questionnaire(s) and must also allow a chance for discussions between each reviewer and the designers, both during the time he is completing the questionnaire(s) and after he has completed them.
The discussions are either one-on-one sessions between a reviewer and designer, or include a small group of each. We found no need for protracted group discussions involving all designers and all reviewers.

Our reviews are conducted in three stages. In the first stage, a brief overview of the module being reviewed is presented. The overview consists of explaining the modular structure, if it is unfamiliar to the reviewers, showing where in the structure the module belongs, and describing the module's secrets. If previous assignments have not been made, reviewers are assigned to reviews and sections of the document. Reviewers are also assigned times during which they may raise any questions they have, and a time to meet with designers after the designers have read the completed questionnaires.

In the third stage the designers read the completed questionnaires and meet with the reviewers to resolve questions the designers may have about the reviewers answers to the questionnaires. The reviewers may be asked to defend their answers. This interaction continues until both designers and reviewers understand the issues, although agreement on those issues may not be reached. After the review the designers produce a new version of the documentation. A discussion of design issues, including those raised during the review, are contained in our documents.

5. Conclusions

Our results with active design reviews have convinced us that the technique is an effective way of achieving our review objectives. In addition to helping us find design errors, the technique is significantly effective in assuring that we use our design methodology consistently.

We believe our success with active reviews is a result of the following factors.

(1) The technique helps us to select appropriate reviewers by requiring that we write down the characteristics of the reviewers needed.

(2) The technique makes good use of the reviewers' skills by focusing their energies on those aspects of the design that they are best suited to evaluate.

(3) The technique makes effective use of our reviewers' time by focusing their attention on the issues that they know about. The structure of the questionnaires allows the reviewers to concentrate on one concern at a time. Since there is no large continuing group review, each reviewer concentrates on his part of the review independently and in parallel with all other reviewers.

(4) Each reviewer must make a considerable commitment of effort to the review. No reviewer merely observes the process without contributing. Reviewers who might hesitate to speak up in a large discussion group will often identify problems in a one-on-one meeting. All reviewers feel that they are contributing to the review process and that they are gaining an understanding of the design.

(5) The conduct of the review fosters rapid focus on specific problems. Reviewers and designers both gain understanding of problems in the design. Designers are matched with reviewers with similar interests and backgrounds and are able to interact freely without distraction.

(6) The structure of our documentation permits us to design the review questionnaires to focus attention on different concerns individually, and to provide the information needed to accomplish a complete, detailed review.

By focusing our reviewers' attentions and making them take an active role, we find many more errors than we would find in a conventional review.

In the second stage reviewers perform their reviews, meeting with designers to resolve questions that they have about the design or about the review questionnaires.

6. Acknowledgements

We thank the many people who have acted as reviewers during the development of the active design review technique, in particular those people at the Naval Weapons Center, China Lake and at the Naval Research Laboratory. Sandra Fryer and Robert Westbrook were particularly helpful in finding people and time to support the reviews carried out at NWC.

Bruce Labaw and Paul Clements contributed many valuable suggestions for improving an earlier draft of this paper. Several improvements were also suggested by Carl Landwehr and by Ken DeJong.

7. References

1. K. Heninger, J. Kallander, D. L. Parnas, and J. Shore, "Software Requirements for the A-7E Aircraft," Memo Report 3876, Naval Research Laboratory (November 1978).
2. D. Freedman and G. Weinberg, *Ethnotechnical Review Handbook, Second Edition,* ETHNOTECH, INC. (1979).
3. M. Fagan, "Design and Code Inspection and Process Control in the Development of Programs," TR 21.572, IBM System Development Division (December 1974).
4. P. C. Clements, R. A. Parker, D. L. Parnas, J. E. Shore, and K. H. Britton, "A Standard Organization for Specifying Abstract Interfaces," NRL Report 8815, Naval Research Laboratory (June 1984).
5. D. L. Parnas and H. Wuerges, "Response To Undesired Events In Software Systems," *Proc. Second International Conference On Software Engineering* (1976).
6. R. A. Parker, K. L. Heninger, D. L. Parnas, and J. E. Shore, "Abstract Interface Specifications for the A-7E Device Interface Module," Memo Report 4385, Naval Research Laboratory (November 1980).

Review C: Consistency Between Assumptions And Functions

The assumptions should be compared to the function and event descriptions to detect whether (a) they are consistent, and (b) the assumptions contain enough information to ensure that the functions can be implemented and the events can be detected. If an access function cannot be implemented unless the device has properties that are not in the assumption list, there is a design error, i.e., either a gap in the assumption list or a function that cannot be implemented for some replacement device. The device interface specifications should be reviewed for this criterion by AVIONICS PROGRAMMER reviewers. After studying the assumptions, the design issues, and the functions and events, they should perform the following reviews:

Review C-1

For each of the access functions the reviewer should answer the following questions:

C1 Which assumptions tell you that this function can be implemented as described?

C2 Under what conditions may this function be applied? Which assumptions describe those conditions?

C3 Is the behavior of this function, i.e., its effects on other functions, described in the assumptions?

Figure 1. Part Of The Reviewer Questionnaire For Review C

Point of View	Expertise Required
DEVICE EXPERT	Familiarity with the device used on the A-7 and with similar devices found on other aircraft. These people ought to know about several devices of this type, about the technology used to build such devices, and about past changes and future trends in these devices. They need not be programmers; people involved with the design or purchase of such devices would be fine.
DEVICE PROGRAMMER	Experience writing or modifying programs that deal with this device or others of the same type. They should be familiar with tricks to use the device effectively with minimum consumption of computing resources. If people with experience on the A-7 cannot be found, people who have used similar devices for other aircraft are acceptable.
AVIONICS PROGRAMMER	Good logical minds and familiarity with avionics programming in general. These people need not have experience writing programs for these specific devices. They will be asked to perform certain checks for *internal* consistency and completeness; consequently a lack of information on the specific devices may be advantageous.
A-7 REQUIREMENTS EXPERT	Sufficient familiarity with the A-7 software requirements document [1] to be able to read the function descriptions that refer to the devices hidden in this module. Familiarity with the A-7 application beyond this document is not essential. We value the ability to make disciplined logical completeness checks above familiarity with the A-7. However, they should have experience writing programs similar to the A-7 software.

Figure 2. Characteristics of Reviewers for the DIM Abstract Interfaces

REVIEW	REVIEWER
A: Assumption Validity	Device Expert
B: Assumption Sufficiency	Device Programmer
C: Consistency Between Assumptions And Functions	Avionics Programmer
D: Access Function Adequacy	Device Programmer

Figure 3. Correspondence Of Reviews And Reviewers