A Software Inspection Exercise for the Systems Analysis and Design Course

Craig K. Tyran
Department of Decision Sciences
Western Washington University
Bellingham, WA 98225-9077
craig.tyran@wwu.edu

ABSTRACT

Software inspections have been found to be one of the most effective ways to promote quality and productivity in software development. Inspections are an especially important tactic to use during the analysis and design phases of software development since the correction of a defect found early in development can be 10 to 100 times less expensive to fix than rework performed at the system testing stage. Given its prominence within the software field, it is surprising that the software inspection process does not receive more attention with respect to education in the area of Systems Analysis and Design. The purpose of this article is to present an experiential exercise for the Systems Analysis and Design course that may be used to promote learning with respect to the software inspection process. The focal point of the exercise is a system specification document that describes the user requirements for a system for a fictional real estate company. The specification document includes three components that are typical of a specification document: a descriptive narrative overview, a project dictionary, and data flow diagrams (DFDs). Survey results regarding students' perceptions of the exercise are also discussed.

Keywords: software inspection, software quality, systems analysis, education, data flow diagram, system specification

1. INTRODUCTION

Software inspections have been found to be one of the most effective ways to promote quality and productivity in software development (Gilb and Graham, 1993; Laitenberger and DeBaud, 2000). The software inspection is a peer review in which a small group of software developers examine another software developer's work. The primary purpose of a software inspection is to identify defects existing within software work products developed throughout the development process (e.g., user requirements specifications, design documents, code). Data collected during the inspection process is used not only to correct defects, but also to evaluate and improve the development process itself. Reports from industry indicate that inspections have gained wide acceptance as a development tactic and can take up to 15 percent of the time allotted to a software project (Ackerman, Buchwald, and Lewski, 1989). Based on the demonstrated value of software inspections, more than one industry expert has listed the software inspection process at the top of the list of desirable software development practices (Boehm, 1987; Glass, 1999).

While inspections have been found to be worthwhile for all phases of the development process, it is an especially important tactic to use during the analysis and design phases of software development since the correction of a defect found early in development can be 10 to 100 times less expensive to fix than rework performed at the system testing stage (Boehm and Basili, 2001; Doolan, 1992; Fagan, 1986). Given its prominence within the software field, it is somewhat surprising that the software inspection process does not receive more attention with respect to education in the area of Systems Analysis and Design. The purpose of this article is to present an experiential exercise for the Systems Analysis and Design course that may be used to promote learning with respect to the software inspection process. The next section provides a brief overview of the stages and guidelines for the software inspection approach. Second, the experiential exercise and the associated exercise materials are discussed. Third, survey results regarding students' perceptions of the exercise are summarized. The article concludes with summary comments.

2. SOFTWARE INSPECTION PROCESS: STAGES AND GUIDELINES

Software inspections were initially introduced and formalized by an employee of International Business Machines (IBM) named Michael Fagan in the early 1970s (Fagan, 1976). As described by Fagan, the software inspection process is a formal process encompassing six stages (Fagan, 1986). While some aspects of the inspection process have evolved over the years, Fagan's stages continue to serve as the basis for software inspections. Fagan's six stages are:

- Planning: Determine that the materials that will be inspected will be suitable. Also, arrange the participation of the appropriate people and a meeting place and time.
- Overview: Educate the inspection meeting participants about the piece of work that will be inspected. Assign roles to the participants (e.g., scribe, moderator).
- Preparation: Participants do their "homework" and work individually to get familiar with the piece of work.
- Inspection: The sole purpose of the inspection stage of the process is to find defects. Developing alternate solutions or redesigning the materials under inspection is strongly discouraged.
- Rework: The author resolves all of the defects documented.
- Follow-up: The team moderator or the entire inspection team checks on the author's rework to make sure that all of the corrections are effective and that no new defects have been introduced.

In addition to defining the stages for the inspection process, Fagan (and other practitioners) have suggested several guidelines for inspection teams. Examples of specific guidelines include (e.g., Ebenau and Strauss, 1994; Fagan, 1976; Yourdon, 1989):

- The number of participants on inspection teams should be manageable (3-6 people).
- Emphasize error detection, not correction.
- Consider the work product guilty until proven innocent.
- The producer of the work product is always innocent (i.e., focus on the product and not the person who developed the product).
- Do not use inspections for purposes of performance appraisal.
- Follow organizational standards (e.g., diagramming/naming conventions, etc.) to reduce misunderstandings or disagreements.
- Recognize that there may be some "open issues" that can not be resolved at the inspection meeting (e.g., when inspecting an analysis documents, there may be some questions that need to be referred to the user).
- Keep length of inspection to be less than two hours to reduce fatigue factor.
- Conduct inspections frequently to find errors as early as possible.

3. OVERVIEW OF SOFTWARE INSPECTION EXERCISE

The purpose of this section is to describe an experiential software inspection exercise which has been developed to introduce students to the software inspection process. The exercise has been designed for a Systems Analysis and Design course which emphasizes the traditional structured analysis and design concepts (e.g., system development life cycle, data flow diagrams, entity relationship diagrams, etc.). A recent survey of Information Systems faculty members conducted by Mahapatra, Nerur, and Slinkman (2005) indicates that over 75% of the instructors surveyed focus on the structured analysis and design concepts, instead of object-oriented techniques (e.g., UML). Hence, the exercise described in this manuscript is likely to be applicable for many instructors. The focal point of the exercise is a system specification document that describes the user requirements for a system for a fictional real estate company. The specification document includes three components that are typical of a specification document (Yourdon, 1989): a descriptive narrative overview, a project dictionary, and data flow diagrams. Based on the early class room experiences with the inspection exercise, the document was refined and optimized.

As suggested by Teague and Pidgeon (1985), the four major criteria for evaluating a system specification include completeness, consistency, communicability, and correctness. Examples of defects corresponding to each of these criteria have been included into the document to provide a challenging inspection task for students. Defects include incomplete project dictionary entries, unbalanced DFDs, poorly labeled data flows, and incorrect logical design of processes and data flows. Over twenty defects appear in the specification document (a list of defects is available upon request from the author).

The steps of the exercise are based on the standard inspection methodology (Ebenau and Strauss, 1994; Fagin, 1986; IEEE, 1989), but aspects of some steps have been modified somewhat to accommodate the class room setting. Fagan's (1986) "Overview," "Preparation", and "Inspection" stages receive the most emphasis for the experiential exercise. However, the "Planning," Rework," and "Follow-up" stages of the process may also be discussed by the instructor when introducing and recapping the exercise. Similar to a real inspection task, the exercise involves a combination of individual work and small group interaction.

The author has found that it works best to administer the inspection exercise about halfway through the Systems Analysis and Design course, after the students have become familiar with topics and activities of the analysis stage such as systems development, process modeling using data flow diagrams, the project dictionary, and data specifications. The exercise has been designed to be a graded, in-class assignment lasting about one and a half hours. The group inspection portion of the exercise involves inspection meetings of student teams and the exercise is graded based on the final list of defects generated by each team. Following is a discussion of the exercise steps.

3.1 Topic Overview: Software Inspections

Prior to the exercise, students are provided with a short lecture (approximately 20 minutes) regarding the inspection software process. The lecture summarizes the motivations for conducting a software inspection, as well as the traditional inspection methodology. As the inspection exercise is being conducted in the Systems Analysis and Design course, the types of work products (e.g., data flow diagrams, project dictionary, user interface designs) that may be examined during the Analysis and Design stages of the system development process are discussed. Also, the advantages of conducting inspections early in the software development process are emphasized. If an instructor has a two-hour class period, then it would be possible to do this lecture on the day of the inspection exercise. However, experience has found that it typically works best to do the lecture prior to the day of the exercise. The author has observed that students tend to be excited about doing the inspection exercise and prefer to start the exercise early in the class period.

3.2 Initiating the Exercise

At the beginning of the exercise, the students are provided with a handout that includes an "Inspection Report Form," an "Action List" form, and the directions for the exercise. Also, the students are informed who their teammates will be. If students are already working with a team for a course project, then these teams may be used for the exercise. Otherwise, inspection teams are assigned by the instructor. Consistent with the industry recommendations (and based on the author's experience with the inspection exercise), a team size of three to four students is preferred. After students are informed of their team composition, the instructor provides a very quick reminder about the purpose underlying the software inspection process and then begins to review the exercise materials.

The first handout form that is reviewed with the students is the "Inspection Review Form" (available upon request from the author). This form has been adapted from a published source (Ebenau and Strauss, 1994) and helps to provide students with a sense of how inspection meetings may be scheduled and documented in an industry setting. The author has also found that a review of this form can be useful because it helps serve as a vehicle for discussing the types of tasks that need to occur before, during, and after an inspection meeting. For example, a discussion of the "Coordinator's Checklist" section of this form helps to emphasize that inspection meetings require coordination prior to the meeting and provides an opportunity for the instructor to discuss the importance of the "Planning" stage of Fagan's inspection process. Also, a discussion of the "Meeting Checklist" section of the form provides a good way for the instructor to review the agenda for the exercise (i.e., a review of the "ground rules," individual inspection, group inspection meeting).

The second form that is discussed is the "Action List" form (available upon request from the author). A discussion of this document helps the students to realize that the ultimate deliverable for the exercise will be a list of defects (or areas of concern) that will be assembled by their inspection team. The author has found that many students find it useful to get a solid sense of what their final output for the exercise will be before they get immersed with the detailed aspects of the exercise.

3.3 The Overview and Preparation Steps

Next, the instructor reviews the exercise directions with the students (see Appendix A). There are three primary steps to the exercise directions. As discussed in Appendix A, the first step is for each student to prepare for the inspection meeting. At this time, the instructor hands out a document that is a "System Specification" for a fictional real estate company (see Appendix B). The document includes a narrative summary of the system processes, along with three data flow diagrams, and a project dictionary. In this step, the instructor provides a brief review of the specification document. This portion of the exercise corresponds to Fagan's "Overview" stage. Next, students are asked to work on their own to read through the contents of the document that they will be inspecting. While the timing may vary, students are typically provided ten to fifteen minutes of individual time to get familiar with the specification document. During this period, students are not asked to search for defects. Instead, they are simply asked to get familiar with the contents of the document. Once students have gotten familiar with the document, the instructor requests that students to move onto the second step of the exercise, in which students are asked to identify as many defects as they can find. Along with the first step, this is an individual activity that is aimed at simulating the "Preparation" stage of the inspection process described by Fagin and others (Ebenau and Strauss, 1994; Fagin, 1986; IEEE, 1989). Students are provided about thirty minutes for this portion of the exercise and are asked to record defects that they find on a sheet a paper.

3.4 The Inspection Step

After the individual detection session, students meet with their teammates to conduct the inspection meeting. This corresponds to the Fagan's "Inspection" stage of the process. As described in Appendix A, during this stage of the process the team members review their individual findings and create the Action List. The Action List will serve as the final deliverable for the team and will include the list of defects, as well as the associated disposition codes. The teams are encouraged to identify new defects during the inspection meeting. About thirty minutes is allocated to this portion of the exercise. At the end of this step, the teams are asked to finalize their Action List and complete a "Group Decision" section at the bottom of the Inspection Report Form, in which the team recommends whether or not the "System Specification" document should be accepted or revised.

3.5 Recapping the Exercise

After the teams submit their Action List and Inspection Report Form, the instructor discusses how these documents will be used to support the "Rework" and "Follow-Up" stages of the inspection process. Also, the instructor reviews the defects that are present in the specification document and answers any questions that students may have regarding the inspection exercise or the software inspection process.

4. CLASS EXPERIENCE

The software inspection exercise has been used in many sections of the undergraduate Systems Analysis and Design course taught by the author. The students taking the course were typically "traditional" undergraduate students with limited work experience in the area of systems development. Based on the author's observations, the students appeared to enjoy doing the exercise and seemed to learn from the experience. To supplement the author's observations, a survey questionnaire has been administered to the students to gain feedback regarding the exercise. The survey was used to measure the students' perceptions and does not measure changes in actual student learning performance. Hence, the survey has limitations. Nonetheless, the survey does provide a way to assess the students' reaction to the exercise. The survey data spans three years and six sections of the course. The survey was completed by all students who completed the exercise, providing a sample size of N=110. The sample represented 36 inspections teams with team sizes ranging from 2 to 4 students (3 teams of two members, 28 teams of 3 members, and 5 teams of 4 members).

A primary reason for doing the inspection exercise was to help students learn more about the software inspection process. Ideally, as an outcome of doing the exercise, the author wanted the students to build their confidence regarding the software inspection process. It appears that the exercise was effective in this regard. As indicated in Table 1, after completing the exercise, the students reported a significantly increased sense of confidence with respect to the task of organizing and conducting a software inspection. Hopefully, this increased level of confidence may encourage students to conduct software inspections in the future.
In addition to helping students learn more about the software inspection process, the author hoped that students would gain other educational benefits. Several published articles have indicated that the software inspection process may offer educational benefits for the participants such as improved analytical abilities and defect detection abilities (e.g., Bisant and Lyle, 1989; Fagan, 1976; Doolan, 1992). As indicated in Table 1, feedback from the students indicates that they strongly believed that the exercise helped them to gain knowledge that would allow them to prepare higher quality specification documents and identify defects.

In order to determine whether or not the students found the team inspection meeting to be useful, the students were asked to rate their understanding of the defects before and after the inspection meeting. It appears that the inspection meeting aspect of the exercise was worthwhile, as the students indicated a significantly better understanding of the defects within the specification document following the exercise (see Table 1).

Lastly, from an instructional perspective, the implementation success of a new class exercise can often hinge on how favorably the students view the exercise. Based on the author's observations, students seemed to be very motivated to do the inspection exercise and actually seemed to enjoy doing the task. These observations were supported by the survey data shown in Table 1, as the students indicated a high level of satisfaction with the exercise process and generally found the exercise to be a pleasant experience. The positive feedback regarding the exercise was not based on a perception that the exercise was easy, as teams generally identified only about half of the defects existing in the specification document.

Questionnaire Item	Mean	Std	
Pre vs. Post Exercise: Change in Confidence			T-Test
Before I did this exercise I was confident that I could organize and conduct a successful software inspection process.	3.49	1.38	p<.000 (t=15.3)
After doing this exercise I am confident that I could organize and conduct a successful software inspection process.	5.55	1.09	
Perception of Knowledge Gains			
As a result of participating in this exercise, I gained knowledge that will help me:			
To prepare higher quality specification documents in the future (i.e., documents similar to the one used for this exercise).	5.90	0.96	
To be more effective in identifying defects in a system specification document such as the document used for this exercise.	5.86	1.00	
Pre vs. Post Group Meeting: Change in Understanding			T-Test
When I got done with the individual defect identification portion of the exercise (step 2), I had a very clear understanding of the defects existing within the specification document.	5.28	1.07	p<.000 (t=3.96)
Following the group inspection portion of the exercise (step 3), I now have a very clear understanding of the defects existing within the specification document.	5.66	1.03	
Satisfaction with Process and Perception of Exercise	Mean	Std	
I was satisfied with the process that was used in this exercise to identify defects and generate the Action List.	5.92	0.94	
Performing this exercise was a pleasant experience.	5.62	1.17	

Notes:
1) Rating value is based on a 7-point Likert scale ranging from 1 (strongly disagree) to 4 (undecided) to 7 (strongly agree).
2) Number of respondents: N=110 (36 teams).

Table 1: Student Perceptions of Inspection Exercise

5. SUMMARY COMMENTS

Overall, the author has found the software inspection exercise to be a worthwhile component to the Systems Analysis and Design course. Given the importance of software inspections to the practice of systems analysis and design, it seems appropriate to allocate some class time to this important topic. The exercise discussed in this article provides an experiential approach to help students gain a better feel for the software inspection process. By working through the exercise, students often gain enough confidence to attempt the approach on their own (using the steps discussed in the exercise). The author has observed that once students have completed the exercise, they will often apply their new inspection skills to other class activities such as team-based software development projects.

As described above, the hands-on portion of the exercise can be completed within an hour and a half. However, the timing for the exercise can be adjusted by the instructor to be shortened or extended. For example, for classes that meet over a two-hour period, the overview lecture on software inspections can be added at the beginning of the class session to fill the class time. Alternatively, for classes that meet over a one-hour period, the Preparation step could be conducted by students as an individual exercise prior to the class session. The exercise has the flexibility to fit within different types of class room timing constraints.

The exercise described here is suitable for a Systems Analysis and Design course which focuses on the traditional structured approach to analysis and design. If an instructor is teaching the Analysis and Design course using object-oriented methods, then the exercise will need to be redesigned. However, the general inspection steps discussed in this article will still be applicable. For instance, the inspection process is just as appropriate for the evaluation of use cases (e.g., Thelin, Runeson, and Wohlin, 2003) as it would be for the evaluation of DFDs. Regardless of the type of system development methodology used in the classroom, it can be useful to develop a student's awareness and capabilities with regard to one of the most powerful approaches for improving software quality: the software inspection process. It is hoped that this article may inspire other instructors to include more coverage of the software inspection process into the Systems Analysis and Design course.

6. REFERENCES

Ackerman, A. F., Buchwald, L.S., and Lewski, F.H. (1989) "Software Inspections: An Effective Verification Process," IEEE Software, Vol. 6, No. 3, pp. 31-36.

Bisant, D.B. and Lyle, J.R. (1989) "A Two-Person Inspection Method to Improve Programming Productivity," IEEE Transactions on Software Engineering, Vol. 15, No. 10, pp. 1294-1304.

Boehm, B. (1987) "Industrial Software Metrics Top Ten List," IEEE Software, Vol. 4, No. 5, pp. 84-85.

Boehm, B. and Basili, V.R. (2001) "Software Defect Reduction Top 10 List," IEEE Computer, Vol. 34, No. 1, pp. 135-137.

Doolan, E.P. (1992) "Experience with Fagan's inspection method," Software: Practice and Experience, Vol. 22, No. 2, 1992, 173-182.

Gilb, T. and Graham, D. Software Inspection, Addison-Wesley, Wokingham, England, 1993.

Glass, R.L. (1999) "Inspections – Some Interesting Findings," Communications of the ACM, Vol. 42, No. 4, pp. 17-19.

Ebenau, R.G. and Strauss, S.H. (1994) Software Inspection Process. McGraw-Hill, New York, NY.

Fagan, M.E. (1976) "Design and Code Inspections to Reduce Errors in Program Development," IBM Systems Journal, Vol. 15, No. 3, pp. 182-211.

Fagan, M.E. (1986) "Advances in Software Inspections," IEEE Transactions on Software Engineering, Vol. 12, No. 7, pp. 744-751.

IEEE Computer Society. (1989) IEEE Standard for Software Reviews and Audits (IEEE Std. 1028-1988). IEEE, New York, NY.

Laitenberger, O. and DeBaud, J. (2000) "An Encompassing Life Cycle Centric Survey of Software Inspection," Journal of Systems and Software, Vol. 50, pp. 5-31.

Mahapatra, R., Nerur, S.P., and Slinkman, C.W. (2005) "Teaching Systems Analysis and Design – A Case for the Object Oriented Approach," Communications of the AIS, Vol. 16, pp. 848-859.

Thelin, T., Runeson, P., and Wohlin, C. (2003) "Prioritized Use Cases as a Vehicle for Software Inspections," IEEE Software, Vol. 20, No. 4, pp. 30-35.

Yourdon, E. (1989) Structured Walkthroughs, Prentice Hall: Englewood Cliffs, NJ.

AUTHOR BIOGRAPHY

Craig K. Tyran is an Associate Professor in the Department of Decision Sciences at Western Washington University. He received his PhD from the University of Arizona in Management Information Systems, his MBA from UCLA, and his undergraduate and master's degrees in engineering from Stanford University. Dr. Tyran conducts research in the areas of technology support for collaboration and learning. His work has been published in MIS Quarterly, the Journal of Management Information Systems, the Communications of the ACM, and a variety of other publication outlets. In addition, he has presented his research at numerous national and international conferences.

APPENDIX A: DIRECTIONS FOR SOFTWARE INSPECTION EXERCISE

<u>Step 1: Preparation</u>

Objective: If you are on an inspection team, a key task that you will need to address <u>before</u> you start looking for defects is to become familiar with the work product that will be reviewed. The purpose of this step will be to get familiar with the system specification for the "Cascade Real Estate Information System."

The Scenario: In this exercise you will be reviewing a draft of an analysis document that describes a proposed information system for a real estate firm. Assume that the document has been prepared by a newly hired systems analyst named Joseph Analyst. Joseph will ultimately be reviewing his work with the management of real estate firm and wants to have his document inspected by others so that he can improve the quality of his work. The document includes a) a “Narrative Overview” description of the system and its processes, b) a project dictionary, and c) data flow diagrams (DFDs). Please take some time to read through the analysis specification and become familiar with it.

Note: It is important for you to have a basis for knowing what the project dictionary and data flow diagrams are supposed to represent. You may assume that the textual “Narrative Overview” section of the specification provides an accurate description of the system processes – for purposes of this exercise, assume that the “Narrative Overview” section does NOT have defects. Using the “Narrative Overview” as a basis, you will find that the project dictionary and DFDs created by Joseph Analyst DO have defects.

<u>Step 2: Defect identification</u>

Objective: The purpose of this step is to search for defects in the document. This step will be an individual activity. (In Step 3 you will share your findings with your team when you conduct the inspection meeting to generate your team's Action List.)

Guidelines:

- <u>When in doubt use “Narrative Overview” as the "basis"</u>: As noted earlier, you may assume that the written narrative section of the specification provides an accurate description of the system.
- <u>Record your findings:</u> Please record your defect findings on a sheet of paper. You will later be sharing your findings with your teammates. Defects may include specification aspects which are incorrect, missing, or difficult to understand.
- <u>Detection, not correction!:</u> A standard guideline for the inspection process is to focus on the detection of defects – not the correction of the defects. Please do <u>not</u> attempt to figure out how to correct the defects that you find. You only need to <u>detect</u> the defect. The author of the work product you are reviewing will be responsible for making corrections.
- <u>If you are not sure whether you found a defect ….:</u> If you are not sure whether you have identified an actual defect or not, go ahead and record it. You and your team will have the chance to address and discuss each potential defect during the team inspection meeting that follows this step of the exercise.

<u>Step 3: Develop "Action List" with team</u>

Objective: Now you are ready to conduct the team inspection. This step will be a team activity. Your team's goal will be to create an “Action List” that will be provided to Joseph Analyst to identify aspects of his work that require improvement.

Your team: Your team has already been assigned. Please gather together with the other members of your team.

Guidelines:
- <u>Review the individual findings</u>: Please have each team member orally step through each of his/her findings. For each finding, the group will need to decide whether the finding is an item that should be included on the "Action List." Your team may encounter one or more potential defects that seem to fall into a gray area (i.e., "Is this a defect or isn't it?"). To help deal with such situations, your team may include the following three types of items on the Action List:
  - A defect: This is something that the team agrees is wrong and wants to see fixed.
  - A suggestion: This is something that may not be necessarily be a defect, but someone on the team would like to see it modified (e.g., a data flow that is labeled somewhat poorly).
  - An open issue: This is an item that appears to be problematic, but requires more information before it can be classified as a defect.
- <u>Create the Action List</u>: As your team reviews the individual findings, you will create your action list using the Action List form that has been provided to you.
  - Please <u>indicate</u> whether a finding is a "Defect," "Suggestion," or "Open Issue."
- <u>New findings? … Include them!:</u> Many inspection teams experience a "synergy effect" and generate new findings during the team inspection meeting. Please be sure to include any of these "new" findings on your team's Action List.

APPENDIX B: SYSTEM SPECIFICATION DOCUMENT FOR REAL ESTATE LISTING SYSTEM

Project Name: Cascade Real Estate Listings Information System

Author: Joseph Analyst

Scenario Background: Cascade Real Estate is a residential real estate firm located in a small rural city named Cascade Village. Due to the small size of the Cascade Village, Cascade Real Estate is the only real estate firm in town and handles all real estate transactions between sellers and buyers. Currently, Cascade Real Estate uses a manual process to receive, store, and organize information about home listings and buyer preferences. (Note: A "listing" refers to a house which has been put on the housing market.) However, in the interest of modernizing its operations, the management of Cascade Real Estate has decided to fund the development of a computerized system support the firm's operations related to information management and reporting for the management and buyers. A context data flow diagram (DFD) is provided to show the key processes and the external entities to the system (see Figure 1).

Narrative Overview of System

Note: Assume that the following Narrative Overview is accurate and does NOT contain defects.

Overview of Major Sub Processes for System: Based on interviews with the management and employees of Cascade Real Estate, the following is a summary of the three major sub processes for the proposed system. These processes are represented in the Level-0 data flow diagram for the system (see Figure 2).

- Process 1.0 (Receive and Store Listings Information): This sub process receives and stores information submitted by home owners who have contracted with Cascade Real Estate to sell their home through Cascade Real Estate. The information submitted by home owners includes data about their home (see data dictionary for specifics). The information should be stored in the Listings File. This process is a functional primitive.

- Process 2.0 (Receive and Store Buyer Information): This sub process receives and stores information submitted by potential home buyers who would like to purchase a new home in Cascade Village. Information includes data about the buyers and their requirements for a new house (see data dictionary for specifics). The information is stored in the Buyer File. This process is a functional primitive.

- Process 3.0 (Generate Reports): This sub process generates two types of reports: 1) A summarized listing report for the firm's managing partner (i.e., a report which summarizes all of the firm's house listings), and 2) A customized listing report for the potential home buyers. This process is not a functional primitive.

Overview of the "Generate Reports" Process: Two types of reports are generated every week: the management report and the customized reports for potential buyers. Based on interviews with the management and employees of Cascade Real Estate, the following is a summary of the three sub processes associated with the "Generate Reports" process. These sub processes are represented in the Level-1 data flow diagram for the "Generate Reports" sub process (see Figure 3). Each of the following sub processes is a functional primitive.

- The management report is prepared by obtaining the listings records from the Listings File. The Listings File includes all homes which have been listed with the Cascade Real Estate firm. The listings are summarized into a report that is sent to the firm's managing partner.

- A customized listing report for each potential buyer (also referred to as a "client") is prepared by accessing data from the Listings file and the Buyer File. The contents of the Listings File are compared with each potential buyer's stated requirements for a new home (e.g., number of bedrooms and bathrooms, size of lot). Each buyer's requirements are accessed from the Buyer File. Homes listed by Cascade Real Estate which are suitable for each buyer are identified and summarized into a customized listing report.

- A mailing label is prepared for each customized listing report by accessing the appropriate potential buyer's address from the Buyer File. Each customized report is then mailed to the appropriate potential buyer after affixing the mailing label to the report.

APPENDIX B: SYSTEM SPECIFICATION DOCUMENT FOR REAL ESTATE LISTING SYSTEM (CONT.)

<u>Project Dictionary</u>

<u>Notes</u>: 1) This Project Dictionary was prepared by Joseph Analyst and may contain defects.
2) The bracket notation used below ({}) represents a repeating group of data.

- Process Descriptions
  - Process descriptions have not yet been prepared for any of the functional primitives
- External Entities (Sources/Sinks)
  - Seller: Sellers enlist the services of Cascade Real Estate to help sell their homes.
  - Buyer: Buyers enlist the services of Cascade Real Estate to help find a home to purchase.
  - Managing Partner: The top management official of Cascade Real Estate.
- Data Stores
  - Buyer File = {Buyer File Record}
  - Listings File = {Listings File Record}
- Data flows and data structures
  - Buyer Address = Buyer Name + Buyer Address Record
  - Buyer File Record = Buyer Requirements
  - Buyer Name = First Name + Last Name
  - Buyer Requirements = Buyer Name + Buyer Requirements Record
  - Buyer Requirements Record = Number of Bedrooms + Number of Bathrooms
    + Square foot size + House style + Lot size + School district
  - Customized Report = Buyer Name + {Listing Number + Listing Date
    + Listing Sales Price + House Address Record + Number of Bedrooms
    + Number of Bathrooms + Square Foot Size + House Style}
  - Customized Report with Mailing Label = Buyer Name + {Listing Number
    + Listing Date + Listing Sales Price + House Address Record
    + Number of Bedrooms + Number of Bathrooms + Square Foot Size + House Style} + Buyer Address Record
  - House Address Record = Street Address + City + State + Zip code
  - Listings = Listing Number + Listing Date + Listing Sales Price
    + House Address Record + Number of Bedrooms + Number of Bathrooms
    + Square Foot Size + House Style
  - Listings File Record = Listing Number + Listing Date + Listing Sales Terms
    + Listing Price + House Address Record + Number of Bedrooms + Number of Bathrooms + Square Foot Size +
    House style + Lot_ size + School district
  - Management Report = {Listing Number + Listing Date + Listing Sales Price
    + House Address Record + Number of Bedrooms + Number of Bathrooms
    + Square Foot Size + House Style}
- Data elements
  - Project dictionary entries have not yet been prepared for any of the data elements.

APPENDIX B: SYSTEM SPECIFICATION DOCUMENT FOR REAL ESTATE LISTING SYSTEM (CONT.)

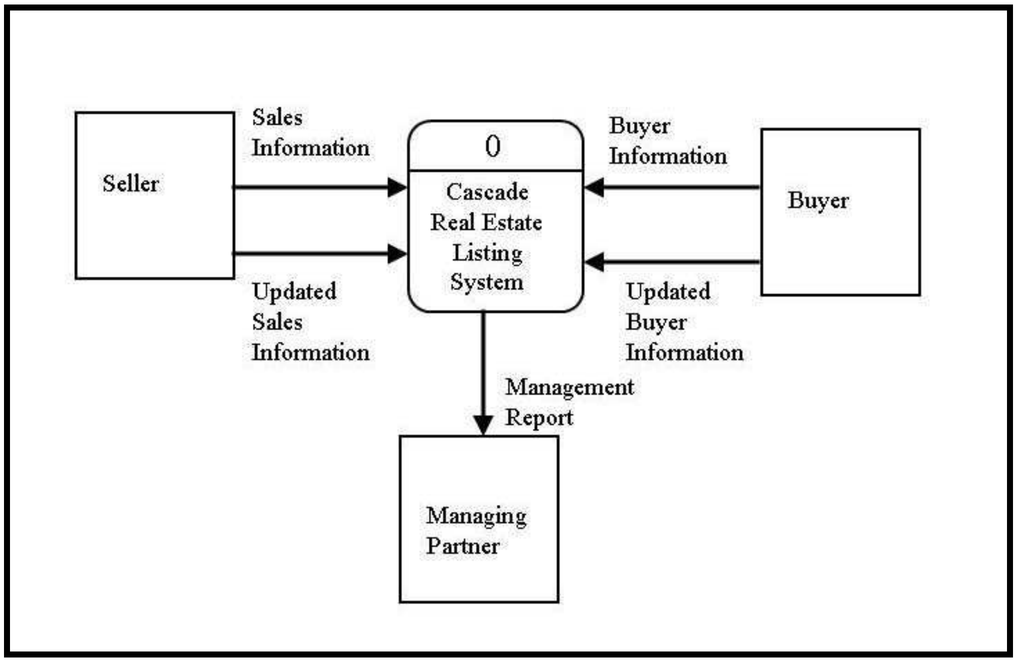
Seller
Sales Information
Updated Sales Information
0
Cascade Real Estate Listing System
Buyer Information
Updated Buyer Information
Buyer
Management Report
Managing Partner

Figure 1: Context DFD for Cascade Real Estate Listing System
(Note: Diagram was prepared by Joseph Analyst and may contain defects)

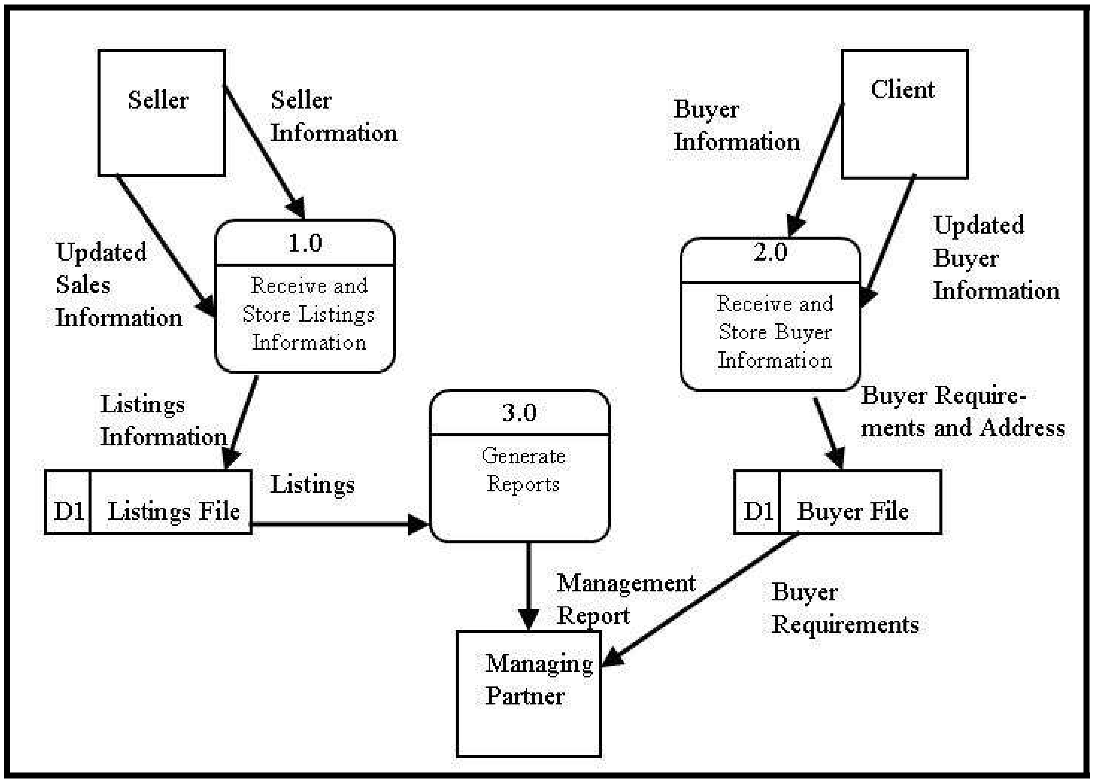

Figure 2: Level-0 DFD for Cascade Real Estate Listing System
(Note: Diagram was prepared by Joseph Analyst and may contain defects)

APPENDIX B: SYSTEM SPECIFICATION DOCUMENT FOR REAL ESTATE LISTING SYSTEM (CONT.)

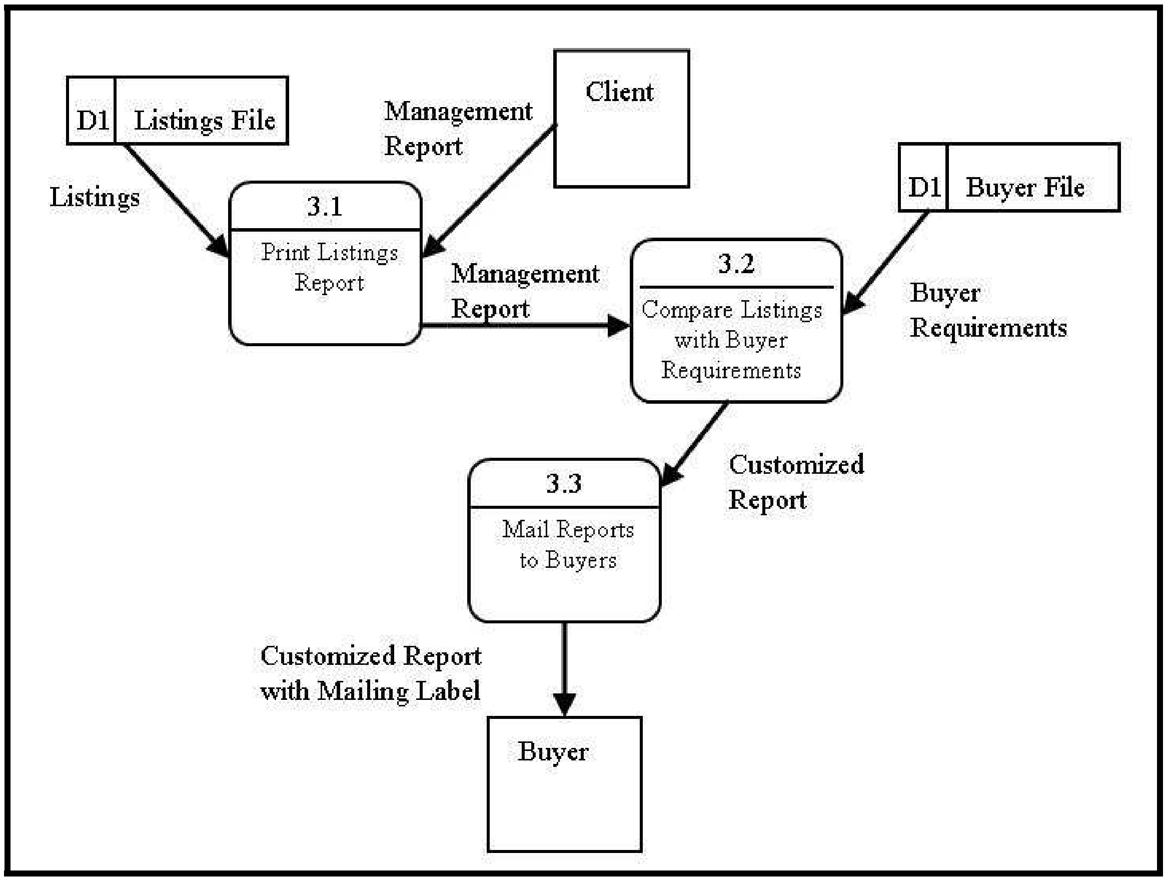

Figure 3: Level-1 DFD for Process 3.0, “Generate Reports”
(Note: Diagram was prepared by Joseph Analyst and may contain defects)

APPENDIX 1 – Extract from Module Template (2004/2005 delivery)

Module Characteristics

The development of modern information systems is a highly complex activity. A typical project often involves a team of people from different professional backgrounds and with many different skills. They may work together for many weeks or months to design and build software that meets the needs of its users. This module gives an insight into the many tasks that must be carried out during such a project. It provides a practical introduction to some of the techniques used at different stages of a project. It also illustrates how these tasks fit together within the overall project framework, and how they can be managed to ensure that the aims of the project are met.

The intention of this module is to provide the student with a practical, integrated overview of the Information Systems (IS) development process, from project selection and inception, through the capture and analysis of user requirements, to the design and production of a simple prototype system that satisfies those requirements. A constrained case study is used to take the student through a complete structured development cycle.

The module also introduces relevant theory including: the concept and different types of IS; the impact of IS on people, organisations and society; the systems development lifecycle (SDLC) and the various forms it can take; the nature and purpose of abstraction; typical models created during systems analysis and design; the key documents produced at stages of the SDLC; and the evaluation and review of a development project.

The module lays a foundation of skills and understanding for a number of later modules including (but not limited to) Database Design and Implementation, Object-Oriented Systems Analysis and Design, Comparative Systems Development Methodologies, Information Services Management and the Final Year Project. It also provides an understanding of the context in which all IS work is undertaken, and thus helps the student to develop a coherent view of their future profession.

Learning Outcomes

1. Explain key concepts in the Information Systems domain, and discuss the impact of IS on individuals, organisations and society.
2. Explain the role, significance and typical activities of project selection, project management, systems analysis and design.
3. Apply appropriate techniques to produce a requirements specification and design for a constrained case study, based on supplied information about user requirements.
4. Apply practical systems development skills to implement a prototype system in an environment such as MS Access.
5. Evaluate the extent to which the implemented system satisfies user requirements.

Indicative Content

- Information Systems: definition, types and components of.
- Systems Development Lifecycle: traditional (waterfall); alternatives to: iterative & incremental, spiral, prototyping.
- Project Selection and Feasibility Assessment: overview of Cost Benefit Analysis; feasibility presentation and report; impact on organisation, individual and society; professionalism and ethics.
- Project Management: role and significance; use of simple techniques such as work breakdown structure and Gantt chart.
- Systems Analysis: function of, tasks undertaken. Abstraction: reasons for and forms of.
- Fact Finding Techniques: SQUIRO. Models for requirements capture, e.g. Use Cases.
- Models for requirements analysis, e.g. Data Flow Diagrams, Entity-Relationship Models. Models for design, e.g. Entity-Relationship Models, Activity Diagrams.
- Documentation such as Requirements Specification, Design Specification: function of and typical format.
- Design issues: elements of HCI, form/report/navigation design. Data design. Process design. Basic elements of systems architecture.
- Realisation of design using, e.g., MS Access. Creation of tables from design model; designing suitable input forms; writing and executing queries using SQL; producing suitable output (screens and reports) to meet user requirements as specified in Requirements Specification.
- Review of prototype produced against user requirements.