Complete Systems Analysis

James & Suzanne Robertson

foreword by Tom DeMarco

The Workbook, The Textbook, The Answers

DH

Complete Systems Analysis

Complete Systems Analysis

James & Suzanne Robertson

foreword by Tom DeMarco

Dorset House Publishing
3143 Broadway, Suite 2B
New York, New York 10027

Library of Congress Cataloging in Publication Data

Robertson, James.
Complete systems analysis : the workbook, the textbook, the answers / by James Robertson, Suzanne Robertson.
p. cm.
Includes bibliographical references and index.
1. System design--Programmed instuction. 2. System analysis--Programmed instuction. 3. Computer-aided software engineering--Programmed instruction. I. Robertson, Suzanne. II. Title.
QA76.9.S88R633 1994 93-44616
004.2'1--dc20 CIP
ISBN 978-0-932633-50-7

Cover and Interior Page Designs, Hardcover Edition: Ellen Moorcraft
(Designs Revised for the Softcover Edition by David McClintock)

Distributed in the English language in Singapore, the Philippines, and Southeast Asia by Alkem Company (S) Pte. Ltd., Singapore; and in the English language in India, Bangladesh, Sri Lanka, Nepal, and Mauritius by Prism Books Pvt., Ltd., Bangalore, India.

Printed in the United States of America

Library of Congress Catalog Number: 93-44616

ISBN: 978-0-932633-50-7 12 11 10 9 8 7 6 5 4

Digital release by Pearson Education, Inc., June, 2013

For Reginald and Helen

This page intentionally left blank

CONTENTS

ACKNOWLEDGMENTS

We have always disliked the acknowledgments sections of books. The authors present an incredibly humble face to the reader, which, we assume, is to make the reader think that the authors are humble, and therefore likable people. We also assume that this display of humility is intended to make the reader like the book all the more.

We now know better.

Authors are humble because writing a book is an incredibly humbling experience. Once written down, thoughts that are entirely clear and lucid appear awkward and ambiguous. Explanations that work in normal conversation become completely inappropriate when set in type. Diagrams that we are convinced are models of ingenuity and perfection are exposed as faulty and somewhat pedestrian.

What this all means is that without a lot of help, authors, or at least the authors of this book, would never publish anything readable. We therefore make these (humble) acknowledgments to the people who have helped us to bring you this book.

We acknowledge the inspiration over the years, and help with this book, given to us by our fellow principals at the Atlantic Systems Guild: Tom DeMarco, Tim Lister, Steve McMenamin, and John Palmer. They all have provided us with many of the insights we present in this book.

We should also mention the major influences of DeMarco's *Structured Analysis and System Specification,* McMenamin and Palmer's *Essential Systems Analysis,* and Matt Flavin's *Fundamental Concepts of Information Modeling.*

Drafts of this book have been reviewed and corrected by many people. We acknowledge the invaluable help given by the team working in Rich Cohen's CompuServe forum: Darrin Chandler, Thor Christensen, James Curran, Michael Diehl, Harry Holt, Jim Hudson, Bob Koss, Jim Langendoen, Lucy Lockwood, Rud Merriam, Tom Ochs, Sue Petersen (who suffered more than most to bring you this book), Russ Ranshaw, Jeff Schweiger, Bryan de Silva, Mark Washick, and Mark Weisz. Special thanks to Rich Cohen for his Herculean efforts in keeping the group focused.

We also acknowledge contributions from our colleagues Truitt Allen, Gary Austin, Therese Lange, Gary Schuldt, Andy Smith, and Nina and Lee Snett. Their suggestions have made this a better book.

This book has seen the insides of upwards of a dozen Macintosh® computers, starting with the first drafts on early Macintosh Plus machines, and ending with the typesetting on a Quadra. The text was written using Microsoft Word™ and most diagrams were drawn with Aldus Freehand™. The Piccadilly Project models were built using TurboCASE™. We gratefully acknowledge the help given us by Shang-cheng Chyou of StructSoft. His words of advice saved us many hours of work.

The original project at Associated Television provided us with the wonderful experience of working with some very talented people. We particularly want to acknowledge the work of Kenda Harris and Wendy Wakley (who is even better at her job than her Stepney Green counterpart in this book).

The design of this book and the cover is the work of Ellen Moorcraft. We think her work speaks for itself. Paula Gair helped with the preparation of the manuscript by organizing and checking its hundreds of components. Meanwhile Ian Gair kept our consulting business running so smoothly we didn't mind taking the necessary time off to write this book. We wish to thank many of our clients for their encouragement during the book's long gestation.

Finally, the publisher. No book would ever see the light of day without some amazing work by the publisher. David McClintock, Janice Wormington, and Tony Yip of Dorset House patiently guided us through the process, made our English readable, and, most importantly, provided encouragement and guidance during the dark hours of despair when it seemed that we would never finish.

To all of the above, a very grateful, and humble, thank you.

FOREWORD

The past decade has been one of striking change for the discipline of systems analysis. As recently as the mid-1980s, analysts everywhere were still inclined to dictate to their users what the new system would be like. They typically passed down their pronouncements in the form of a written specification. Today, that approach will no longer fly anywhere. Today, we analysts find ourselves serving not so much as inventors of the new system but as catalysts for the invention process. Instead of text, we rely on models of all kinds: function models, data models, object schemas, state models, prototypes, GUI frameworks, and the like.

At the key interface between analyst and user, the dialogue has changed, too. Before, we tended to say, "Here's what you're getting." Now, we show one of the models and ask, "How about something like this?" Once a model is on the table, the process comes alive. The users get their hands on the model and start to reshape and remold. "Close," they say. "If only we could change it, though, to something like this . . ." When that begins to happen, I know the project is on track.

When we first show users a model, we know it is, at best, close to what is required. We show it specifically to elicit change. As such, the model isn't supposed to be an exact replica of the system, but rather an example of the kind of system we'll probably be building. Modern analysis is, accordingly, a "by-example" discipline. We show examples at each and every stage of the process.

In *Complete Systems Analysis,* Suzanne and James Robertson have hit upon the charming idea of guiding you through this by-example discipline with a by-example presentation. From the very first page, an extraordinary and wonderful page, you will know this book is fundamentally different from any other analysis texts you may encounter. It doesn't lecture at you, it doesn't take up your time telling you anything you already knew. It's a book that you don't exactly read at all; instead, you sort of ski through it, along a path of your own choice. (Well, they'll explain all about that.)

When my own analysis book was still in manuscript, the publisher sent it out for evaluation to a number of referees. Among the comments that came back to me from this process was the following one: "I guess you can't really like this book unless you're willing to like the author." Since that particular referee had never shown herself to be "willing to like the author," I knew the comment was supposed to be a criticism. But I felt just the opposite. I felt it was the nicest possible compliment. After all, I wasn't trying to hide behind the page, to make myself anonymous. The books I had most admired in our field (Fred Brooks's *The Mythical Man-Month,* for example, or virtually any of Jerry Weinberg's works) were personal and personally revealing and more meaningful because of it. That was the kind of book I had been trying to write.

Complete Systems Analysis is similarly personal. You'll come to know its authors, Suzanne and James Robertson, as you read through their work. I predict you'll find them (like the book itself) to be honest and on-target and funny and inventive and curmudgeonly and wise . . . but, most of all, honest. There is something of their own lives and professional experience on every page. The work they share with you is a real one, based on their extensive involvement in automation within the British television industry, as well as numerous other organizations. The lessons they drew from that project at the time and redraw for you now are not simplistic—there are no easy answers in systems analysis—but they are useful lessons. They help you to discover some of what is the most useful, more or less the way they discovered it themselves. You will probably make a few of the same mistakes they made in the discovery process, and if you can profit from making those mistakes in the safe environment of these pages instead of in your own work, the benefits should be obvious.

October 1993
Camden, Maine

Tom DeMarco

SECTION 3

Project Reviews

This page intentionally left blank

REVIEW: START WITH THE CONTEXT 3.1

Before You Reached Here ...
You have done the problem set in Chapter 1.2 *Start with the Context.*

The Context Diagram for Piccadilly Television

The context diagram that we derived from the description of Piccadilly Television is shown in Figure 3.1.1. Compare your diagram with it.

The Case of the Missing Users

In some cases, we've interpreted something that was said in a user interview or statement, and consequently your names may vary from ours. As long as the intention is comparable, though, your answer is acceptable. Of course, when you're doing real-world systems analysis, you'll need more than just good intentions. You'll need the users to help you out.

Only the users can confirm the factual substance of your models. The goal of analysis modeling is to have a close collaboration between the users and the analyst. Because you are working from a book and there are no users sitting with you at your desk, you have to compensate. For this Project, concentrate on the form of your answer and its intent. There are no users present, and so your interpretation of the substance is, within reason, as good as ours.

The Boundary of Your Project

Probably the hardest part of the exercise is determining precisely what is inside and what is outside the system. When you draw the context diagram of the system, you are determining exactly what you are going to study. For our solution, we looked for those things that Piccadilly controlled and that could be studied reasonably by an analyst. We also chose to study parts of the business that we thought we could affect if we devised some way to improve the process. We declined to study anything that was owned by other organizations or any part that we couldn't influence or improve.

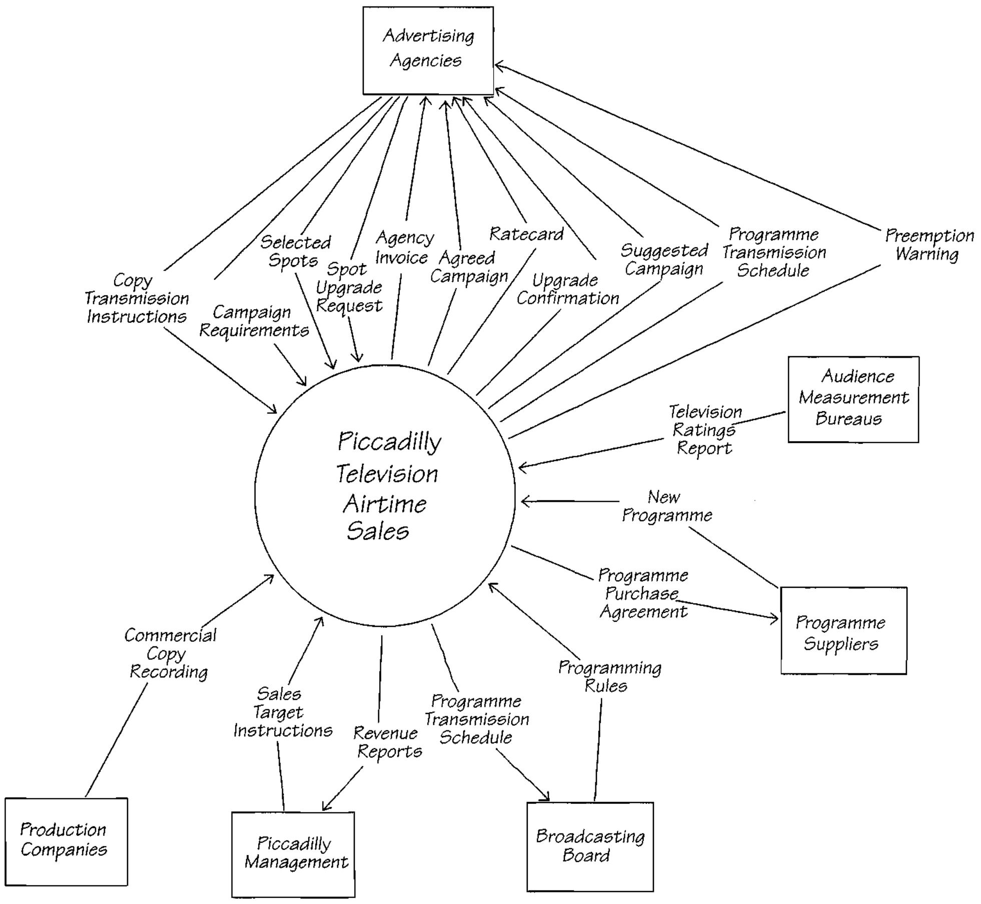

Figure 3.1.1: Our interpretation of the Piccadilly Television context diagram, which summarizes the scope of the Project. The names used for the terminators and data flows are, as much as possible, those used in the text.

Interpreting the Business

You had a written description of the operations of Piccadilly as input to the problem. Like all pieces of natural language prose, it was ambiguous. When you drew the context diagram, you imposed an interpretation on it. Whether or not you made the right interpretation is now to be decided by you and your users. By showing the users your context diagram, you are saying, "This is where I think the project boundaries are. These are the data flows that are at the limits of the system's responsibilities. Now let's go through each of them to see if we all share the same understanding of the scope of this system."

For example, the information communicated between the advertising agencies and Piccadilly sales executives may have been ambiguous as it was presented in the text. On the other hand, the context diagram states very clearly that the ADVERTISING AGENCIES send CAMPAIGN REQUIREMENTS, SELECTED SPOTS, SPOT UPGRADE REQUEST, and COPY TRANSMISSION INSTRUCTIONS to the Piccadilly system. This means that the system you are studying begins its responsibility when it receives these data flows. So you are declaring that you intend to study how the sales executives process these data. If you and the users do not intend to study this part of the system, you must remove these data flows from the diagram.

You proceed the same way for the data flows leaving the context. The data flow AGREED CAMPAIGN leaves the system and goes to the ADVERTISING AGENCIES. The diagram states that your analysis will study how that flow of information was created. However, the responsibility of the system ends once the data making up AGREED CAMPAIGN have been produced. What the agencies do with the data is outside the scope of your study.

The text mentions that the ADVERTISING AGENCIES communicate with film PRODUCTION COMPANIES. The agencies must be telling the production companies which commercials to make and where to send the commercial copy. Why doesn't this communication show up in the context diagram? Consider the other data flows between these two terminators and your context. They all point to the fact that the commercial has already been made by the time Piccadilly hears about it. If your system were interested in the production of commercials, these boundary flows would be very different. So you must rule out any interest in the communication regarding the production of commercials. Your interest in the PRODUCTION COMPANIES is limited to the fact that they are the ones that send COMMERCIAL COPY RECORDING to the system. Your diagram states that you intend to study the processing that takes place after Piccadilly has received this data flow.

It's the Message, Not the Medium

Did you include the data flow PREEMPTION WARNING in your diagram? You should have. Recall that the sales executive telephones the agency to warn his contact about a possible preemption. Even though the data are traveling by voice and telephone line, the information is still a data flow. You will have to find out what processing and data are used to create this warning.

Your context must include all of the interfaces between your project and the terminators, not just those interfaces in the form of documents or other tangible media.

Internal or External?

The terminator PICCADILLY MANAGEMENT receives REVENUE REPORTS and produces SALES TARGET INSTRUCTIONS. Our diagram says that you will study how the revenue reports are produced and how the sales target is used. By treating PICCADILLY MANAGEMENT as a terminator, you are stating that you will not study either what managers do with the revenue reports or how they set the sales target.

The context of your study is not defined by the terminators; it is defined by the data flowing to and from the terminators.

Although PICCADILLY MANAGEMENT appears as a terminator, that doesn't exclude some of the processes within your study from being carried out by the same managers. For instance, you heard that managers are concerned with setting new rates so that a new RATECARD can be sent to the agencies. When you study the Project in detail, you'll find that management is involved in one of the processes inside your context of study.

Your interpretation may well differ. Suppose, for example, you decided that your project will study the rules for setting the sales target and using the revenue reports. Then, both data flows, SALES TARGET INSTRUCTIONS and REVENUE REPORTS, would disappear from the context diagram. They would become data flows inside the system, and they would reappear when you studied the details of the project.

Which of the above interpretations is right? The answer is they both are. Almost all of the information you receive when analyzing a system can be understood in more than one way. This problem of multiple interpretations is the reason to build a context diagram. The context diagram totally eliminates the ambiguity because it can be interpreted in only one way. While this doesn't make the diagram correct every time, it does mean that because the users can see your precise understanding of the system, they can either agree that you show it as they mean it, or disagree and ask you to change it. The point is that you make it possible to raise questions and to work toward a consensus.

No doubt there will be changes to the context as work on the system progresses. New requirements will be added to the system, while other requirements, previously thought indispensable, will be deleted. Many adjustments will result when you start to do the detailed analysis and everyone realizes just how much work is involved. The only real constant is that the context diagram will continue to display the precise boundaries of the system. How and where the boundaries are set is negotiated by you and the users.

Naming the Flows

Always try to name the data flows with the names the users know them by, since it makes your models much more recognizable to the users. However, sometimes you'll find names with buried meanings. For instance, the first time we talked to the Piccadilly users, they referred to the "green book." Since that is not too informative, we needed to clarify the meaning: "What do you mean by 'green book'?" we asked. "What do you use it for? What data does it hold? Show us a green book." The buried meaning turned out to be the data flow TELEVISION RATINGS REPORT. (The book, as we found out, wasn't even green at all. At some time, the procedure changed and the ratings came in a yellow binder. This did not, however, cause people to change their terminology in the least.)

Try to use the same names as the users do, but make sure the names reflect the real business purpose of the data flow.

Were you tempted to abbreviate the names? Maybe you used C REQ instead of CAMPAIGN REQUIREMENTS. Don't do it. Otherwise, you create your own buried meanings just as obscure as some of those the users have invented. Remember, your objective is to understand the business requirements so that you can raise relevant questions. You want to uncover misunderstandings before a lot of detailed work has been done. That means adapting to the language of the business you're studying.

With a general description of the users' business, you've turned it into an unambiguous statement of the Project's context of study. You now have a well-defined starting point. Later, your detailed analysis may reveal things that change the context. However, the context diagram helps to keep the Project under control. Any change to the context means a change to the size of the Project. Each change can be measured for its impact on the Project, as we'll discuss.

✚ Ski Patrol

The ✚ Ski Patrol can help if you are having trouble with the Piccadilly Project, especially with the technical aspects of the model. If you are satisfied that your diagram conveys your intended meaning, that a user could understand it enough to question it, and that you can produce a reasonable context diagram for your next project, you have no need of the patrol. Proceed directly to the Trail Guide below.

If you have a technical problem with your model, read on. First, your model should have the same form as ours. There should be only one bubble, and about the same number of terminators as in ours. If not, we suggest you review the discussion in Chapter 2.2 *Data Flow Diagrams* about context diagrams. If your model shows several bubbles, you have already started to partition the system. While this is not necessarily wrong, our experience has shown that it always pays to have the context reasonably decided before breaking the system into its components. At this stage, you should be concentrating on the boundary data flows, instead of the functional areas.

We trust that you've not committed the crime of leaving any of your data flows unnamed. If any of the other data flow conventions gave you problems, again, return to Chapter 2.2 *Data Flow Diagrams,* work through the exercises, and study the sample answers. There will be more on data flow diagrams later in the book (Chapters 2.6 *More on Data Flow Diagrams* and 2.7 *Leveled Data Flow Diagrams).*

Trail Guide

● Easiest: Go to Chapter 2.3 *A Variety of Viewpoints* for a discussion of the viewpoints used to build analysis models.

■ More Difficult: Go to Chapter 2.3 *A Variety of Viewpoints.* This trail assumes that you already know how to build models, so it focuses on the viewpoints that you use to build them.

◆ Most Difficult: Go to Chapter 1.3 *What About the Business Data?* for more work on the Piccadilly Project.

❄ Promenade: The Project chapters are not on your trail, but if you have found this interesting, you might consider switching trails. Otherwise, you can pick up your own trail in Chapter 2.1 *Analysis Models.*

REVIEW: WHAT ABOUT THE BUSINESS DATA? 3.2

Before You Reached Here ...

You have built the first-cut data model for Piccadilly Television. This chapter gives you a sample answer and a discussion of how we derived our model. While we asked you for a first-cut model, we have cleaned up the model to an extent by eliminating entities and relationships that you may have included on your first pass.

The important task in this chapter is not for you to see if your model is a precise match to ours, but for you to be able to rationalize all the decisions that you made to build your model and to understand our reasoning. In some cases, we admit we have cheated. Since we have already analyzed this system, we possess background knowledge that you don't have. Even if we had tried to give you all this background knowledge in the problem statement, there still would be variations between your model and ours. It's more important that you make a model of how you interpret the policy, and that you raise questions for the users, than to simply grind through a mechanical translation into a data model. In doing systems analysis, you can never get a thorough and complete statement the first time. However, once you have a first-cut model, you do have some control over the analytical tasks of questioning and improvement.

Looking for Potential Entities

We gave you a rule of thumb that says when you have a description of the business, look for nouns to indicate potential entities. Let's apply it to part of "The Story of Piccadilly Television" from Chapter 1.2 *Start with the Context.*

"The advertising agencies buy commercial spots that make up campaigns to advertise the products they represent. Each agency sends its campaign requirements to the Piccadilly sales executive who deals with that agency. The executive then models the campaign by selecting commercial breaks for the spots to occupy that will be profitable to Piccadilly, and that will deliver the required ratings to the advertiser. When the executive is satisfied with his selections, the suggested campaign is communicated to the agency. The agency responds by selecting spots from the executive's suggestions and informing him of the choices. The executive finalizes the deal by sending the agency written confirmation of the agreed spots that make up the campaign."

The nouns in the description are *advertising agency, commercial spot, advertising campaign, product, campaign requirement, sales executive, commercial break, rating, advertiser, selection, suggestion, choice,* and *confirmation.* Note that in this listing, we have expressed the nouns as singular and used more descriptive names for *advertising agency* and *advertising campaign.* So let's use these nouns as names for potential entities.

Are These Entities Relevant?

Testing for relevancy means asking if the system has a genuine business need to remember the facts that describe that entity. Ask, for instance, if there is a legal reason for storing those facts. Are they used by people or computers to do their jobs? Are they retrieved for reports? To illustrate a test for relevancy, you could ask if Piccadilly needs to remember anything about the potential entity ADVERTISING AGENCY. You already know that Piccadilly sells commercial airtime to the agencies, so there is a need to remember who the customers are. At the very least, Piccadilly is interested in the name and address of the advertising agency in order to mail the invoices. Given this need to remember, add the entity ADVERTISING AGENCY to your data model. You can test this entity by looking at the boundary data flows for data elements that must be remembered about agencies. If there are any, these data elements become attributes of ADVERTISING AGENCY, and confirm that this is a relevant entity.

The rule of thumb gave us a potential entity called CAMPAIGN REQUIREMENT. These are sent by the agency to the executive who uses them to plan and negotiate the campaign. Once the campaign is settled and the agency has agreed to the spots, there is no requirement for the executive to remember the agency's original requirements. (You would have to get this decision from the users.) Any facts that must be remembered are attributes of ADVERTISING CAMPAIGN (there is a need to remember campaigns), so CAMPAIGN REQUIREMENT fails the relevancy test and can be omitted from the model. Similarly, ADVERTISER, SELECTION, SUGGESTION, CHOICE, and CONFIRMATION are potential entities that the system has no reason to remember. They fail the relevancy test and they, too, are omitted from the data model.

The entities we determined to be relevant from this test are shown in Figure 3.2.1. The next step is to find any relationships between these entities.

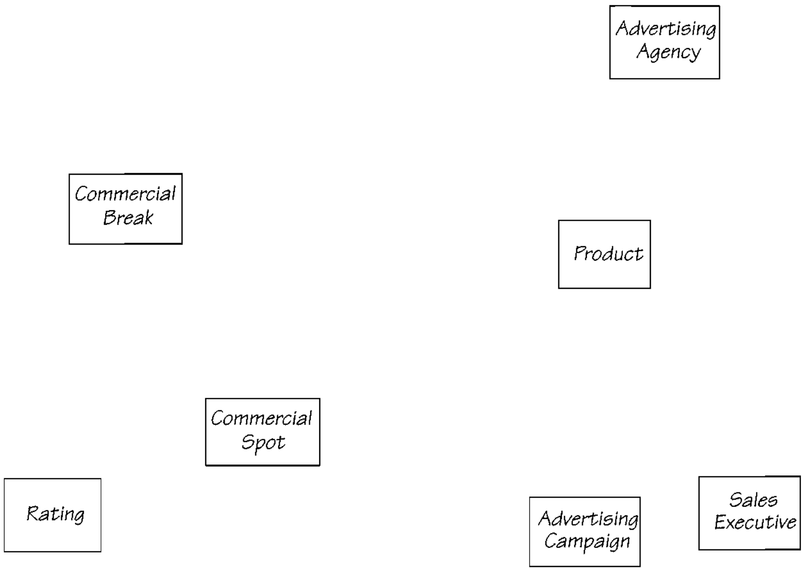

Figure 3.2.1: *These seven entities are a starting point for the data model. Detailed analysis will specify the precise use of each entity and will add to and refine the data model.*

Finding Relationships

Our rule of thumb also says that verbs in business descriptions indicate relationships. If we look again at the same paragraph:

"The advertising agencies buy commercial spots that make up campaigns to advertise the products they represent. Each agency sends its campaign requirements to the Piccadilly sales executive who deals with that agency. The executive then models the campaign by selecting commercial breaks for the spots to occupy that will be profitable to Piccadilly, and that will deliver the required ratings to the advertiser. When the executive is satisfied with his selections, the suggested campaign is communicated to the agency. The agency responds by selecting spots from the executive's suggestions and informing him of the choices. The executive finalizes the deal by sending the agency written confirmation of the agreed spots that make up the campaign."

The verbs and gerunds (words ending in "ing" that are the noun form of a verb) in the above are *buy, make up, advertise, represent, sends, deals, models, selecting, occupy, deliver, satisfied, communicated, responds, selecting* (again), *informing, finalizes, sending, make up* (again). Let's treat each of these as potential relationships, and test them for relevancy. For each relationship, test its relevancy by asking, "What is the reason for this relationship? Does the system need to remember this relationship?"

Examine the first potential relationship: "The advertising agencies *buy* commercial spots ..." First, let's discuss the reason for the relationship. You can see in the context diagram (Figure 3.1.1) 'a number of data flows between advertising agencies and the system. CAMPAIGN REQUIREMENTS, SELECTED SPOTS, and SUGGESTED CAMPAIGN exist because the agency is deciding to buy some spots. Selling spots to the agencies is a necessary part of Piccadilly's business, so you can say that the reason for relating an agency to its commercial spots is that the agency is buying them.

The context diagram also yields the answer to the second question about the need to remember the relationship. After transmitting spots, Piccadilly sends an AGENCY INVOICE to the agency. The invoice lists the spots the agency is buying. Because Piccadilly must be able to identify which agency is buying which transmitted spot, this establishes the need to remember a relationship between agency and spot. The result of the relevancy test is that now you can say an agency and a commercial spot participate in a BUYING relationship.

These verbs—*sends, deals, models, selecting, satisfied, communicated, responds, selecting* (again), *informing, finalizes, sending*—did not pass the relevancy tests. "Each agency *sends* its campaign requirements to the ..." You've already omitted campaign requirements from your model, so there can be no relationship with it. "The executive then *models* the campaign by *selecting* commercial breaks ..." Does Piccadilly have a reason to remember which executive models a campaign, and which modeling effort selects which commercial breaks? It is highly unlikely. However, there is a strong reason for remembering which spots *make up* the final composition of a campaign, but modeling the campaign is a temporary state, and not worth remembering. Similarly, Piccadilly has no need to remember which breaks are part of the executive's model: They aren't invoiced, and sales executives cannot reserve spots for their clients. Similarly, *communicated, satisfied, selecting, responds, informing, finalizes,* and *sending* are all transitional in nature, and there appears no good reason to remember them. If you have included any of these in your model, don't be too worried at this stage; remember that these are still *potential* relationships. Later analysis will confirm or deny the need for them and discover any that are missing, as you'll see later in the Project.

After examining all of the potential relationships, we have added the relevant ones to the model (see Figure 3.2.2).

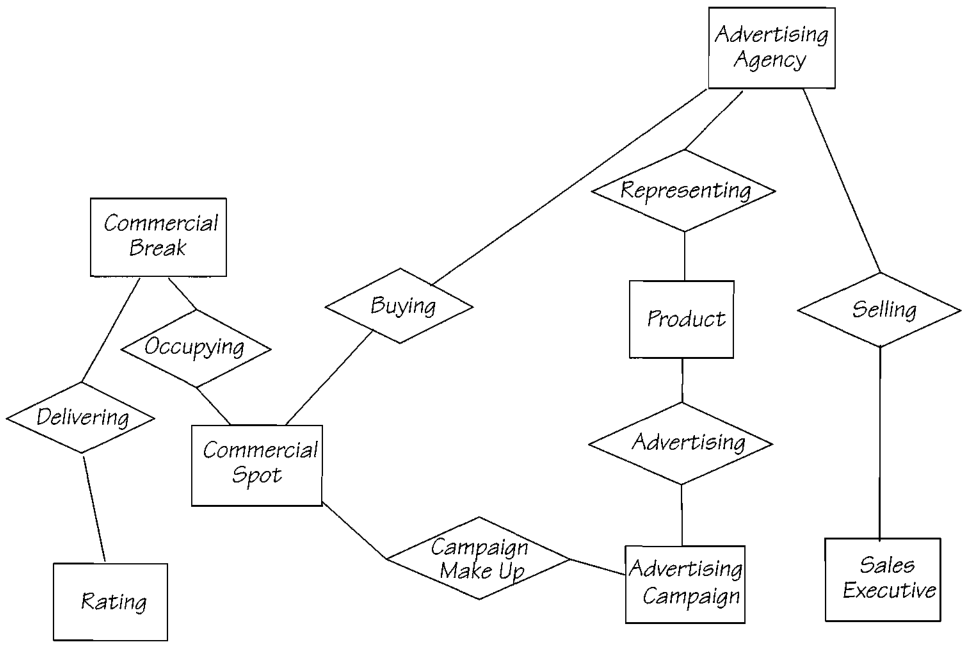

Figure 3.2.2: *The data model with relationships added. We derived the relationships initially by identifying the verbs and gerunds in the Piccadilly description, and then testing each one for relevancy.*

Adding Cardinality

Adding cardinality to each relationship in your data model is a way of learning more about the data. For each relationship in the data model, ask this question of the entities at either end of the relationship: "For one instance of this entity, how many of the other entity can participate in this relationship?"

For example, you know that the "advertising agencies buy commercial spots that make up campaigns to advertise the products they represent." This tells you that for one agency, there are potentially many spots. From your knowledge of Piccadilly, you know that one campaign is made up of many spots, and you suspect that one agency has many products. If you question the entity at the other end of the relationship, you may find that one spot can be bought by only one agency. (If another agency wants it, the second agency must pay a higher price and preempt the first agency.) Also, a spot may belong to only one campaign, and a product is represented by only one agency at a time. Piccadilly keeps a credit history on previous purchases by agencies that owe money because Piccadilly needs a way to follow up on bad debts.

Only one sales executive is assigned to an agency, and although an executive may have more than one agency, you are not given any evidence of it. If you ask the executives, they each will confirm that they do indeed deal with more than one agency. You can reasonably assume that a product will have more than one campaign, and that one break will carry several spots. Breaks are two or three minutes long, and Piccadilly has to ensure that the executives fill the breaks with spots.

The model in Figure 3.2.3 shows the cardinality that results from testing the existing relationships.

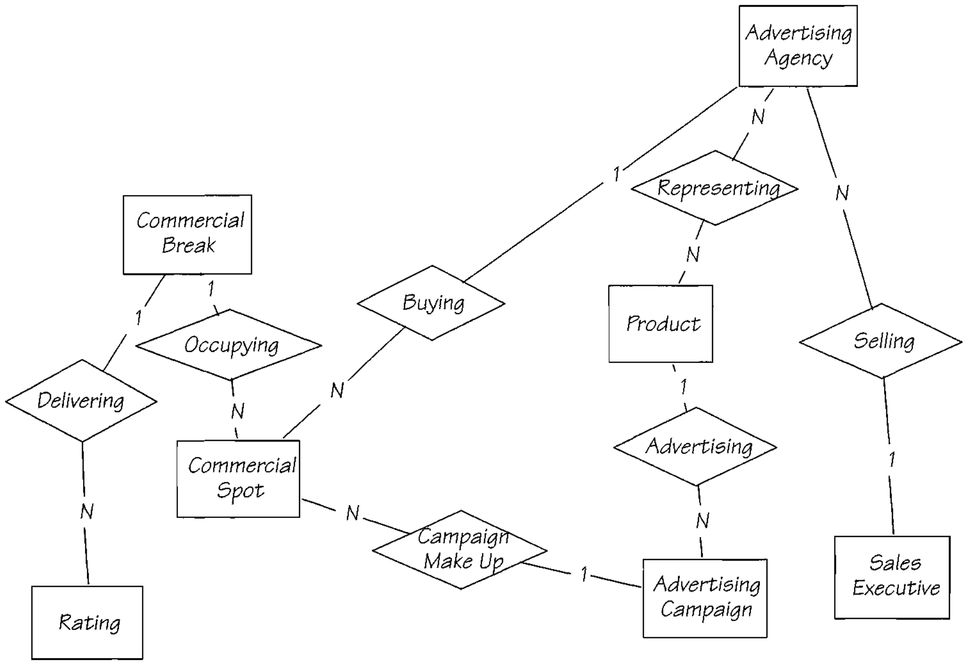

Figure 3.2.3: To determine the cardinality of relationships, ask how many of each entity participate in a relationship. Knowing the cardinality gives you a better understanding of the data.

Building a data model always raises interesting questions, and you can learn a lot about the system from answering them. For example, we said that a product probably has more than one campaign, but can an advertising campaign be for more than one product? The answer is no. Piccadilly considers it too difficult to get paid for multi-product campaigns. While an advertiser may use copy that promotes several products, Piccadilly writes its contracts as if there is only one product.

Can a product be represented by more than one agency? The answer is that only one agency can represent a product. However, sometimes a product changes its agency. In that case, Piccadilly keeps track of the agency that previously represented the product because of the need to collect all its debts.

Can an agency be in a selling relationship with more than one sales executive? The users' answer to this one is no. Piccadilly management is only interested in which sales executive has the current responsibility for selling to an agency. The answers to these questions show up as cardinal operators in the data model.

So far, the data model has defined only a subset of your knowledge about Piccadilly. Let's look at some of the other parts of the policy statement from Chapter 1.2 *Start with the Context.*

"Piccadilly produces some of its own programmes, and buys others from a variety of programme suppliers both in England and overseas. These programme suppliers inform Piccadilly of their offerings, which include first-run films, sporting events, documentaries, talk shows, and old movies. Some of the programmes, such as the talk shows and documentaries, may be a series with a number of episodes."

Now you have some more entities: PROGRAMME SUPPLIER, PROGRAMME, and EPISODE. Not all programmes have episodes. For example, a movie is a stand-alone programme. However, television broadcasters like to show movies under an umbrella programme name like "Prime Time Movie," "Movie of the Week," and so on. Therefore, it's easier to think of *all* programmes as having a number of episodes. The entities and relationships for this piece of the policy are shown in Figure 3.2.4.

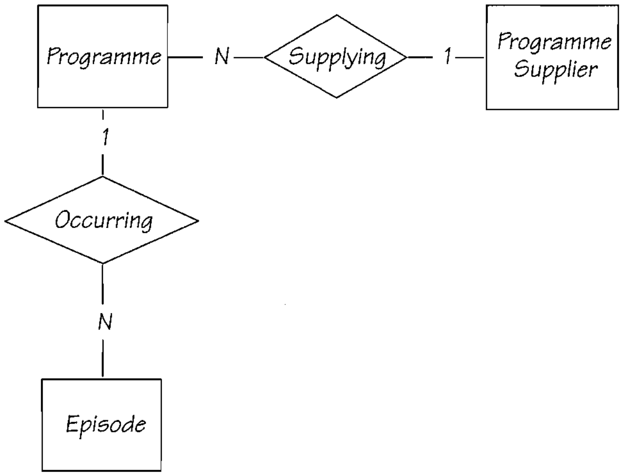

Figure 3.2.4: These entities and relationships show what Piccadilly remembers about the programming part of the system.

Let's go on: "Piccadilly's programme schedulers have the complicated job of deciding the date that each programme should be transmitted, and where in the programme the commercial breaks should be placed. To make these decisions, the schedulers use the weekly ratings that are supplied by the audience measurement bureaus and that tell them how many people are watching which programmes. The schedulers must also follow the Broadcasting Board's rules for placement of programmes and for the number and placement of commercial breaks within those programmes. Four times a year, the schedulers set a new programme transmission schedule for the coming quarter."

There is nothing in this description to tell you why this system needs to remember anything about the programme schedulers. Perhaps another system such as payroll may need to, but unless this system is to report on schedulers' performance or suchlike, then you can eliminate the scheduler as an entity. The result of the schedulers' efforts is the quarterly schedule. While there is a need to remember it, the published schedule is made up of the programmes and commercial breaks that the schedulers have decided. As there is only one schedule for this system, and as we already have entities and relationships that provide all the necessary information (PROGRAMME, EPISODE, COMMERCIAL BREAK), there is no need to have an entity called "schedule."

The term "programme" is being used loosely here. If we are thinking there are many episodes of each programme, it's the episode that is broadcast, not the programme. This means that COMMERCIAL BREAK relates to EPISODE, as the break's proximity to the programme content of the episode is what attracts advertisers. The EPISODE will be one of many broadcasts on a given date.

The schedulers use RATING to anticipate the audience that each episode will attract. If an existing programme is continued in the new quarter's schedule in the same time slot, its ratings will be similar, with allowances made for seasonal factors (ratings are higher in the fourth quarter of the year), and for competing channels' programmes. For new programmes, the schedulers look at the actual ratings of similar programmes to work out the predicted ratings. The sales executives use the predicted ratings when they are selling commercial spots. So there are two types of ratings: predicted and actual. The upshot of this is that RATING relates to EPISODE, and that RATING has two subtypes: PREDICTED RATING and ACTUAL RATING. From this part of the business, we deduce the entities and relationships shown in Figure 3.2.5.

The entity DATE isn't shown in Figure 3.2.5, and you may not have it in your model. After all, the date of transmission could be an attribute of EPISODE. However, the users will tell you that there is a "day of transmission" or, as Piccadilly calls it, "breakchart day." Usually, the station starts its broadcasting day with the morning talk shows, and finishes some time in the early hours of the following morning with the "Late, Late Show." (We are just as perplexed as you why something in the early hours is called "late.") While this constitutes a *day's* transmission, it actually covers two *dates.* Look again at the ratecard in Chapter 1.2 (Figure 1.2.1) and answer this: If an advertiser buys a spot in the late Friday segment, and the spot is not transmitted until after midnight, what rate does he pay? The answer is the weekday 23.40-to-close rate. So the breakchart day is different from the actual date. We thus change the name of this entity to make it more appropriate to the system.

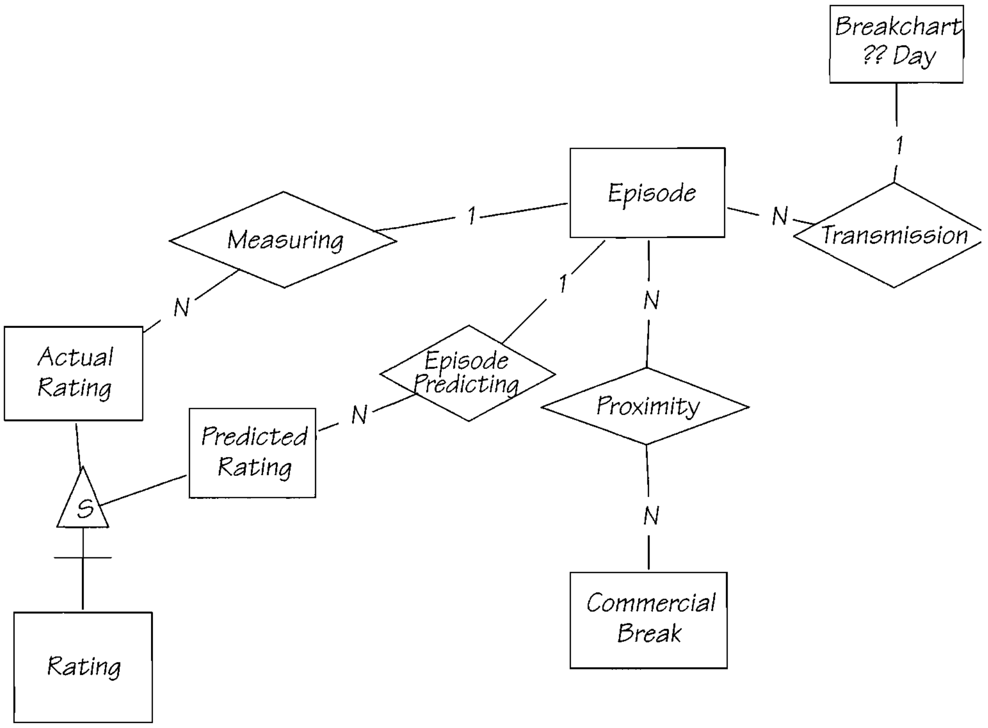

Figure 3.2.5: *A partial data model for the scheduling part of the system. Note that some of the entities in this model have already appeared in other partial models.*

How do you show the rules provided by the Broadcasting Board in a data model? One approach is not to show them: They could be considered as part of the scheduling process policy, and not as stored data at all. However, in this case, the PROGRAMMING RULES are shown in the context diagram as a data flow entering the system. So there must be a process like the one in Figure 3.2.6.

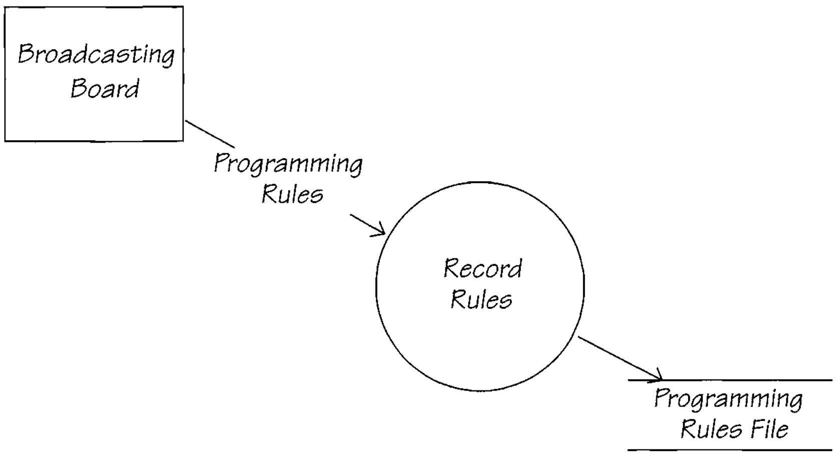

Figure 3.2.6: *The programming rules from the Broadcasting Board are stored for use by the schedulers.*

You can safely assume that the rules will change from time to time and that there may be special rules that apply to special seasons. So it makes good business sense to have the rules remembered. This gives an entity that we call BROADCASTING RULE because it is concerned not just with programmes but also with commercial breaks. But to what does it relate? The statement "The schedulers must also follow the Broadcasting Board's rules for placement of programmes and for the number and placement of commercial breaks ..." indicates that the rules relate to the scheduling of breaks within and around programmes. In other words, an entity relates to a relationship. This seems a little mind-boggling, but we can neatly solve the problem by having a three-way relationship between the participating entities (see Figure 3.2.7). Similarly, the rules apply to scheduling episodes of programmes, and so they can participate in the SCHEDULING relationship. The way to make sense of this relationship is to view it in three different ways:

- For each instance of one EPISODE and one COMMERCIAL BREAK, there are many instances of BROADCASTING RULE.
- For each instance of one COMMERCIAL BREAK and one BROADCASTING RULE, there are many instances of EPISODE.
- For each instance of one EPISODE and one BROADCASTING RULE, there are many instances of COMMERCIAL BREAK.

As you can see, *n*-ary relationships are more complex to understand than binary relationships. Often, they indicate that an analyst doesn't really understand a piece of policy, so he has related a clump of entities to each other. If this is the case, you'll find it particularly difficult to give the relationship a meaningful name. The best advice we can give you is always look for binary relationships first. If you cannot define your meaning using binary relationships, that is an indication you have a real *n*-ary relationship.

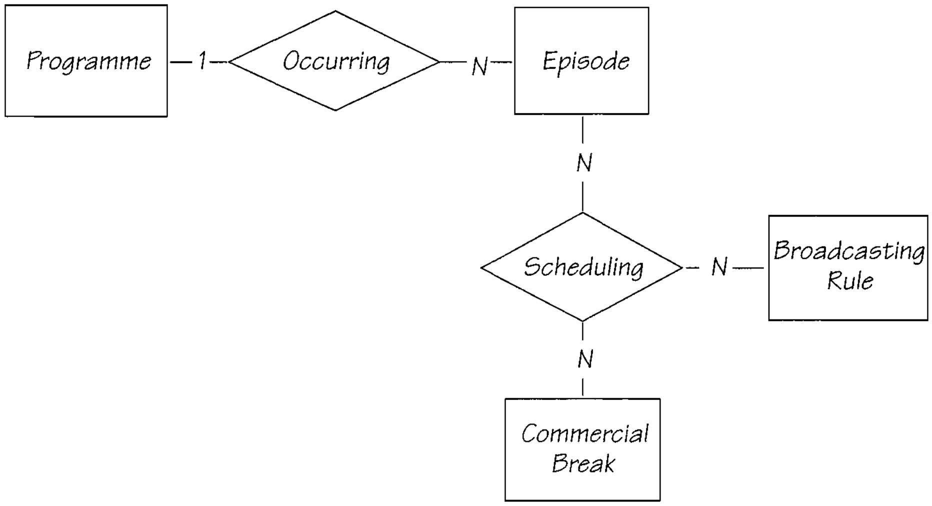

Figure 3.2.7: *The programming rules participate in the relationship between the breaks in and around an episode, and the occurrence of episodes of a programme.*

Bear in mind as you read through our explanation that your model does not have to agree entirely with ours. After all, you may have interpreted the statement differently and anticipated different answers from the users. The point of this discussion is to show you the data model that results from one interpretation of the business policy.

Let's go back to the context diagram in Figure 3.1.1 to see the details of what data enter the system and need to be remembered. Take the data flow COMMERCIAL COPY RECORDING, which comes from PRODUCTION COMPANIES. The system needs to remember the copy and where it came from. This gives two entities: COMMERCIAL COPY and PRODUCTION COMPANY, with a FILMING relationship between them. The COMMERCIAL COPY entity must also relate to ADVERTISING CAMPAIGN. Now consider this from "The Story of Piccadilly Television": "Some advertisers use several different commercials in a campaign, and the agency must send instructions on which copy is to be transmitted in each spot." The boundary data flow COPY TRANSMISSION INSTRUCTIONS delivers the data, which must be remembered. Digging a little deeper, we will discover that the reason for the copy instructions is to make sure that Piccadilly transmits each copy in a defined sequence. So Piccadilly saves the dates in the COPY TRANSMISSION INSTRUCTIONS as attributes of the COMMERCIAL COPY entity. The spot manipulators use the copy transmission dates when doing the ALLOCATING of COMMERCIAL COPY to COMMERCIAL SPOT.

There are two relationships, OCCUPYING and TRANSMITTING, between COMMERCIAL BREAK and COMMERCIAL SPOT. The reason is that OCCUPYING represents a temporary placing of the spot within the break. Remember that for some spot rates, there is no obligation to keep it in the break, nor indeed to transmit the spot. The TRANSMITTING relationship is established when the spot is broadcast. So this relationship represents an actual happening and is used for invoicing.

When you assemble all the fragments of the data model, you have a complete (to date) description of the business; the assembled model is in Figure 3.2.8. Compare it to your own, and make sure before going on that you can reconcile all of your (and our) decisions on how to portray the business policy.

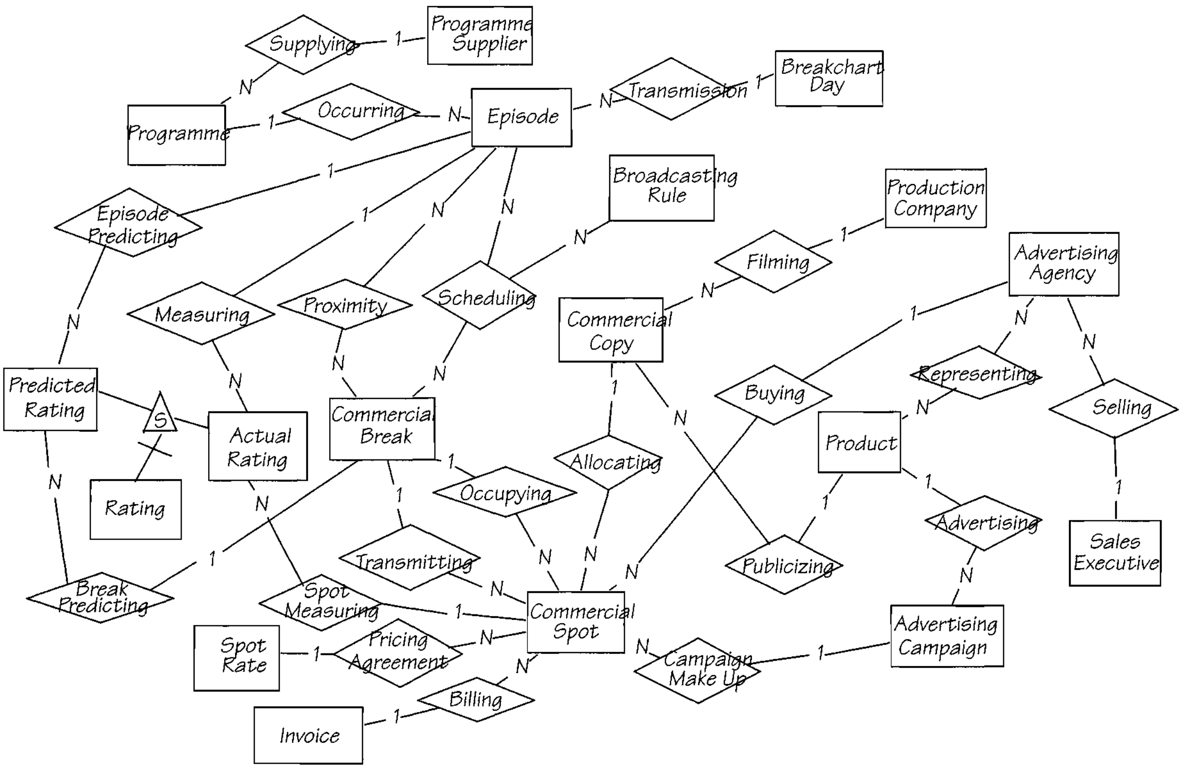

Figure 3.2.8: This first-cut data model serves as a guide for your detailed analysis. As you progress with the Piccadilly Project, you will define the data in each entity and relationship, as well as the processes that store and retrieve those data.

Defining Your Entities

We asked you to write down the attributes for the entity ADVERTISING CAMPAIGN. Some of the attributes are established by the executive when he uses (the data flow) CAMPAIGN REQUIREMENTS to plan the campaign. First consider what data make up the flow. When the agency gives Piccadilly the campaign requirements, the agency also must tell Piccadilly its name. So the name of the agency will be part of the flow, along with the name of the product to be advertised. The executive needs to know at least the budget for the campaign, the target audience, the ratings the agency wants to achieve, the length of the campaign, and, for scheduling purposes, the duration of each spot used in the campaign.

In the data dictionary, you can write the above description as

```
Campaign Requirements = * Data flow. An agency's description of require-
                        ments for an advertising campaign. *
                        Agency Name + Product Name + Campaign Budget
                        + Target Audience + Target Rating Percentage
                        + Campaign Duration
                        + {Required Spot Duration}
```

The braces around REQUIRED SPOT DURATION indicate there are many of them, whereas there is only one occurrence of each of the other data elements. (This notation is explained fully in Chapter 2.9 *Data Dictionary.* If you aren't comfortable with it, look ahead, but for now just accept it as a convenient shorthand.)

Which of the items in this data flow need to be part of the ADVERTISING CAMPAIGN entity? For the moment, we think this is a fair definition:

```
Advertising Campaign = Campaign Budget + Target Audience
                        + Target Rating Percentage + Campaign Duration
                        + {Required Spot Duration}
```

Note that each of the attributes describes an advertising campaign and only an advertising campaign. Other items from the data flow, such as AGENCY NAME and PRODUCT NAME, are not included as attributes of the entity ADVERTISING CAMPAIGN as they describe other entities. AGENCY NAME is an attribute of the entity ADVERTISING AGENCY, and PRODUCT NAME is, of course, an attribute of the entity PRODUCT. If you look at the data model, you'll see that ADVERTISING CAMPAIGN has a relationship called REPRESENTING with PRODUCT. The relationship links the product to its campaign, so that the system knows which campaigns belong to which products.

As the analysis progresses and you learn more about the system, add what you know to your models. You should write definitions for the entities and relationships as soon as you have something to say about them. At this stage, an examination of the boundary data flows reveals many data elements that you should attribute to your entities and relationships. It is also helpful to write a short statement explaining the purpose of each entity and relationship; this explanation is enclosed by asterisks.

Advertising Agency = ∗ Entity. Buyer of commercial spots for advertising campaigns. ∗
Agency Name + Agency Address + Agency Phone Number

Representing = ∗ Relationship. Keeps track of which agency is responsible for a product. Keeps track of historical relationships between agencies and products for collecting bad debts. Cardinality: for each Advertising Agency, there are many Products; for each Product, there are many Advertising Agency(s). Participation: Advertising Agency mandatory, Product mandatory. ∗
Representation Start Date + Representation End Date

Notice that the relationship definition specifies the purpose of the relationship, as well as the cardinality and participation rules for each entity involved in the relationship. The representing relationship also has two data elements attributed to it because both representation start date and representation end date truly describe the relationship. However, there will be many cases in which a relationship does not have any attributes.

The reason for writing these definitions is that they almost always raise questions. By asking these questions and getting answers, you'll learn still more about the system. Adding that knowledge to your models raises more questions, and ... Believe us, it does end eventually.

Look at the other entities in the data model and consider the attributes they each contain. Later, after reading the data dictionary chapters (2.9 *Data Dictionary* and 1.5 *Building the Data Dictionary*), you will define them completely.

Another Way to Build the Data Model

Having just put you through the exercise of building a data model from a statement of the users' business, we now confess that this is not the only way to build such a model. Our reason for doing it this way is to give you some practice with data modeling and to raise questions about Piccadilly. Shortly, you will partition the system by events and build event-response data models. (If you are unfamiliar with these terms and models, your trail will introduce them before you need them for the case study.)

Why do you need several methods? Because we have found in our own projects the enormous benefit of building a first-cut data model of the users' business. This model is an invaluable vehicle for starting to understand the business policy. However, the first-cut model is too large to support very detailed thinking. Later, you will use event-response data models to partition the data model into head-sized pieces so that you can confirm all the details and assumptions. The result will be a data model that is descriptive and reliable because you have used the policy in the event-response models to verify it.

✚ Ski Patrol

Your data model should be substantially the same as ours. Naturally, there are sure to be some differences in interpretation of the meaning of data and in your choice of names. This is expected when building a first-cut model. Your reason for building the model is to discover potential misunderstandings and arrive at a consensus. If, after reviewing the answer, you feel you've accomplished the objective of the exercise, proceed directly to the Trail Guide for further directions.

If you had some problems—if perhaps the whole exercise of building a data model had no meaning for you—then Chapter 2.4 *Data Viewpoint* will give you the reason for needing a data model. Similarly, if you feel it to be overkill to build both a data model and a data flow model of the same system, we remind you that the purpose is to analyze and specify the complete system. Your specification is more rigorous and easier to build if you have both data and process models. This will be reinforced when you get to the essential modeling chapters.

Some commonly encountered problems in data modeling concern the questions of which attributes make up an entity, when to make a relationship, and how to name relationships. Chapter 2.5 *Data Models* will answer these questions, and the exercises there will give you some more practice. Also, remember that your data model is only a first cut, based on your current fragmentary knowledge of Piccadilly. As you do more of the Project, your increased knowledge will add to and improve your data model.

Trail Guide

This Trail Guide duplicates the one at the end of Chapter 1.3 *What About the Business Data?*

● Easiest: Go to Chapter 2.6 *More on Data Flow Diagrams.* You will leave data models for the moment to expand your knowledge of data flow models. You will rejoin the Piccadilly Project shortly.

■ More Difficult: Now is the time to consider an appropriate viewpoint for this stage of the Piccadilly Project. Go to Chapter 2.8 *Current Physical Viewpoint.*

◆ Most Difficult: Continue on with the Piccadilly Project, when you will now get some more background into the company. Go to Chapter 1.4 *The Piccadilly Organization.*

❄ Promenade: This wasn't intended to be part of your journey. However, if you have already been through the data modeling chapters (2.4 *Data Viewpoint* and 2.5 *Data Models)*, look at the data model in Chapter 3.2 (Figure 3.2.8), then pick up your trail in Chapter 2.10 *Essential Viewpoint.*

REVIEW: THE PICCADILLY ORGANIZATION 3.3

Before You Reached Here ...
You have drawn Diagram 0 and updated your context and data models, as presented in Chapter 1.4 *The Piccadilly Organization.*

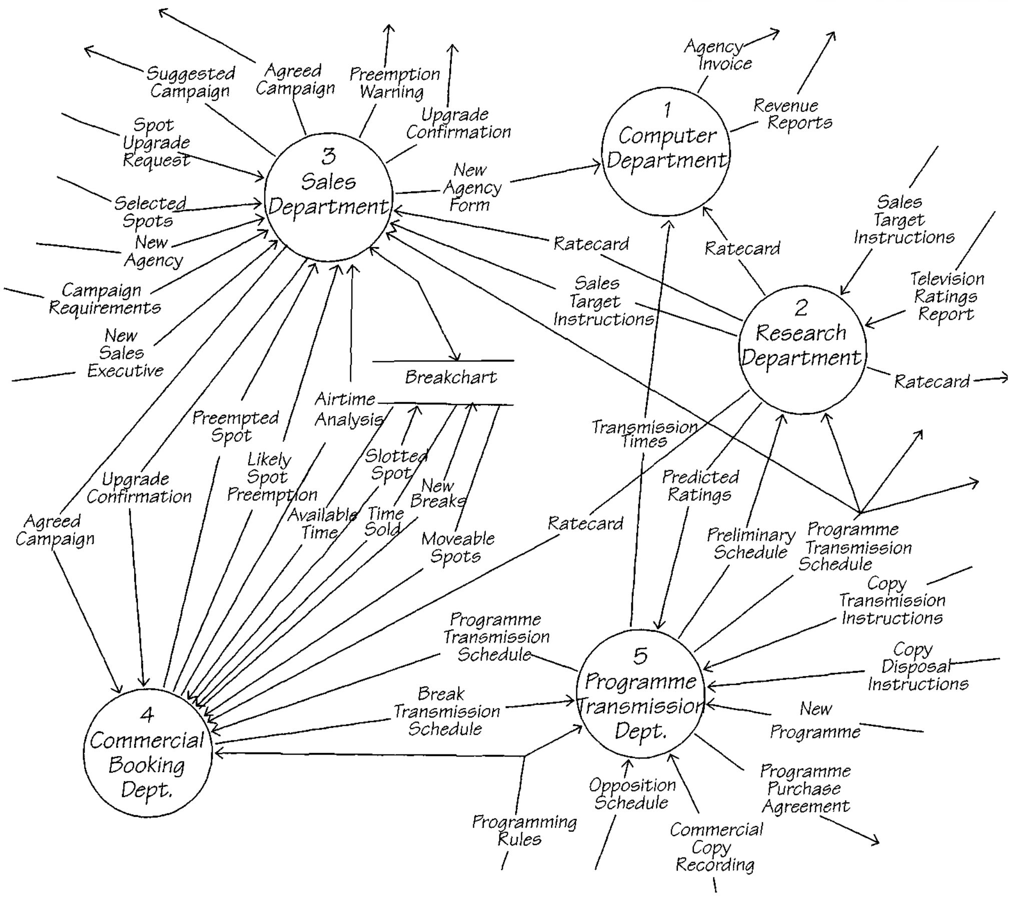

Figure 3.3.1: Diagram 0. This model is partitioned to highlight Piccadilly's current organization.

Sample Model of Piccadilly

The partitioning of this current physical model mirrors the business that you see when you visit Piccadilly's offices. The model is useful to you at the moment, as you do not yet have a complete understanding of the television business. The model also highlights some problems. The largest number of interfaces is between the Sales and Commercial Booking departments. This heavy traffic indicates that the work done by these two departments is closely connected. Perhaps there is some functionality being handled in the wrong department, or perhaps there is some overlapping or duplicate processing. It is too early yet to know. These data flows simply tell you that you must analyze both departments before you can see the complete picture.

High-level current physical models such as this one can be used to explain your project to other people. In the real Piccadilly Project, we pinned this model to a prominent wall and used it whenever we discussed the Project with the users or with others on the analysis team. We also found it an extremely good starting point for introducing new people into the Project.

Verifying Your Context

The model you have just built, Diagram 0, is a more detailed version of the context diagram. The functionality of the system is now decomposed into five lower-level bubbles. These five bubbles must interface with each other, which means you must introduce new data flows into the model. Note that the boundary data flows from the context diagram must also appear in the lower-level diagram.

Compare the data flows in your context diagram with the flows that enter or leave your Diagram 0. They should match, with the exception of any new flows that were discovered through the lower-level description. The flows NEW AGENCY, NEW SALES EXECUTIVE, OPPOSITION SCHEDULE, and COPY DISPOSAL INSTRUCTIONS do not appear in the context diagram (Figure 3.1.1). The business description mentions OTHER TV CHANNELS and PERSONNEL DEPARTMENT. Since you have no way of changing other television companies, they are outside your context; show them as a terminator. In a real-world project, you'd have to get a ruling from user management whether the Personnel Department is inside or outside the context before attempting to correct your models. In this Project, let's say that Piccadilly management has decided that the Personnel Department is outside the context. So we'll correct the context by adding PERSONNEL DEPARTMENT as a terminator, along with the other new terminator and the new flows. The updated context diagram appears in Figure 3.3.2.

As you learn about the details of a system, you will usually find that some information has been omitted from the higher-level models. That's because people forget to tell you something, or they assume that you already know something, or, as happens occasionally, they just don't know enough to give you the complete picture. People also tend to contradict one another.

These inconsistencies are irritating, but not debilitating. The reason that you are modeling the system is to come up with a precise definition, and it's far easier and cheaper to find the inconsistencies during analysis than at some later stage.

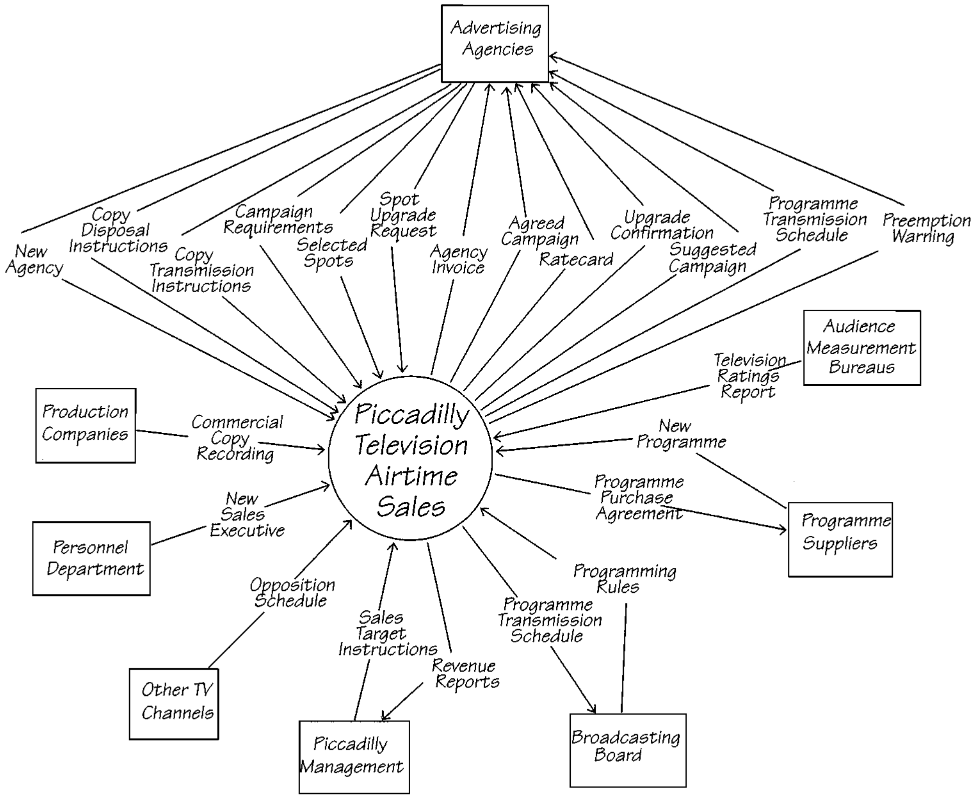

Figure 3.3.2: Updated context diagram. The data flows NEW AGENCY, NEW SALES EXECUTIVE, OPPOSITION SCHEDULE, *and* COPY DISPOSAL INSTRUCTIONS, *plus the terminators* PERSONNEL DEPARTMENT *and* OTHER TV CHANNELS *are added to the context diagram to make it balance with Diagram 0.*

Adding to Your Data Model

From the business description, you know that Perry Vale in the Programme Transmission Department needs to record the opposition's programme schedules. He does this so that he can schedule high-rating programmes whenever the opposition is planning a blockbuster. He cannot afford to let Piccadilly give away too big a share of the audience.

Therefore, the system has to remember the opposition's programming schedules, which means that the data model also changes. Figure 3.3.3 is an updated version of the first-cut data model we showed you in Figure 3.2.8.

Some Analysts' Questions

Suppose I didn't have a written description of Piccadilly. How would I model the organization?

There are other possible sources of information. As the users describe their work and workplace to you in person, draw data flow diagrams as they talk. They can help if you get stuck. Another source of information is reports and documents used within the company. Start by drawing a bubble for each department in the organization chart, and then treat each report or document as a data flow. Some of these will flow between departments, and some will be boundary data flows to or from terminators. Then, complete the model by asking the users, and looking for data flows that travel by telephone or voice.

Will the Piccadilly users think my model is too complicated?

The users won't find this picture nearly as complicated as you do. After all, it is a model of something they know very well: their own organization. The data flow names are terms they use every day. You can enhance the model by noting in each bubble the name of the manager or the key people in each department. Your objective is not to deny the complexity, but to control it.

One simplification that we have chosen to use in this model is the single unnamed data flow between the Sales Department and the breakchart. As we are not yet certain of the exact data content in this case, we use this flow as a summary of all the flows. Later, when we study the Sales Department in detail, we will identify the individual flows in the lower-level model. We were much more certain about the communication between the Commercial Booking Department and the breakchart. In this case, we felt confident enough to show separately named flows in the model, and these will be confirmed by our detailed study of the department.

When reviewing your models with the users, don't try to explain what the symbols mean; they'll either pick up the meaning from the context or ignore it. Instead, talk about the business you are modeling. Use your models to focus on the topics and areas you need clarified. When you point to a data flow, don't call it a data flow, call it by its business name. Talk through the model and make it come alive.

For instance, if you were talking to Dagenham Heathway in the Research Department, you could point to the data flow SALES TARGET INSTRUCTIONS and

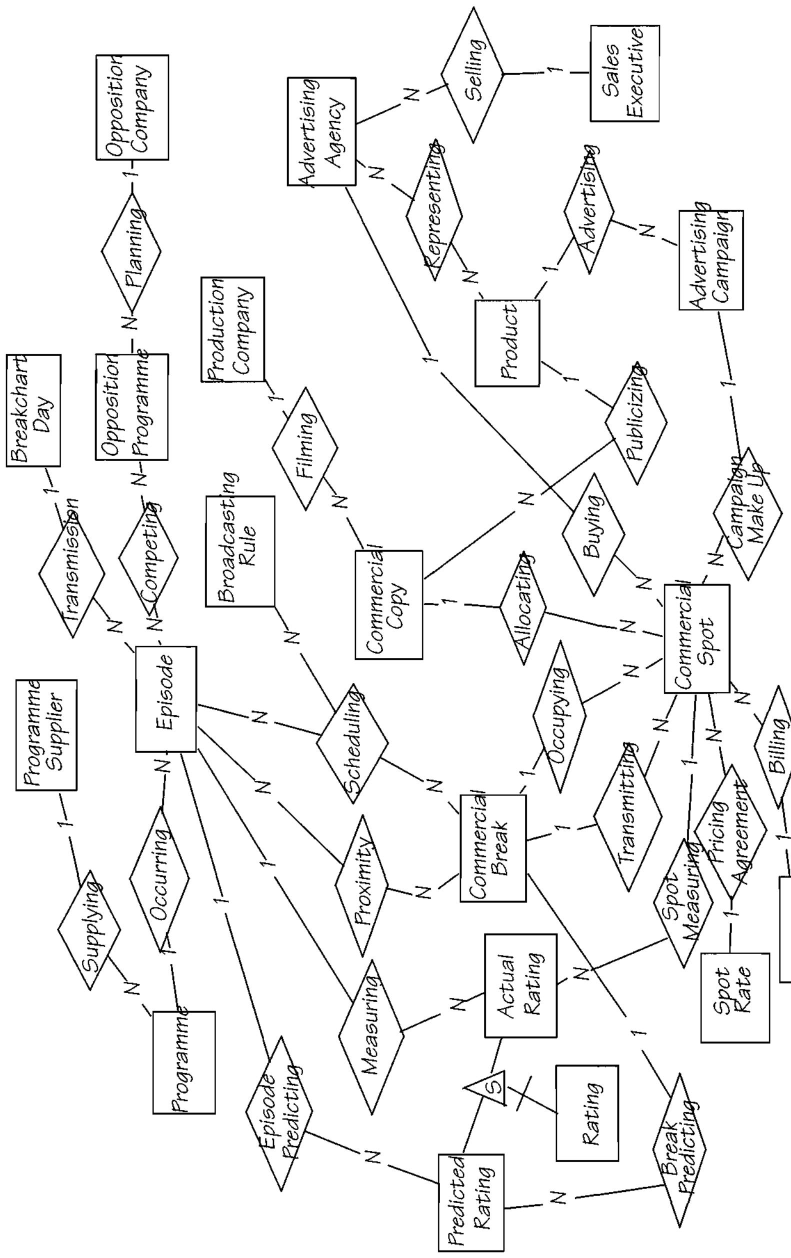

Figure 3.3.3: The first-cut data model is updated to reflect the additional information about the opposition companies and their programming. There are two new entities: OPPOSITION COMPANY *and* OPPOSITION PROGRAMME. *A new relationship keeps track of which company is* PLANNING *to schedule which opposition programme. Another new relationship keeps track of all the opposition programmes that are* COMPETING *with a Piccadilly programme.*

say, "When you get the sales target instructions from management, you use the programme transmission schedule and the television ratings report to do some financial modeling and to set the ratecard for the next quarter. Have I understood that correctly? Are there any other data you use that I don't know about?" Talking through the diagram leads Dagenham to verify your understanding and tell you about anything that is missing. If he corrects your model, thank him. Remember that it is not an interruption to your work if somebody makes changes to your model—it is the purpose of your work.

How long does it take to build a current physical model?
Sometimes, you can sketch a current physical model on a whiteboard in less than half an hour. In other cases, it can take weeks. We have found the time depends on three factors:

- how well you know the people involved
- how much user knowledge you have yourself
- how available are the people you need to see

Remember that the current physical model is not the specification for your future system. This model's purpose is to give you an understanding of the business. Instead of asking "How long does it take?" perhaps a better question is "How much time should I spend?" The answer is as much time as you need to understand the business, and no more. Later, after discussing event modeling, we can define more precisely how much knowledge you'll need. Also, as you progress through the Piccadilly Project, you will see the point at which we suggest that you stop physical modeling and move on to another view.

Are there any dangers in building current physical views of a system?
Yes. Many projects have wasted a great deal of time by building overly detailed current physical models. Some projects have built models that specify every detail right down to the mini specifications. That is not the purpose of the current physical model. Shortly before it was canceled, one project we know of had produced three thousand mini specifications detailing the current system. This was tragic, because the analysts could have captured enough knowledge with one or two levels of data flow diagrams, some data dictionary definitions, and *no* mini specifications.

The reason this happened was that the people building the models were not sure why they were doing it. Without a clear plan of how much physical modeling is needed, analysts find it is all too easy to continue doing it past the point of being useful.

Physical modeling is also fun to do. Most analysts enjoy the experience of talking to users, modeling what they said, verifying the model, and moving down to the next level of detail. There is also the problem that when analysts are talking to the users, they are providing details of their business. Many analysts feel that they have to model all the details so that they don't lose any of them and have to ask for the information again. However, at this stage of the Piccadilly Project, you don't know what details are needed, and what will become superseded by the future system. We urge you to procrastinate. You can always postpone details and then go back to pick up the details when you know what you need. It is much more difficult to get rid of lovingly crafted models even after they have proved to be unnecessary.

Your goal is to specify a system, and to do that, you have to know how to *selectively* build your physical models.

Why do you recommend starting the Piccadilly Project with a current physical view?
We recommend this approach because you had no prior knowledge of Piccadilly's business, and the current physical model helps you to understand its business. This model is a convenient vehicle to collect information about the system.

A current physical model is not wasted effort, and you've used it to accomplish two things. First, the model helps to ensure that you are beginning the project with the correct context. Second, it helps you get to know the key users. You've demonstrated your interest in what they told you by building a model of it. You've discovered most of the data that flows around and is stored by the company. You also know, at a high level, what processes exist to treat those data. While it is early yet, you have already traveled quite a distance.

Once you have built up enough knowledge of Piccadilly's business, it will be time to move on to other views of the system.

✚ Ski Patrol

We have mentioned that a possible pitfall with the current physical model is the temptation to show a bubble for each process mentioned in the text, thus resulting in a model with far too many bubbles. If this happened to you, a quick revisit to Chapter 2.7 *Leveled Data Flow Diagrams* would be useful for you to redo your model.

The data flow names should be substantially the same as those used in the text. You are free to change names, but you should always have a solid reason for doing so. Remember that you have to confirm this diagram with the users before proceeding, so they must be able to recognize the name.

Your diagram should have balanced with the context. If it didn't, or if you didn't do the balancing check, or if you didn't add the new boundary flows to your context diagram, visit Chapter 2.7 *Leveled Data Flow Diagrams* to read about keeping the leveled models in balance with each other. Balancing is a critical part of your modeling effort.

You also should have discovered the additions to the data model. You should be happy with your data model at this stage as a reflection of your growing knowledge of the business. It does not have to look exactly like ours. In later modeling efforts in the Project, you will refine your model, and at the same time prove or disprove your ideas. However, we want you to feel comfortable with the idea and notation of data models before proceeding. Look in Chapters 2.4 *Data Viewpoint* and 2.5 *Data Models* for help on data models.

Trail Guide

● Easiest: The next step in the Piccadilly Project is to define the data flows and stores. To find out how to do this, go to Chapter 2.9 *Data Dictionary*.

■ More Difficult: If you already know about data dictionaries, proceed to Chapter 1.5 *Building the Data Dictionary*. If you need to brush up on some rusty skills, may we suggest a quick detour through Chapter 2.9 *Data Dictionary*.

◆ Most Difficult: Put your skill with data dictionaries into practice by going to Chapter 1.5 *Building the Data Dictionary*.

❄ Promenade: This chapter is not required reading for you. Turn to Chapter 2.8 *Current Physical Viewpoint*, where you can pick up your trail.

REVIEW: BUILDING THE DATA DICTIONARY 3.4

Before You Reached Here ...
You have written some data dictionary definitions and probably updated your data model, as assigned in Chapter 1.5 *Building the Data Dictionary.*

Defining Piccadilly Entries

Your task was to define the information that Piccadilly receives from its programme suppliers. This is the minimum that Perry Vale needs to make his programme buying decisions. The entries are listed alphabetically.

```
Director Name = * Data element. Identifies the director of a programme. *

New Programme = * Data flow. Describes a programme offered by an English
                or overseas programme supplier. *
                Programme Name + Programme Type + Programme Description
                + Programme Duration + Programme Price
                + ?Programme Episodes
                + {Performer Name}
                + ?(Producer Name + Director Name)
                + ?Supplier Name

Performer Name = * Data element. Identifier of an actor or actress appear-
                ing in a programme. *

Producer Name = * Data element. Identifies the producer of a programme. *

Programme Description = * Data element. Synopsis of the contents of a pro-
                gramme. *
```

*Programme Duration = * Data element. Running time of a programme. Units: hrs/mins/secs. **

*Programme Episodes = * Data element. The number of episodes of a programme covered by an agreement with a supplier or by Piccadilly's internal production plans. **

*Programme Name = * Data element. The name that uniquely identifies this programme, for example, News at Ten, Brideshead Revisited, Coronation Street. **

*Programme Price = * Data element. Price paid to the supplier of a programme. Units: pounds sterling. **

*Programme Type = * Data element **
[First-Run Film | Sporting Event | Documentary | Talk Show | Old Movie]
** The types of programmes that Piccadilly transmits. Note that there are other programme types to add to these. **

*Supplier Name = * Data element. Identification for a programme supplier. **

Although not specifically mentioned, a supplier would likely tell Piccadilly who he is when the new programme information is sent, since Perry must inform the supplier of his buying decision. Because the information was not specifically mentioned by the user, a question mark is used to indicate that an assumption is being made. A question mark can be used anywhere you're not certain of what you are writing. Definitions and data flows highlighted by this notation can then be clarified by the users. There is nothing wrong with showing what you don't know, but there is everything wrong with hiding it.

The supplier's address is not listed, as it is likely to be on file and can be retrieved using the name. Perry can confirm this when you go back with questions.

You were given the information "Some programmes include the names of the producer and director." When names appear in the data flow, does it mean that both the producer's and the director's names are there? Or does it mean that one or the other may be present? The English language doesn't have any precedence rules for "and," but the data dictionary does. You could have defined (PRODUCER NAME + DIRECTOR NAME) to mean that if any names appear, then both do; or (PRODUC-

ER NAME) + (DIRECTOR NAME) to say that one or both or none may be present. Alternatively, if it is possible to have more than one producer and more than one director, then {PRODUCER NAME} + {DIRECTOR NAME} would be used. Again, a question mark is used in the definition as you cannot be sure about the correct meaning until you have checked with Perry Vale.

Although the user did not mention it, the analyst realized that there might be several episodes of a programme. The ?PROGRAMME EPISODES has been included in the definition so that the analyst remembers to ask the question.

In the description, you were told that there are other types of programmes. The comment in the data dictionary entry for PROGRAMME TYPE reminds you that some more values need to be defined.

Defining Ratecard

This definition was derived from a sample page of a ratecard presented in Figure 1.5.2.

```
Ratecard = * Data flow.  Prices, moveability, and preemption rules of time
                available for sale. *
                Rate From Date
                + ?{Rate Spot Duration
                + {Rate Segment Day
                + {Rate Segment Start + Rate Segment End
                + {Rate Moveability + Spot Price}}}}

Rate From Date = * Data element.  The commencement date for a new rate-
                card period.  Format: Day/Month/Year. *

Rate Moveability = * Data element. Rate moveability as defined in the rate-
                card. *
                [ Fixed: fixed on a nominated day and break | Broad:  moveable
                within a specified segment on a nominated day | ROD: run-of-
                day, moveable to any similarly priced segment on a nominated
                day | ROW: run-of-week, moveable to any similarly priced seg-
                ment during a week ]

Rate Segment Day = * Data element.  Day(s) of week on which this segment
                of time occurs. *
                [ Weekday | Saturday | Sunday ]
```

Rate Segment End = ∗ Data element. The end time for a ratecard segment. ∗

Rate Segment Start = ∗ Data element. The start time for a ratecard segment. ∗

Rate Spot Duration = ∗ Data element. Duration of spots as defined in the ratecard. Units: seconds. ∗
[10 | 20 | 30 | 40 | 50 | 60]

Spot Price = ∗ Data element. Ratecard price for a spot rate. Units: pounds sterling. ∗

Your names may be different from those that we have used, but their meanings should be substantially the same. Take a moment to reconcile your names with ours. Also, make sure that you understand ours because, naturally enough, we will be using them for the rest of the Project.

Our definition of RATECARD has a ? before RATE SPOT DURATION. We have assumed that when a new ratecard is issued for a quarter, the rates for all durations are changed. So for one RATE FROM DATE, there are a number of RATE SPOT DURATIONs. The brace indicates that there are several RATE SPOT DURATIONs, and the question mark says that we are not certain.

Make sure that you have your pairs of braces in the right places. If you specify the repeating groups incorrectly, it will mislead the file designers later when they receive your specification.

The ratecard is made up of a number of components; these are defined in turn. When the component is a data element, it has the comment ∗ *Data element* ∗. For example, RATE SPOT DURATION is a data element, and its definition shows all the possible values it can have. RATE FROM DATE is another data element, but has an almost infinite number of possible values. There is no point attempting to list them in the dictionary, and dates are commonly understood. So the definition simply has a comment, and points out that the format is day/month/year.

It is not possible to define all possible values of SPOT PRICE by looking at the sample ratecard page in Figure 1.5.2. There are other pages of the ratecard, for spots of different durations. You were told there are similar ratecard pages for 10-, 20-, 40-, 50-, and 60-second spots. Besides, over the life of the system, there will be an almost infinite number of values for this item. To find more information about this data element, you would look through the other pages of the ratecard. You'll find that the cheapest price is £150 for a 10-second ROW spot in the last segment of the day. The highest-priced spot, a 60-second fixed in the prime segment, sells for £25,000. You could enhance your definition in the data dictionary with

*Spot Price = * Data element. Ratecard price for a spot rate. Units: pounds sterling. Value range: >= 150 and <= 25000. **

Adding to Your Data Model

You were also asked to study your definitions of the data flows to see if they revealed any new or updated entities. Then you added any new information you found to the data model. Figure 3.4.1 is an updated version of the data model in Figure 3.3.3.

When you built your first-cut data model, you included an entity called PROGRAMME. At that stage, you didn't have enough information to write a detailed definition of it. However, you now know that most of the data content of PROGRAMME comes from the data flow NEW PROGRAMME. Now that you have defined the content of the flow, you can begin to define the entity PROGRAMME.

Let's say that the users have accepted our version of the definition for a new programme:

```
New Programme = * Data flow. Describes a programme offered by an English or
                  overseas programme supplier. *
                  Programme Name + Programme Type + Programme Description
                  + Programme Duration + Programme Price
                  + ?Programme Episodes
                  + {Performer Name}
                  + ?(Producer Name + Director Name)
                  + ?Supplier Name
```

The first five items are data elements, and, as they describe a programme and are remembered by the system, you can say they are attributes of the entity PROGRAMME. Entities are defined by listing their components just as if they were data flows (a sample appears below).

PROGRAMME EPISODES has a ? beside it in the definition of the data flow. The reason is because a question occurred to the analyst: "Do the suppliers tell Piccadilly if a new programme has multiple episodes?" If the answer to this question is yes, the question mark will be removed. Otherwise the element will be removed from the definition. Let us say that in this case the users agreed that PROGRAMME EPISODES should be part of the definition. So we can also attribute it to

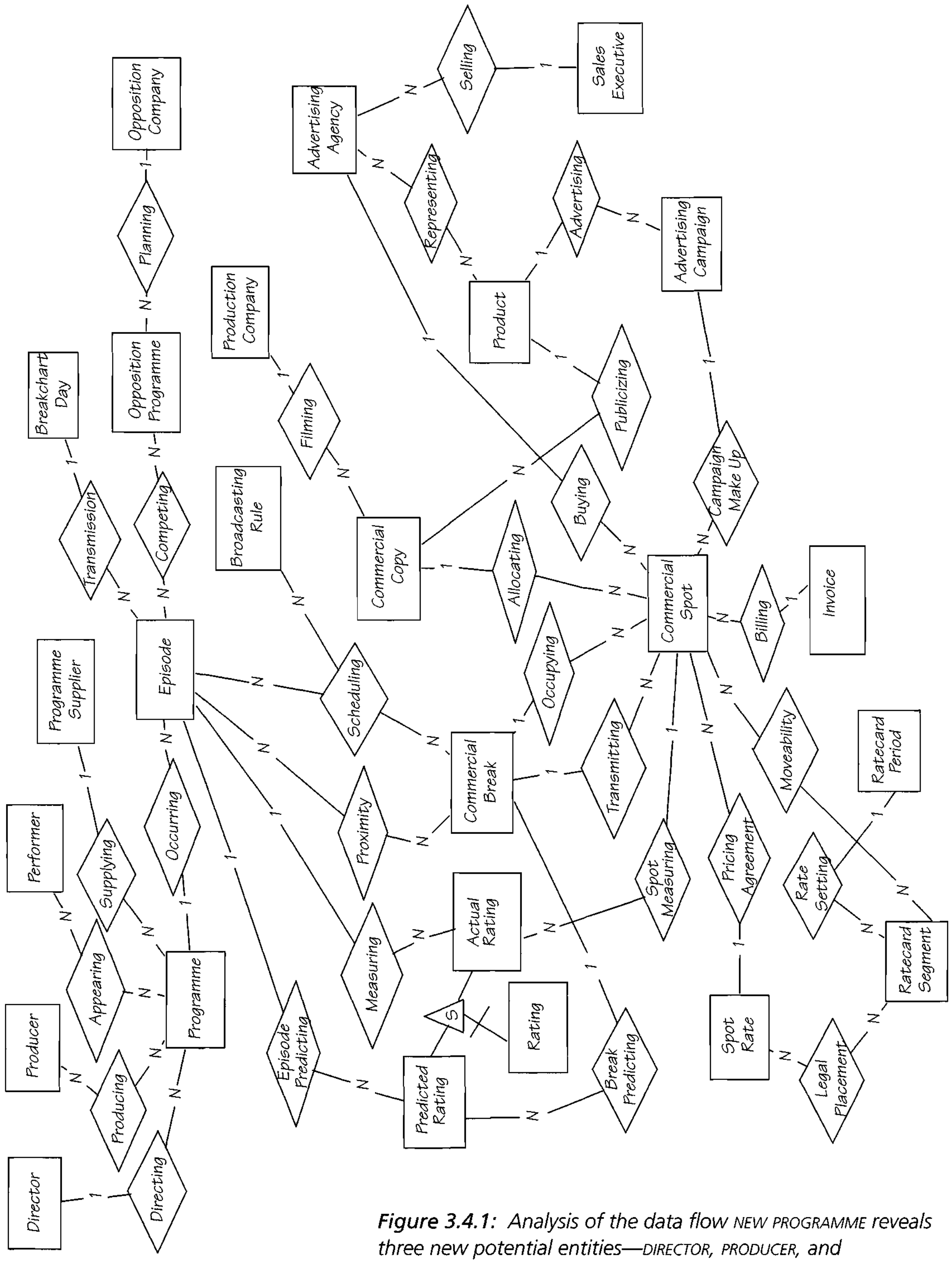

Figure 3.4.1: Analysis of the data flow NEW PROGRAMME *reveals three new potential entities—*DIRECTOR, PRODUCER, *and* PERFORMER*—and appropriate relationships. Analysis of* RATECARD *unveils* RATECARD PERIOD *and* RATECARD SEGMENT, *together with some new relationships.*

the entity PROGRAMME. Your dictionary can now support your data model with this definition:

```
Programme  = * Entity. A television programme made by Piccadilly or bought
               from a programme supplier. *
               Programme Name + Programme Type + Programme Description
               + Programme Duration + Programme Price
               + Programme Episodes + Programme Purchase Date
```

We added the * *Entity* * comment to differentiate between the types of definitions in the data dictionary. You will find this approach a useful aid in finding your way around a large data dictionary.

Did you include {PERFORMER NAME} in the definition of PROGRAMME? We hope not. While {PERFORMER NAME} lists the main performers in the programme, you cannot say that a performer's name describes a programme; it describes a performer. Instead of attributing this data item to PROGRAMME, it should be attributed to a new entity called PERFORMER. As usual, you support your data model with a data dictionary definition:

```
Performer = * Entity.  Actor or actress appearing in a programme. *
               Performer Name
```

Two other entities derived from the data in NEW PROGRAMME are DIRECTOR and PRODUCER. Adding these two entities to the data model, you define them:

```
Director = * Entity.  The director of a programme. *
               Director Name

Producer = * Entity.  The producer of a programme. *
               Producer Name
```

The NEW PROGRAMME data flow carries data about a number of different entities that are grouped in one data flow because there is a need for Piccadilly to know which director has directed a programme. When the data are stored as entities, there is a need to remember the relationship between them. The name DIRECTING given to the relationship in Figure 3.4.1 describes the reason for the link. Similarly, the data flow reveals a relationship PRODUCING between the PROGRAMME and its PRODUCER.

Think of these additions to the data model as potential entities and relationships. For example, we are not yet sure, within this context of study, whether Piccadilly thinks of DIRECTOR as an entity or whether the DIRECTOR NAME is an attribute of the PROGRAMME entity. We will verify this when we do a detailed analysis of the use of the data. Until then, you need to show all potential entities and relationships in the data model because by doing so, you make the questions obvious.

In the first version of the data model that you built (Figure 3.2.8), an entity called SPOT RATE is related to many commercial spots. At that stage of your analysis, Piccadilly's pricing looked quite straightforward. Since analyzing the sample ratecard, you now find that the pricing is more complex than it appeared when you did the first-cut data model. Your detailed study of the ratecard has resulted in this definition:

```
Ratecard = * Data flow.  Prices, moveability, and preemption rules of time
                available for sale. *
                Rate From Date
                + ?{Rate Spot Duration
                + {Rate Segment Day
                + {Rate Segment Start + Rate Segment End
                + {Rate Moveability + Spot Price}}}}
```

The data model should reflect the reality of the business. So instead of the single entity called SPOT RATE, you can break it down into separate entities, each one describing something that is familiar and important to the business:

```
Ratecard Period = * Entity. Period during which given rates apply. *
                Rate From Date

Ratecard Segment = * Entity. Continuous band of time defined on ratecard. *
                Rate Segment Day + Rate Segment Start + Rate Segment End

Spot Rate = * Entity *
                Rate Spot Duration + Spot Price + Rate Moveability
```

Relationships

Like entities, relationships must reflect the policy of the business. The LEGAL PLACEMENT relationship between RATECARD SEGMENT and SPOT RATE is there to define the segment placement offered by a particular set of moveability rules. This relationship is introduced because the price paid for a spot depends in part on the segments in which it can be transmitted. The relationship is many to many, as a rate can apply to several segments, while the one segment can have many rates applicable to it.

The MOVEABILITY relationship between COMMERCIAL SPOT and RATECARD SEGMENT is established when the agency agrees to buy a spot with certain moveability conditions. (Broad spots can be moved within a segment; run-of-day spots moved to similarly priced segments on the same day; and run-of-week spots moved to similarly priced segments over the week.) The relationship exists to link a commercial spot with the segments in which it may be broadcast.

Eventually, you'll verify each relationship and define each of these relationships in the data dictionary, and you can do this when you acquire more knowledge of the processes that create and use the relationships.

✚ Ski Patrol

Here's some first aid with the data dictionary definitions. At this stage, we want you to feel secure about defining data. We trust that the reason for writing definitions was revealed by the exercise, namely to better understand the data and the system that uses the data. If you think you can model a system without defining the data, please rethink. If you are not happy with the meaning of the operators, we can only suggest reviewing them in Chapter 2.9 *Data Dictionary.* If you were not happy with our interpretation of the data, take a few moments to reread the problem statement in Chapter 1.5 *Building the Data Dictionary.* This time, have the sample answers beside the problem statement. Note how the nouns that are important to Piccadilly's business make up the data dictionary definitions.

We suspect the data model was your biggest problem. The most important thing we have to say about the data model is, "Don't panic!" Data models don't come easily to most people, so you are not alone. Data modeling requires knowledge of the subject matter and the ability to view that subject at a high level of abstraction. The Piccadilly Project will give you plenty of practice in developing these skills, and you will progressively improve your data model. If you attempted to do the data modeling part of the exercise without reading and doing the exercises in Chapters 2.4 *Data Viewpoint* and 2.5 *Data Models,* we suggest a detour through those chapters.

So far, we have approached the data model somewhat piecemeal. We have attempted to build a data model from a general description of the business, and have made some small alterations to it by analyzing data flows. This approach could be classified as a "fuzzy top-down approach." Its strength lies in providing a way of getting started with a complex problem.

There is more to come: When we work through the chapters on essential event-response models, we'll look at another way to model the system's stored data. This approach works by partitioning the data model into small logical chunks based on the need to support one event (this is explained in Chapter 2.11 *Event-Response Models*). By the time you have experienced both approaches, we're sure that data modeling will be much clearer to you. And, as is necessary in practice, you will be in a position to blend both approaches.

Finish comparing your data dictionary and data model upgrades with the samples. Make sure that you can reconcile any differences between your answers and ours.

Trail Guide

● Easiest, ■ More Difficult, and ◆ Most Difficult: Go to Chapter 1.6 *Selling the Airtime.* There you will expand on your model of the Piccadilly system.

❄ Promenade: The most appropriate destination for you is to rejoin your trail in Chapter 2.8 *Current Physical Viewpoint.*

REVIEW: SELLING THE AIRTIME 3.5

Before You Reached Here ...
You've built the data flow diagrams in Chapter 1.6 *Selling the Airtime.*

The Data Flow Diagram As a Recording Device

During your early interviews with the users, the data flow diagram can serve as a note-taking device. As people talk, draw processes and data flows. Your diagrams won't be great art, but they nevertheless have a use. You capture what the users said, and immediately afterward confirm that you have the correct interpretation of what they said.

For example, the model in Figure 3.5.1 shows that bubble 3.4 SET SALES POLICY RATES updates the BREAKCHART with BREAK MINIMUM RATE. You show this model to Stamford Brook, sales manager, and ask him, "Is this right? Is it true that when you set the sales policy, you update the minimum rates on the breakchart?" If this isn't true, if he said something he didn't mean or you misunderstood what he said, now is the time to find out. You have no ego invested in this model (nobody can love a diagram with fourteen bubbles), and you cannot mind changing it. After all, its purpose is to provide you with an accurate and unambiguous statement of what happens in the Sales Department. Changes now will save time later on.

The model should be comprehensible to Stamford. If he were available, you would walk through the model with him and raise all the questions you could. Maybe he would disagree with your model, or contradict something he said, or remember something he forgot to tell you. What you would be doing is communicating with him in a language he could understand.

Don't worry if initially there are no data flows joining all the processes in your diagram; you'll find these when you go over the model with the users. Also don't be too concerned if your model breaks some of the rules discussed in this book. You can clean it up later.

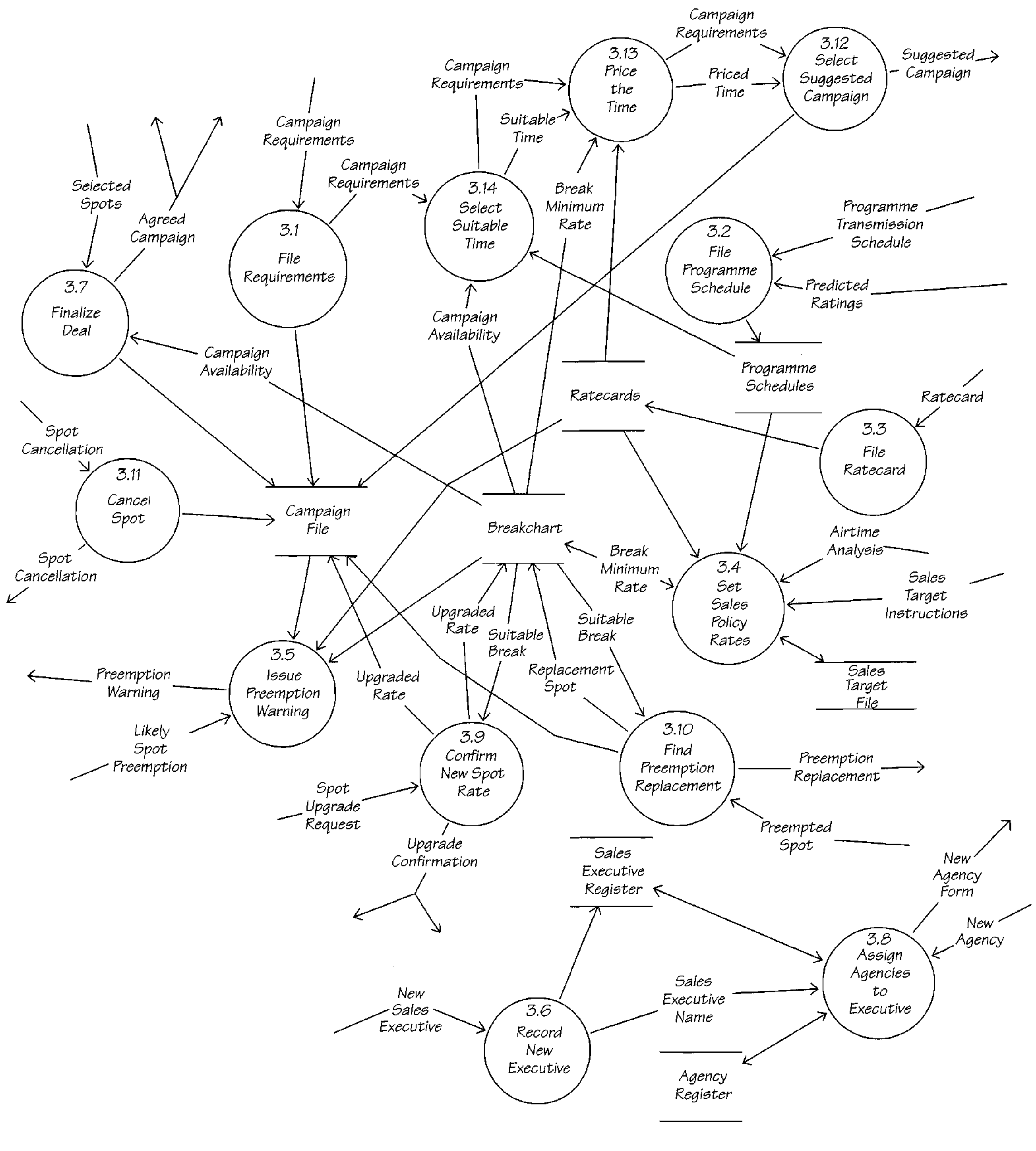

Figure 3.5.1: Preliminary version of Diagram 3 Sales Department, reflecting what Stamford told you. There are too many processes in this diagram, but this can be fixed later by leveling upward. At this stage, the model is used to confirm Stamford's description of the business.

Leveling Upward to Reduce Complexity

The first thing that strikes you about this model is that there are more bubbles in it than we recommend in Chapter 2.7 *Leveled Data Flow Diagrams.* When you use a data flow diagram as a note-taking device, this often happens. Sometimes, it results in models even more complex than this one. As an information-gathering tool, that's fine, but you cannot leave the model in this state forever.

To control the complexity and reduce the number of bubbles, group processes to create a high-level summary. Before making this higher level, though, you have to identify which processes can be logically grouped together.

There are two guidelines. The first is to group processes so that you minimize the number of interfaces at the highest level. This can be done by trial and error, but it is probably faster to look for groupings of bubbles that are closely connected to each other but loosely connected to anything else. The second guideline is to look for groups of bubbles with similar functionality so that the group can be given a functional name that is meaningful to the business being studied.

Figure 3.5.2 is the result of leveling upward. Bubbles 3.5, 3.9, 3.10, and 3.11 (from Figure 3.5.1) were functionally related and were consolidated into the honestly named process MAKE CHANGES TO SPOTS. We also grouped bubbles 3.1, 3.12, 3.13, and 3.14, which had data flow connections, into one high-level bubble called PLAN CAMPAIGN.

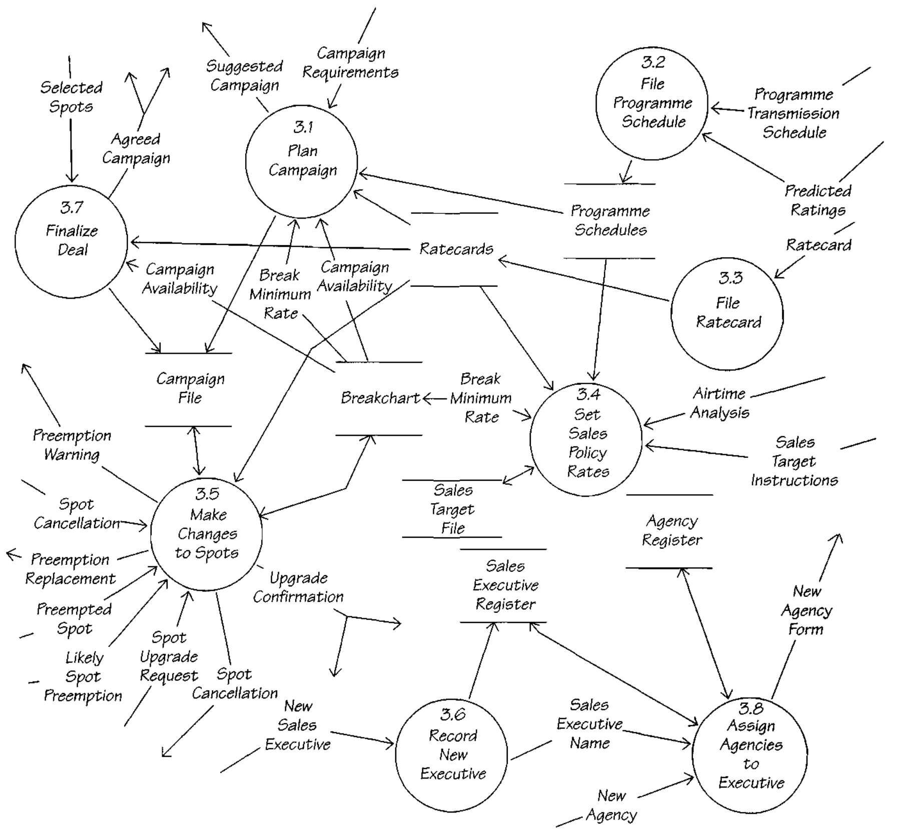

Figure 3.5.2: Second version of Diagram 3 Sales Department, combining some of the bubbles of the previous version. It was leveled upward to make the model less complex. Notice how many of the processes have interfaces through the use of common physical data stores.

The model shown in Figure 3.5.2 is more presentable. It shows all of the functionality of the Sales Department, but does it in only eight bubbles. This model would be used to present an overview of the department, or to reassure Stamford if he became apprehensive over the complexity of the previous version. The details of the department are not lost, and they are now shown in two lower-level diagrams 3.1 and 3.5. These appear in Figures 3.5.3 and 3.5.4, respectively.

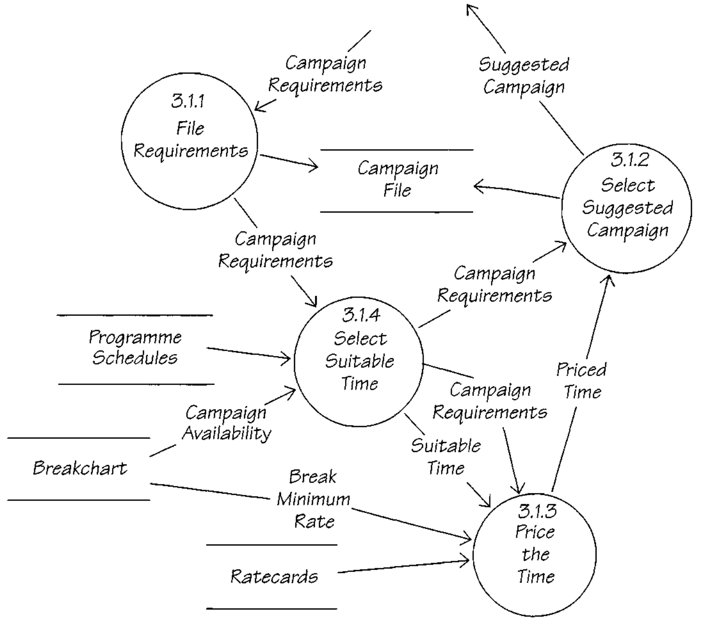

Figure 3.5.3: *Diagram 3.1 Plan Campaign.*

The Big Picture

Your work in the Sales Department has identified two new data flows: SPOT CANCELLATION, which comes from the agency; and PREEMPTION REPLACEMENT, which goes to the agency.

To keep your models in balance, you have to add these data flows to Diagram 0 (Figure 3.3.1) and the context diagram (Figure 3.3.2). Figures 3.5.5 and 3.5.6 are updated versions of these diagrams.

✚ Ski Patrol

A common error with lower-level data flow models is to omit the processes that store data. If a data flow that originates outside the context of your diagram is to be stored, there must be a process to store it. For example, some of Piccadilly's data

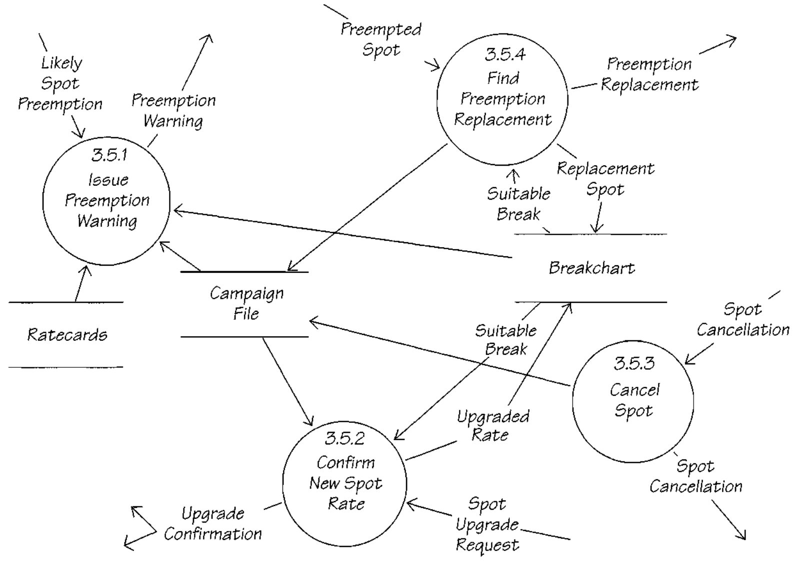

Figure 3.5.4: *Diagram 3.5 Make Changes to Spots. This diagram shows the processes that were summarized in the higher-level diagram.*

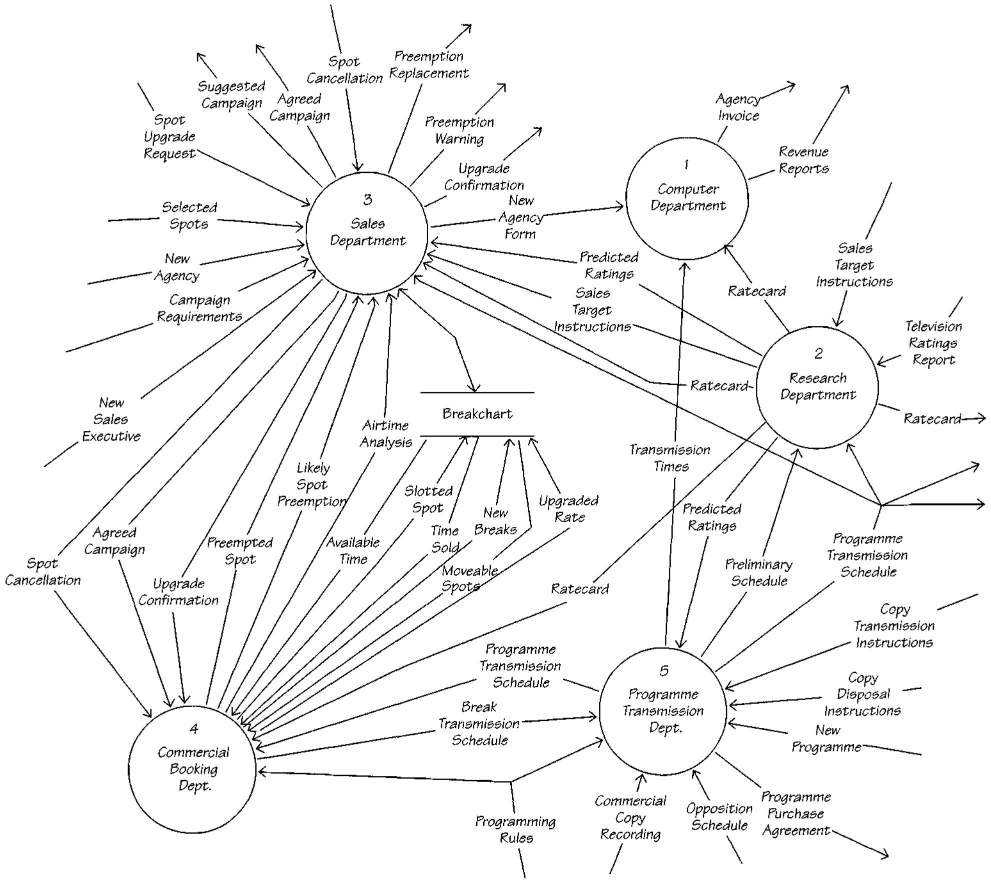

Figure 3.5.5: *Updated Diagram 0 contains the two new data flows identified during your work in the Sales Department.*

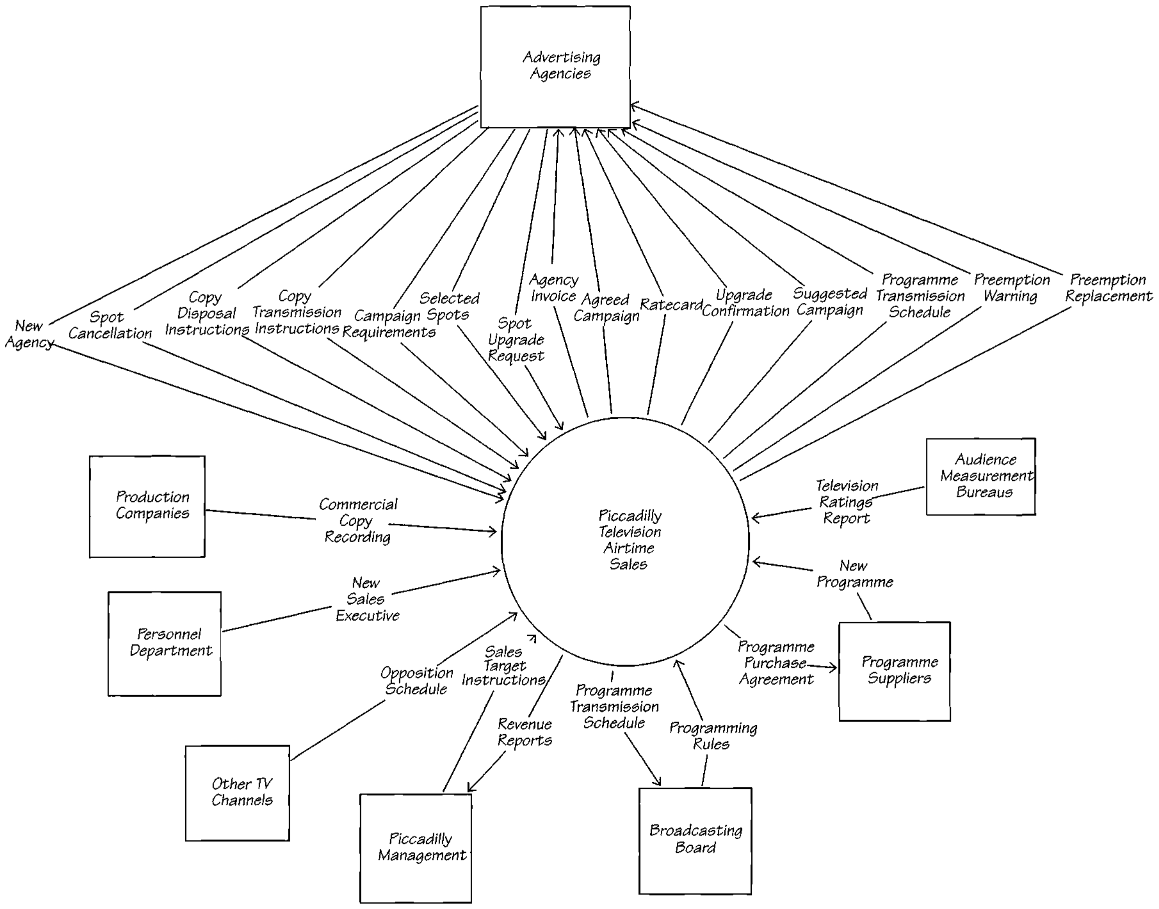

Figure 3.5.6: *Updated context diagram. The two additional data flows were both interfaces with terminators, so they must be shown in the context diagram.*

originate in the advertising agencies, but they have no mechanism to store data inside Piccadilly's system. Besides, it is not desirable for outsiders to be able to change data. Similarly, if the data originate in a bubble that is not part of your diagram, you still must have a process to store it. Data cannot come from out of nowhere and go directly into your data stores. You must control the storage.

It is not necessary that you have exactly the same diagrams as these samples. You may have partitioned the problem a different way. However, you should have one or more models that you would feel confident about talking through with Stamford Brook.

By this stage of the Project, we want you to feel comfortable with drawing leveled data flow diagrams, and maintaining the correct balance between levels. Remember that having balancing abnormalities does not always mean that the lower-level diagram is wrong. Sometimes, the parent diagram needs to be changed because it fails to show a flow or store that appears in the more detailed child diagram.

If you have any questions about the symbology used in data flow diagrams, review Chapters 2.2 *Data Flow Diagrams* and 2.6 *More on Data Flow Diagrams.* Chapter 2.7 *Leveled Data Flow Diagrams* provides reference material on how leveling works. If you are having trouble with why you want to build a model of the current system, try a detour through Chapter 2.8 *Current Physical Viewpoint.* We also want you to be confident with your ability to record data dictionary entries. A refresher on the notation can be found in Chapter 2.9 *Data Dictionary.*

What to Do

Just so you don't have to spend all your time building current physical models, we've built the lower-level models for the other Piccadilly departments. They are packaged in Chapter 3.6. You will be using the current physical model as the basis for your future work, so take some time now to find your way around it.

Trail Guide

You'll need to remember your next destination before leaving here because you are going to Chapter 3.6, the complete physical model for Piccadilly. You will visit Chapter 3.6 many times from many different locations, so we are not able to provide a trail guide out of there.

● Easiest, ■ More Difficult, and ◆ Most Difficult: Go to Chapter 1.7 *Strategy: Focusing on the Essentials.* Now you'll leave the physical model to move on to a more logical view of the system.

❄ Promenade: You may rejoin your trail in Chapter 2.8 *Current Physical Viewpoint,* where you will learn more about the change in direction the Project is about to take.

3.6 COMPLETE CURRENT PHYSICAL MODEL

Before You Reached Here ...

You could have been at any of a number of locations, since many of the Project chapters refer to the models here. Don't forget where you came from, as we have no way to direct you back there.

This chapter contains the complete current physical model for Piccadilly Television. Since you would spend too much of your time if you were to build the complete model, we have completed the parts that you've not done. The model is for reference purposes. Look through it so that you are familiar with its contents.

Contents

This complete current physical model is made up of a collection of leveled data flow diagrams, a data model, and a supporting data dictionary, specifically:

1. A package of data flow diagrams:
   - Context diagram (Piccadilly Project at the highest level)
   - Diagram 0 (Piccadilly's current implementation)
   - Diagram 1 Computer Department
   - Diagram 2 Research Department
   - Diagram 3 Sales Department, version 2
   - Diagram 3.1 Plan Campaign
   - Diagram 3.5 Make Changes to Spots
   - Diagram 4 Commercial Booking Department
   - Diagram 5 Programme Transmission Department
2. Updated version of the data model.
3. Data dictionary, with the definitions of all terms in the data flow diagrams and data model. The entries are listed in alphabetical order. Each entry has a comment that classifies its type as one of the following:

∗ Data element ∗
∗ Data element grouping ∗
∗ Data flow ∗
∗ Data store ∗
∗ Entity ∗
∗ Material store ∗
∗ Relationship ∗

Sometimes, the definition of an entry is followed by one or more comments that define the purpose, define the values, or point to other sources of knowledge about the entry.

✚ Ski Patrol

Your patroller cannot know why you are reading this model; you could have reached this chapter from a number of others, so the only help the ✚ Ski Patrol can give is to refer you to the appropriate chapter for each of the model's components:

- data flow diagrams discussed in Chapters 2.2 and 2.6
- leveled data flow diagrams in Chapter 2.7
- data dictionary described in Chapter 2.9
- data models in Chapter 2.5

Trail Guide

● ■ ◆ ❄ All trails: This chapter is used as reference material for many other chapters. As we have no idea where you came from, our advice is to return whence you came.

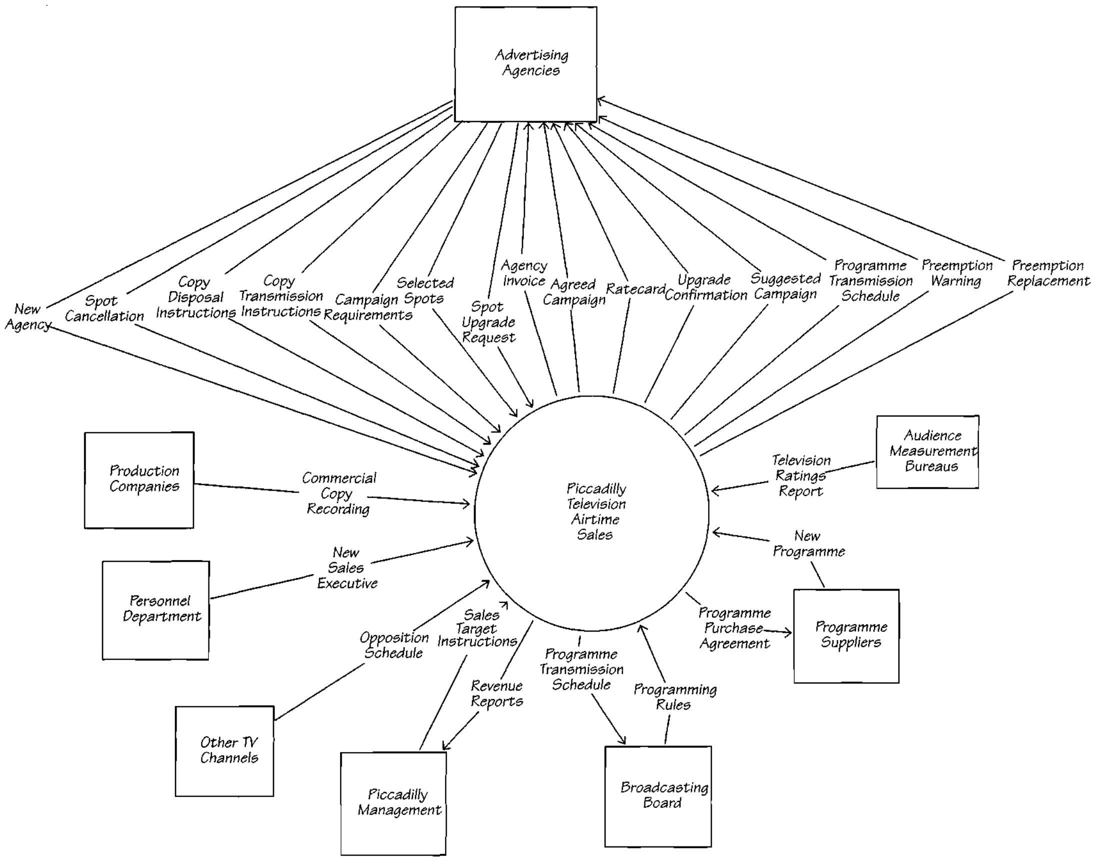

***Figure 3.6.1:** Context diagram, which defines the scope of the current Piccadilly system.*

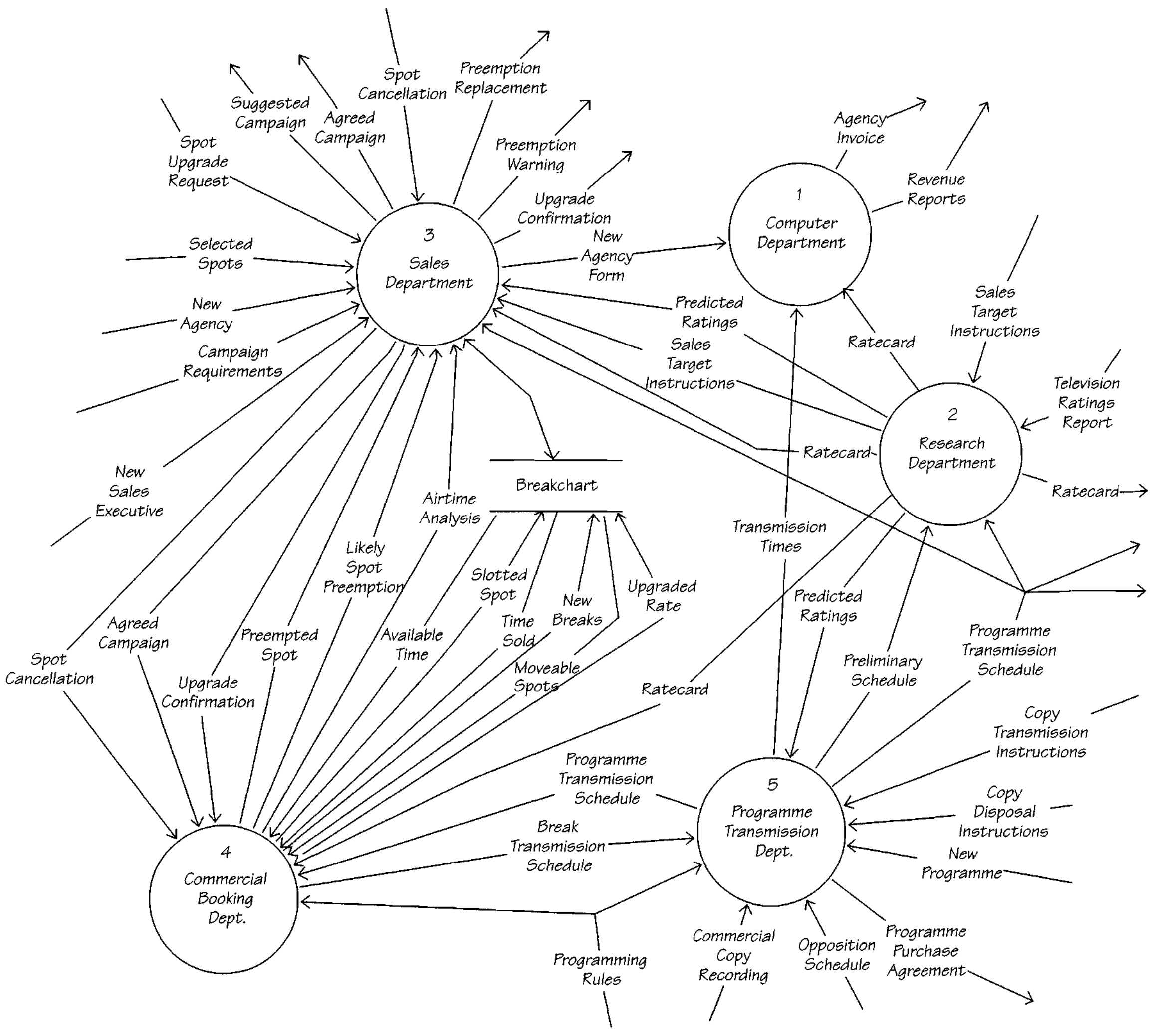

Figure 3.6.2: *Diagram 0, the first breakdown of the context diagram. This is intended to be a readily recognizable physical model, so the departments have been chosen as the partitioning theme.*

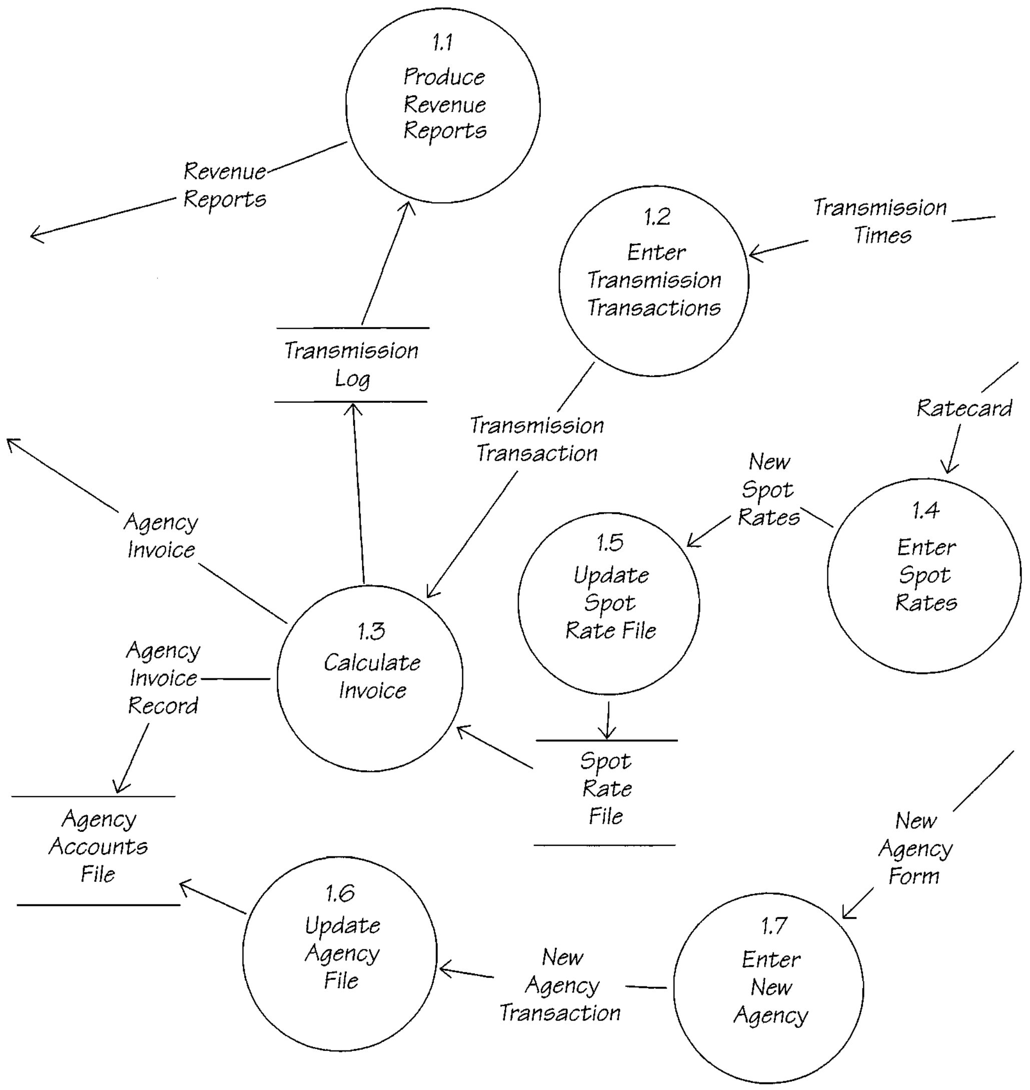

Figure 3.6.3: *Diagram 1 Computer Department.*

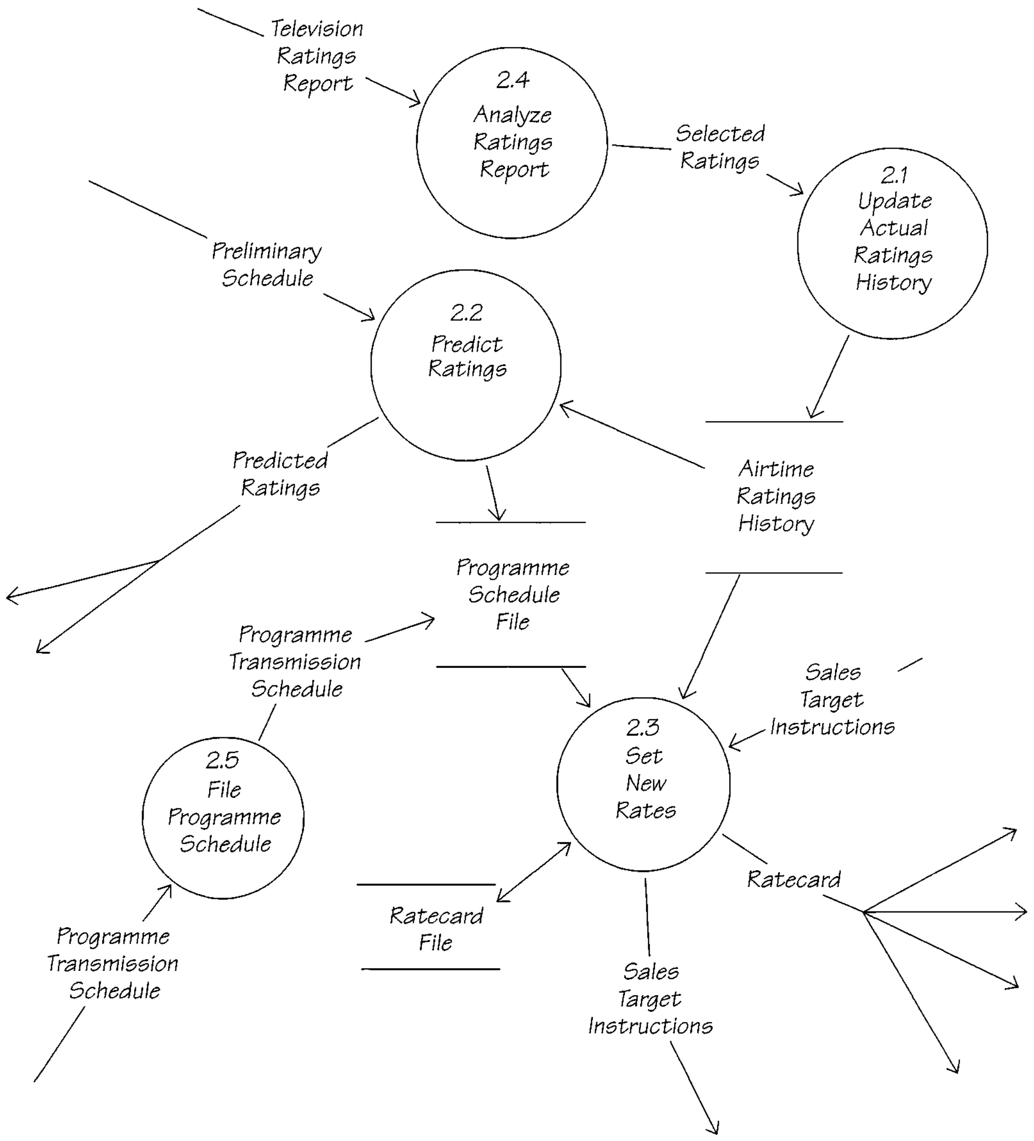
Television Ratings Report
2.4 Analyze Ratings Report
Selected Ratings
2.1 Update Actual Ratings History
Preliminary Schedule
2.2 Predict Ratings
Predicted Ratings
Airtime Ratings History
Programme Schedule File
Programme Transmission Schedule
Sales Target Instructions
2.3 Set New Rates
2.5 File Programme Schedule
Ratecard File
Ratecard
Programme Transmission Schedule
Sales Target Instructions

Figure 3.6.4: *Diagram 2 Research Department.*

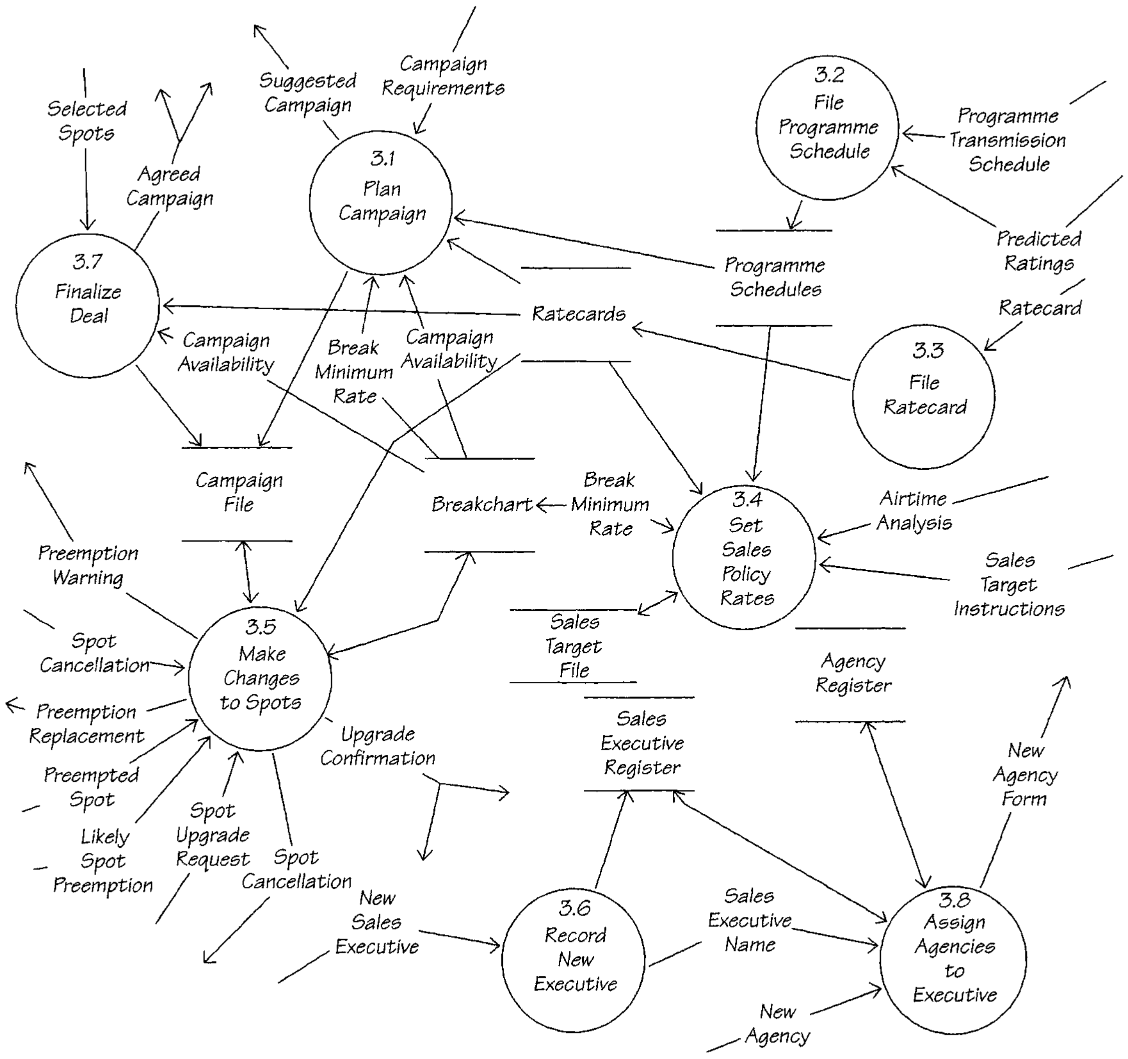

Figure 3.6.5: Diagram 3 Sales Department, version 2.

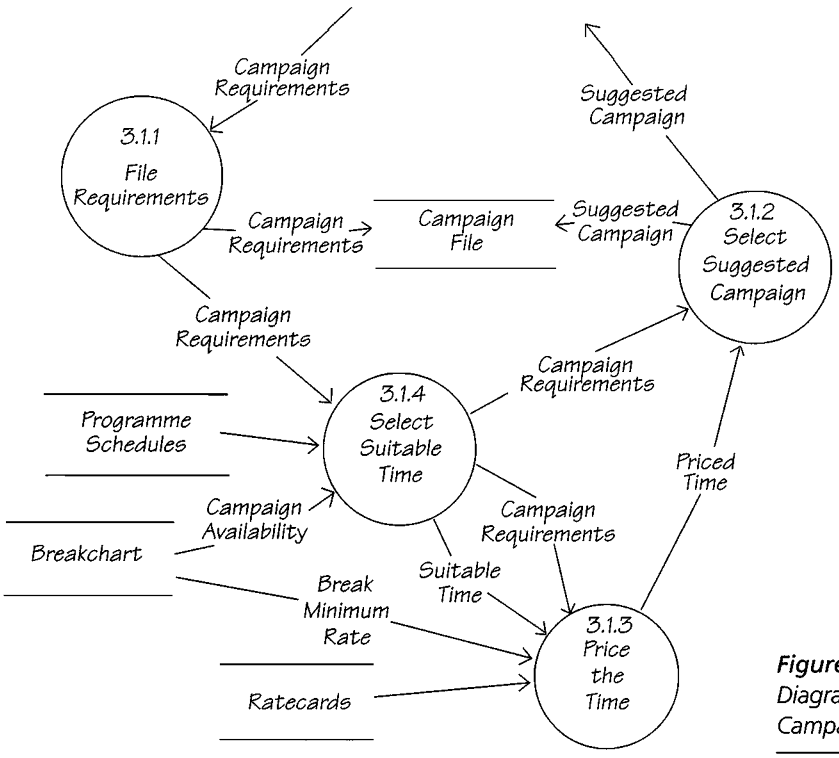

Figure 3.6.6: *Diagram 3.1 Plan Campaign.*

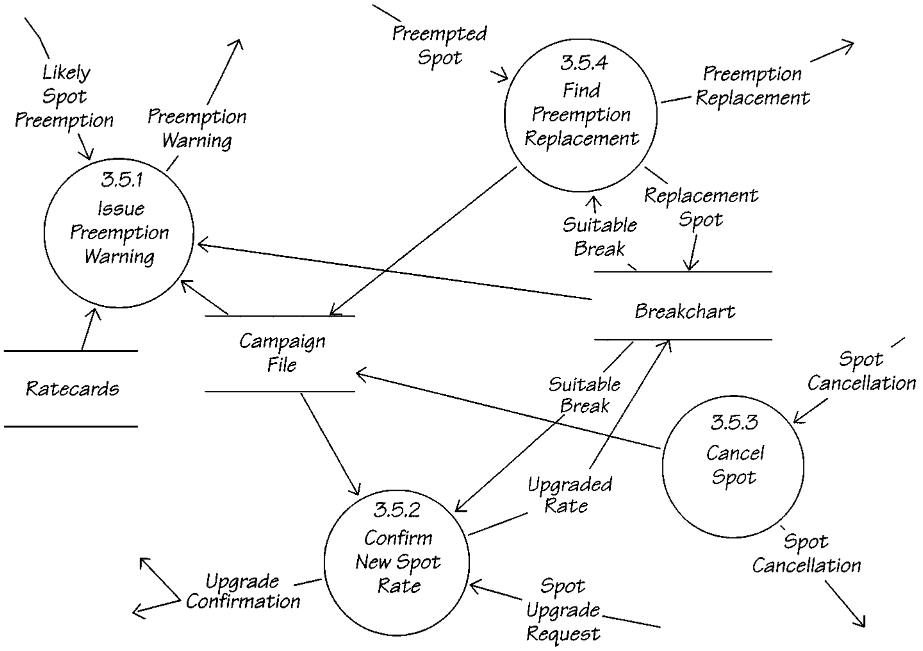

Figure 3.6.7: *Diagram 3.5 Make Changes to Spots.*

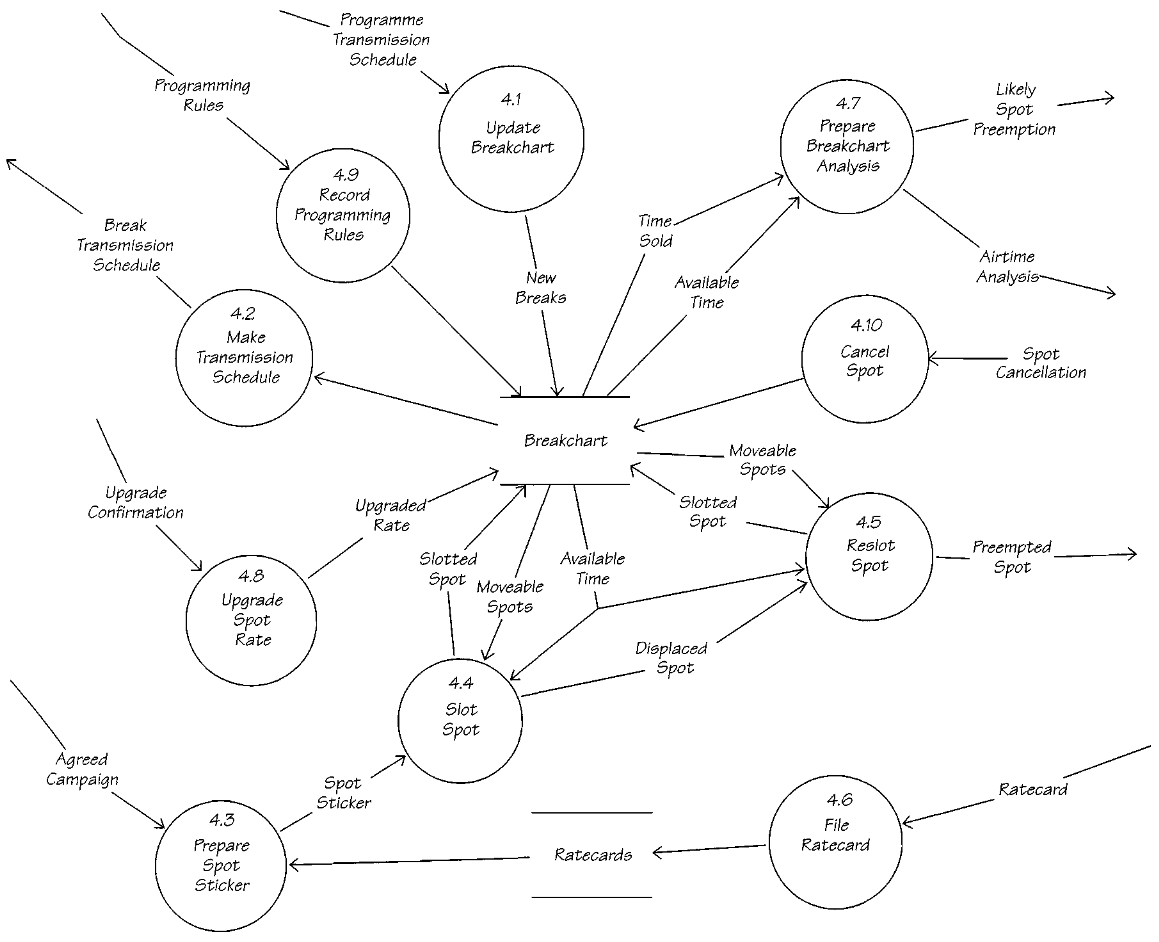

Figure 3.6.8: *Diagram 4 Commercial Booking Department.*

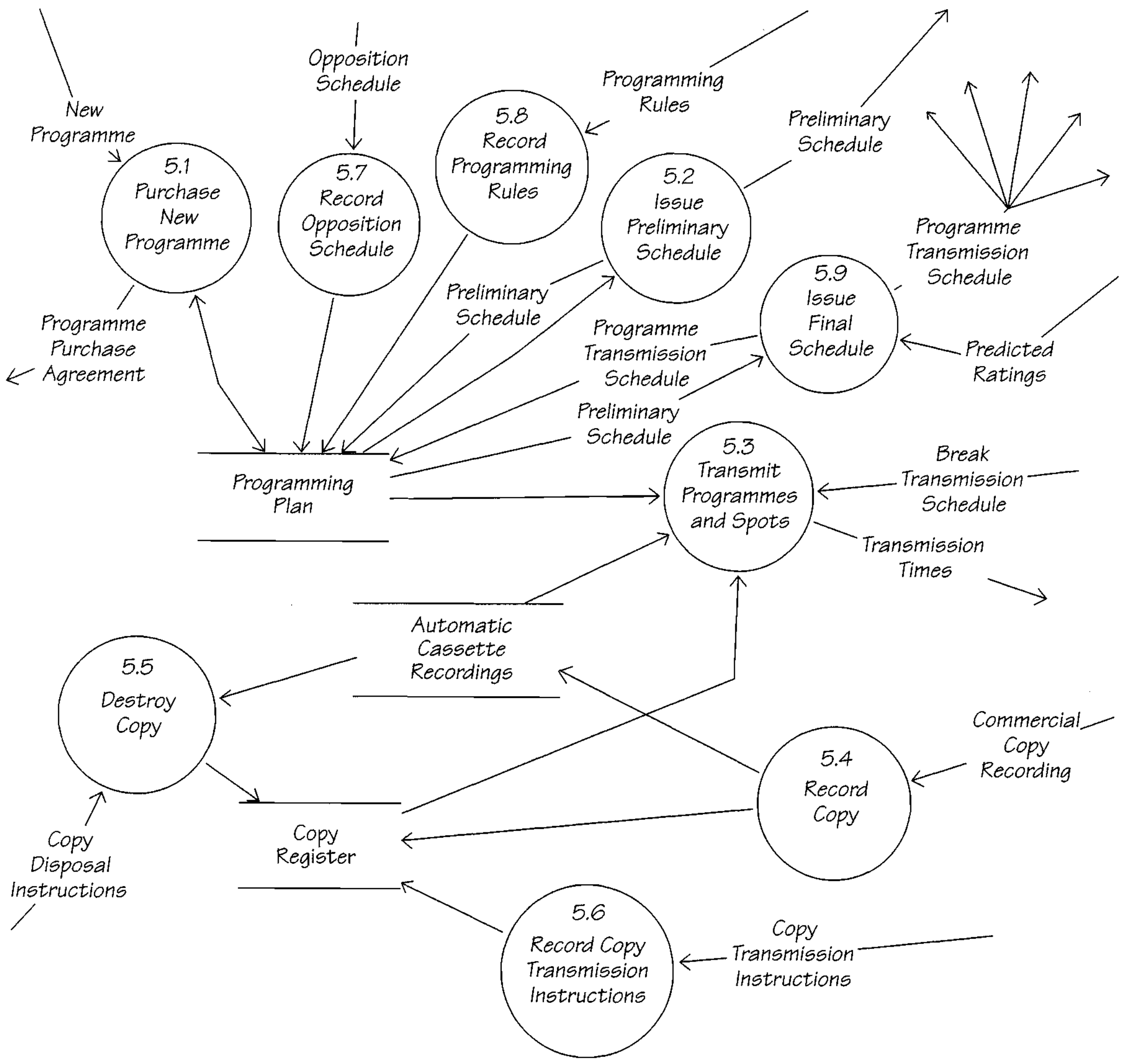
New Programme
5.1 Purchase New Programme
Programme Purchase Agreement
Opposition Schedule
5.7 Record Opposition Schedule
5.8 Record Programming Rules
Programming Rules
5.2 Issue Preliminary Schedule
Preliminary Schedule
Preliminary Schedule
Programme Transmission Schedule
5.9 Issue Final Schedule
Predicted Ratings
Programme Transmission Schedule
Preliminary Schedule
Programming Plan
5.3 Transmit Programmes and Spots
Break Transmission Schedule
Transmission Times
Automatic Cassette Recordings
5.5 Destroy Copy
Copy Disposal Instructions
Copy Register
5.4 Record Copy
Commercial Copy Recording
5.6 Record Copy Transmission Instructions
Copy Transmission Instructions

Figure 3.6.9: Diagram 5 Programme Transmission Department.

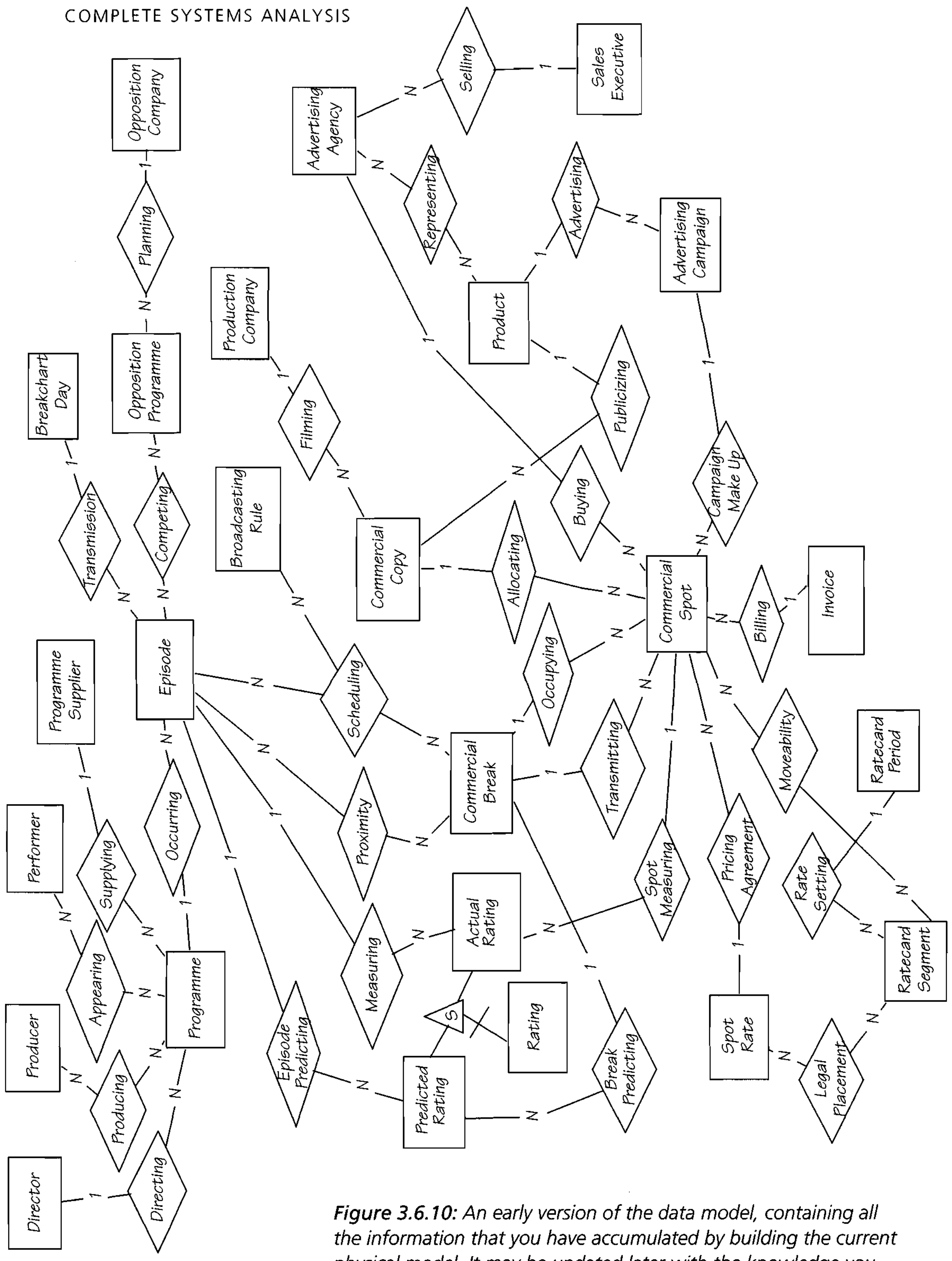

Figure 3.6.10: *An early version of the data model, containing all the information that you have accumulated by building the current physical model. It may be updated later with the knowledge you gain from doing essential modeling.*

Data Dictionary

This data dictionary defines the terms used in the current physical model and in the first-cut data model. You will notice that some of the definitions are not complete and that there are some questions in the dictionary. For example, some of the entities and relationships don't have a note defining their purpose, which indicates there are still unanswered questions about the relevancy and use of the data. Some of the entities have their unique identifiers underlined; those that don't need composite identifiers, which will be specified in the mini specifications. When you build your essential requirements models, you will get answers to such outstanding questions as you make changes, additions, and deletions to the definitions in the data dictionary.

Actual Rating = * Entity. A rating measurement taken during transmission. *
Actual Rating Percentage + Rating Time

Actual Rating Percentage = * Data element. The percentage of a given audience type watching Piccadilly Television at the time this rating was measured. *

Advertising = * Relationship. Keeps track of which products are involved in advertising campaigns. Cardinality: for each Product, there are many Advertising Campaigns; for each Advertising Campaign, there is one Product. Participation: Product optional, Advertising Campaign mandatory. *

Advertising Agency = * Entity. Buyer of commercial spots for advertising campaigns. *
<u>Agency Name</u> + Agency Address + Agency Phone Number

Advertising Campaign = * Entity. Records the conditions and aims for a campaign to advertise a product. *
<u>Campaign Number</u> + Campaign Start Date
+ Campaign End Date + Target Audience
+ Target Rating Percentage + Campaign Predicted Rating
+ Campaign Budget Total + Piccadilly Budget Amount
+ Campaign Duration + {Required Spot Duration}
* Work necessary to remove or justify the repeating group *

Agency Accounts File = ∗ Data store. Computerized current system accounts file. ∗
{Agency Name + Agency Address + Agency Phone Number + {Agency Invoice} + {Agency Payment}}

Agency Address = ∗ Data element ∗

Agency Invoice = ∗ Data flow ∗
Agency Name + Agency Address + Invoice Number + Invoice Date + {Campaign Number + {Spot Number + Spot Duration + Rate Moveability + Spot Transmitted Time + Spot Transmitted Date + Spot Price }} + Invoice Total

Agency Invoice Record = ∗ Data flow. A transaction in the current computer system. ∗
Agency Invoice

Agency Name = ∗ Data element. Identifier for an advertising agency. ∗

Agency Phone Number = ∗ Data element ∗

Agency Register = ∗ Data store. File of agencies and responsible sales executives kept by the Sales Department. ∗
{Agency Name + Agency Address + Agency Phone Number + Sales Executive Name + Servicing Start Date}

Agreed Campaign = ∗ Data flow. Spots and rates agreed between Piccadilly and an advertising agency. ∗
Agency Name + Agency Address + Campaign Number
+ Product Name + Campaign Start Date + Campaign End Date
+ {Spot Number + Spot Duration + Spot Price
+ Rate Moveability + Spot Booking Agreement
+ ([Breakchart Date + Break Start Time | {Breakchart Date}])}
+ {Unavailable Slot}

Airtime Analysis = ∗ Data flow ∗
{Breakchart Date + {Break Start Time + Break Sold Value + Break Unsold Value + {Seconds Available + (Rate Moveability)}}} + Total Sold Value + Total Unsold Value

Airtime Ratings History = ∗ Data store. Television ratings supplied by audience measurement bureaus. ∗
{Television Ratings Report}

Allocating = ∗ Relationship. Keeps track of which commercial copy should be transmitted for a commercial spot. Cardinality: for each Commercial Spot, there is one Commercial Copy; for each Commercial Copy, there are many Commercial Spots. Participation: Commercial Copy optional, Commercial Spot optional. ∗

Appearing = ∗ Relationship. Keeps track of which performers appear in programmes because this can affect which products can be advertised during a programme. Cardinality: for each Performer, there are many Programmes; for each Programme, there are many Performers. Participation: Programme optional, Performer mandatory. ∗

Audience Type = ∗ Data element. Used to classify ratings figures. ∗
[Homes | Homemakers | Adults | Men | Women | Children]

Automated Cassette Recording = ∗ An actual cassette containing a recording of a television commercial. This technology is used by the current system. ∗

Automatic Cassette Recordings = ∗ Data store and material store ∗
{Commercial Copy Number + Automated Cassette Recording + Production Company Name + Agency Name + Product Name}

Available Time = ∗ Data flow ∗
{Breakchart Date + {Break Start Time + {Seconds Available + (Rate Moveability)}}}

Average Break Predicted Rating = * Data element. Derivable: average of all Predicted Ratings for a Target Audience for all Episodes with a Proximity relationship to a given Commercial Break. Integer. *

Billing = * Relationship. Keeps track of which spots have been invoiced. Cardinality: for each Invoice, there are many Commercial Spots; for each Commercial Spot, there is one Invoice. Participation: Invoice mandatory, Commercial Spot optional. *

Breakchart = * Data store. Board containing hanging files plus breaksheets, used to record available time and sold time. *
{Breakchart Date + {Programme Name + Episode Start Time
+ Episode End Time} + {Break Start Time + Break End Time
+ Break Minimum Rate + {Spot Number + Product Name
+ Spot Duration + Spot Price + Rate Moveability
+ Spot Booking Agreement}}} + {Programming Rule}

Breakchart Date = * Data element. A date on which Piccadilly will transmit programmes and commercials. *

Breakchart Day = * Entity *
Breakchart Date + Breakchart Day Start
+ Breakchart Day End + Daily Revenue

Breakchart Day End = * Data element *

Breakchart Day Start = * Data element *

Break Duration = * Data element. Duration of a commercial break. Units: mins/secs. *
["1 min" | "1 min 30 secs" | "2 mins" | "2 mins 30 secs" | "3 mins"]

Break End Time = * Data element *

Break Minimum Rate = * Data element. Minimum rate to be charged for a break. *

Break Predicting = ✳ Relationship. Cardinality: for each Commercial Break, there are many Predicted Ratings; for each Predicted Rating, there is one Commercial Break. Participation: Commercial Break optional, Predicted Rating mandatory. ✳

Break Sold Value = ✳ Data element. Derivable: total of Spot Prices for each Commercial Spot that has an Occupying relationship with a given Commercial Break. ✳

Break Start Time = ✳ Data element ✳

Break Transmission Schedule = ✳ Data flow. Commercial Booking Department's notification of the scheduled transmission time and the spots that will occupy the next day's breaks. ✳
Breakchart Date + {Break Start Time + {Spot Number + Campaign Number + Spot Duration + Product Name}}

Break Unsold Value = ✳ Data element. Derivable:
((Break Duration x Break Minimum Rate) – Break Sold Value). ✳

Broadcasting Rule = ✳ Entity ✳
<u>Programming Rule + Rule Effective Date</u>

Budgeting = ✳ Relationship. Keeps track of the sales target for a number of breakchart days. Cardinality: for each Sales Target, there are many Breakchart Days; for each Breakchart Day, there is one Sales Target. Participation: Sales Target mandatory, Breakchart Day optional. ✳

Buying = ✳ Relationship. Cardinality: for each Advertising Agency, there are many Commercial Spots; for each Commercial Spot, there is one Advertising Agency. Participation: Advertising Agency optional, Commercial Spot mandatory. ✳

Campaign Availability = ✳ Data flow. Airtime available for a specific campaign.✳
{Breakchart Date + {Break Start Time + {Seconds Available + (Rate Moveability)}}}

Campaign Budget Total = ∗ Data element. The total campaign budget that the agency intends to spend on television advertising. Units: pounds sterling. ∗

Campaign Duration = ∗ Data element. Number of weeks during which spots for a campaign will be transmitted. Integer. Value range: >= 2 and <= 52. ∗

Campaign End Date = ∗ Data element. Date marking the last day spots will be transmitted for a campaign. ∗

Campaign File = ∗ Data store. File kept in the Sales Department. ∗
{Campaign Requirements + Suggested Campaign}

Campaign Make Up = ∗ Relationship. Keeps track of all the spots that make up a campaign plan to achieve the desired target ratings. Cardinality: for each Advertising Campaign, there are many Commercial Spots; for each Commercial Spot, there is one Advertising Campaign. Participation: Advertising Campaign optional, Commercial Spot mandatory. ∗

Campaign Number = ∗ Data element. Identifier for an advertising campaign. ∗

Campaign Predicted Rating = ∗ Data element. Derivable: total of the Predicted Ratings for Commercial Breaks that have an Occupying relationship with Commercial Spots in a given Advertising Campaign. ∗

Campaign Requirements = ∗ Data flow. An agency's description of requirements for an advertising campaign. ∗
Agency Name + Product Name + Campaign Budget Total
+ Piccadilly Budget Amount + Target Audience
+ Target Rating Percentage + Campaign Duration
+ Campaign Start Date + Campaign End Date
+ {Required Spot Duration}

Campaign Start Date = ∗ Data element. Date marking the first day spots for a campaign will be transmitted. ∗

*Commercial Break = * Entity. Each commercial break represents a number of seconds that can be sold for the transmission of advertising copy. **
Break Start Time + Break End Time + Break Duration
+ Break Sold Value + Break Unsold Value + Break Minimum Rate

*Commercial Copy = * Entity. The material that is transmitted during the time occupied by a commercial spot. **
<u>Commercial Copy Number</u> + Physical Copy
+ {Copy Transmission Date} + Disposal Date
** Note that this contains a repeating group so is not yet fully normalize. **

*Commercial Copy Number = * Data element. Unique identifier for commercial copy. **

*Commercial Copy Recording = * Data flow plus physical cassette. Commercial copy recording sent by the production company to Piccadilly's Programme Transmission Department. **
Commercial Copy Number + Automated Cassette Recording
+ Production Company Name + Agency Name + Product Name

*Commercial Spot = * Entity. Represents time that has been sold to an advertising agency. **
<u>Spot Number</u> + Spot Duration + Spot Booking Agreement

*Competing = * Relationship. Keeps track of which opposition programmes are in competition with the episodes that Piccadilly plans to transmit. Cardinality: for each Episode, there are many Opposition Programmes; for each Opposition Programme, there are many Episodes. Participation: Episode optional, Opposition Programme optional. **

*Containing = * Relationship. Keeps track of which commercial breaks are planned for a breakchart day. Cardinality: for each Commercial Break, there is one Breakchart Day; for each Breakchart Day, there are many Commercial Breaks. Participation: Breakchart Day optional, Commercial Break mandatory. **

Copy Disposal Instructions = * Data flow. Instruction from an agency to dispose of an outdated commercial copy recording. *
Commercial Copy Number + Product Name + Disposal Date

Copy Register = * Data store *
{Commercial Copy Number + Product Name + Campaign Number + {Copy Transmission Date} + (Disposal Date)}

Copy Transmission Date = * Data element. The date that a piece of copy is scheduled to be transmitted. *

Copy Transmission Instructions = * Data flow. Details of when particular commercial copy is to be transmitted. *
Agency Name + Product Name + Commercial Copy Number + {Copy Transmission Date}

Current Rate = * Data element. Derivable: the average rate currently being paid for a 30-second spot within a particular break. Units: pounds sterling. Integer. *

Daily Revenue = * Data element. Revenue made by Piccadilly for all the commercial spots transmitted on one breakchart day. Derivable: total of all Spot Prices for every Commercial Spot having a Transmitting relationship with the Commercial Breaks on today's date. *

Directing = * Relationship. Cardinality: for each Director, there are many Programmes; for each Programme, there is one Director. Participation: Director mandatory, Programme optional. *

Director = * Entity. The director of a programme. *
<u>Director Name</u>

Director Name = * Data element. Identifies the director of a programme. *

Displaced Spot = * Data flow *
Spot Number + Spot Price + Rate Moveability + Spot Booking Agreement

Disposal Copy Material = ✳ Data flow ✳
Commercial Copy Number + Physical Copy

Disposal Date = ✳ Data element. Date by which a piece of commercial copy must be destroyed. ✳

Episode = ✳ Entity. An episode is one occurrence of a programme. ✳
Episode Number + Episode Scheduled Date
+ Episode Start Time + Episode End Time

Episode End Time = ✳ Data element ✳

Episode Number = ✳ Data element ✳

Episode Predicting = ✳ Relationship. Cardinality: for each Episode, there are many Predicted Ratings; for each Predicted Rating, there is one Episode. Participation: Episode optional, Predicted Rating mandatory. ✳

Episode Scheduled Date = ✳ Data element ✳

Episode Start Time = ✳ Data element ✳

Filming = ✳ Relationship. Cardinality: for each Production Company, there are many Commercial Copy(s); for each Commercial Copy, there is one Production Company. Participation: Production Company mandatory, Commercial Copy mandatory. ✳

Invoice = ✳ Entity. A record of money owed by an advertising agency. ✳
<u>Invoice Number</u> + Invoice Total + Invoice Date

Invoice Date = ✳ Data element ✳

Invoice Number = ✳ Data element ✳

Invoice Total = ✳ Data element. Derivable: sum of the Spot Price related to each Commercial Spot related to a given Invoice. ✳

Legal Placement = ✳ Relationship. Keeps track of the ratecard-selling conditions concerning the price of a spot and the segments of time that it can occupy. Cardinality: for each Spot Rate, there are many Ratecard Segments; for each Ratecard Segment, there are many Spot Rates. Participation: Spot Rate mandatory, Ratecard Segment mandatory. ✳

Likely Spot Preemption = ✳ Data flow. Trends in airtime sales indicate that this spot may be preempted. ✳
Spot Number + Product Name + Break Start Time
+ Spot Duration + Rate Moveability + Spot Price
+ Recommended Rate

Measuring = ✳ Relationship. Keeps track of the actual ratings for an episode. Cardinality: for each Episode, there are many Actual Ratings; for each Actual Rating, there is an Episode. Participation: Actual Rating mandatory, Episode optional. ✳

Moveability = ✳ Relationship. Keeps track of the ratecard segments in which a particular commercial spot may be transmitted. Cardinality: for each Commercial Spot, there are many Ratecard Segments; for each Ratecard Segment, there are many Commercial Spots. Participation: Commercial Spot mandatory, Ratecard Segment optional. ✳

Moveable Spots = ✳ Data flow ✳
Break Start Time + {Spot Number + Spot Duration + Spot Price + Rate Moveability + Spot Booking Agreement}

New Agency = ✳ Data flow ✳
Agency Name + Agency Address + Agency Phone Number

New Agency Form = ✳ Data flow ✳
Agency Name + Agency Address + Agency Phone Number

New Agency Transaction = ✳ Data flow ✳
Agency Name + Agency Address + Agency Phone Number

New Breaks = ✳ Data flow. Breaks progressively added to the breakchart for three months in the future. ✳
{Breakchart Date + Break Start Time + Break End Time + Break Duration}

New Programme = ✳ Data flow. Describes a programme offered by an English or overseas programme supplier. ✳
Programme Name + Programme Type + Programme Description + Programme Duration + Programme Price + Programme Episodes + {Performer Name} + (Producer Name + Director Name) + Supplier Name

New Sales Executive = ✳ Data flow ✳
Sales Executive Name + Sales Executive Address + Sales Executive Start Date

New Spot Rates = ✳ Data flow. A transaction in the current computer system.✳
Ratecard

Occupying = ✳ Relationship. Keeps track of which spots are occupying a break so that the spot manipulators can make decisions about moving and slotting spots. Cardinality: for each Commercial Spot, there is one Commercial Break; for each Commercial Break, there are many Commercial Spots. Participation: Commercial Spot optional, Commercial Break optional. ✳

Occurring = ✳ Relationship. Keeps track of the episodes of a programme. Cardinality: for each Programme, there are many Episodes; for each Episode, there is one Programme. Participation: Programme optional, Episode mandatory. ✳

Opposition Company = ✳ Entity. Piccadilly competes with the government channels for viewing audiences. Piccadilly also competes with other commercial channels for the advertiser's money. ✳
<u>Television Company Name</u>

Opposition Predicted Rating = ✳ Data element. The rating predicted for one of the opposition company's programmes. ✳

*Opposition Programme = * Entity. Piccadilly's programme purchasing and scheduling is influenced by the programmes planned for transmission by other commercial channels and by the government channels. **
<u>Opposition Transmission Date + Opposition Transmission Time + Opposition Programme Name</u> + Opposition Predicted Rating

*Opposition Programme Name = * Data element. The name of a programme transmitted by an opposition company. **

*Opposition Schedule = * Data flow **
Television Company Name + {Opposition Transmission Date + Opposition Transmission Time + Opposition Programme Name + (Opposition Predicted Rating)}

*Opposition Transmission Date = * Data element. The date an opposition programme is scheduled for transmission. **

*Opposition Transmission Time = * Data element. The time an opposition programme is scheduled for transmission. **

*Performer = * Entity. Actor or actress appearing in a programme. **
<u>Performer Name</u>

*Performer Name = * Data element. Identifier of an actor or actress appearing in a programme. **

*Physical Copy = * A physical copy of the material to be transmitted for a commercial spot. Current system uses ACR. **

*Piccadilly Budget Amount = * Data element. The portion of the campaign budget total that the agency intends to spend with Piccadilly. Units: pounds sterling. **

*Planning = * Relationship. Cardinality: for each Opposition Company, there are many Opposition Programmes; for each Opposition Programme, there is one Opposition Company. Participation: Opposition Company optional, Opposition Programme mandatory. **

*Predicted Rating = * Entity. Predicted ratings are used to plan advertising campaigns and programme scheduling. **
Predicted Rating Percentage + Predicted Rating Date

*Predicted Rating Date = * Data element. Date on which a predicted rating was made. **

*Predicted Rating Percentage = * Data element. Percentage of an audience type predicted to watch a given programme episode. **

*Predicted Ratings = * Data flow. The Research Department's predictions of programme ratings. **
{Breakchart Date + {Programme Name + Episode Number
+ {Audience Type + Predicted Rating Percentage
+ Predicted Rating Date}}}

*Preempted Spot = * Data flow. A spot that has been preempted by another spot and dropped from the breakchart. **
Spot Number + Spot Duration + Rate Moveability

*Preemption Replacement = * Data flow. Details of a spot that has been booked to replace a preempted spot. **
Agency Name + Product Name + Campaign Number
+ Preempted Spot + [Replacement Spot | Upgrade Unavailable]

*Preemption Warning = * Data flow. A sales executive warning to an agency that a spot is in danger of being preempted . **
Agency Name + Product Name + Campaign Number
+ Spot Number + Breakchart Date + Break Start Time
+ Spot Duration + Recommended Rate

*Preliminary Schedule = * Data flow. The Programme Transmission Department's first cut of a programme transmission schedule. **
{Programme Transmission Date + {Programme Name
+ Episode Start Time + Episode End Time}}

Priced Time =* Data flow *
Agency Name + Product Name + Campaign Start Date + Campaign End Date + {Breakchart Date + {Break Start Time + Seconds Available + Total Available Price + Current Rate + Audience Type + Average Break Predicted Rating}}

Pricing Agreement = * Relationship. Keeps track of ratecard pricing conditions under which a spot is sold. Cardinality: for each Commercial Spot, there is one Spot Rate; for each Spot Rate, there are many Commercial Spots. Participation: Commercial Spot mandatory, Spot Rate optional. *

Producer = * Entity. The producer of a programme. *
<u>Producer Name</u>

Producer Name = * Data element. Identifies the producer of a programme. *

Producing = * Relationship. Cardinality: for each Programme, there are many Producers; for each Producer, there are many Programmes. Participation: Programme optional, Producer mandatory. *

Product = * Entity. Piccadilly needs to record which products are being advertised because some of the broadcasting rules govern the placement of commercials depending on the product. *
<u>Product Name</u>

Production Company = * Entity. Producer of television commercials. *
<u>Production Company Name</u>

Production Company Name = * Data element *

Product Name = * Data element. Identifier for one of the products represented by an advertising agency. *

Product Number = * Data element. Piccadilly's unique identifier for a product. *

*Programme = * Entity. A television programme made by Piccadilly or bought from a programme supplier. **
<u>Programme Name</u> + Programme Type + Programme Description + Programme Duration + Programme Price + Programme Episodes + Programme Purchase Date

*Programme Description = * Data element. Synopsis of the contents of a programme. **

*Programme Duration = * Data element. Running time of a programme. Units: hrs/mins/secs. **

*Programme Episodes = * Data element. The number of episodes of a programme covered by an agreement with a supplier or by Piccadilly's internal production plans. **

*Programme Name = * Data element. The name that uniquely identifies this programme, for example, News at Ten, Brideshead Revisited, Coronation Street. **

*Programme Price = * Data element. Price paid to the supplier of a programme. Units: pounds sterling. **

*Programme Purchase Agreement = * Data flow. Terms negotiated with an external supplier. **
Programme Name + Supplier Name + Programme Price

*Programme Purchase Date = * Data element. Date of purchase from a programme supplier. **

*Programme Schedule File = * Data store. File kept by the Research Department. **
{Programme Transmission Schedule + Preliminary Schedule + Predicted Ratings}

Programme Schedules = ✳ Data store. File kept in the Sales Department. ✳
{Programme Transmission Schedule} + {Predicted Ratings}

Programme Supplier = ✳ Entity ✳
<u>Supplier Name</u>

Programme Transmission Date = ✳ Data element. The actual transmission date of a particular episode. ✳

Programme Transmission Schedule = ✳ Data flow. Piccadilly's planned transmission for the next quarter. ✳
{Programme Transmission Date + {Episode Number
+ Episode Start Time + Episode End Time + Programme Name
+ Programme Description + Programme Type
+ Predicted Rating} + {Break Start Time + Break End Time}}

Programme Transmission Time = ✳ Data element. The actual transmission time of an episode. ✳

Programme Type = ✳ Data element ✳
[First-Run Film | Sporting Event | Documentary | Talk Show | Old Movie]
✳ The types of programmes that Piccadilly transmits. Note that there are other programme types to add to these. ✳

Programming Plan = ✳ Data store. File in the Programme Transmission Department. ✳
{Programme Name + Programme Type + Programme Description
+ Programme Duration + Programme Price
+ Programme Purchase Date + {Performer Name}
+ (Producer Name) + (Director Name) + Supplier Name
+ {Programme Transmission Date}} + Preliminary Schedule
+ Programme Transmission Schedule + Opposition Schedule
+ Programming Rules

Programming Rule = ✳ Data element. A rule set by the Broadcasting Board. Each rule addresses some aspect of the mixture, content, and placement of programmes and commercial breaks. ✳

Programming Rules = * Data flow. File in the Programme Transmission Department. *
{Programming Rule}

Proximity =* Relationship. Keeps track of which breaks are within one hour of an episode so that the spot manipulators can apply the appropriate programming rules when moving and slotting spots. Cardinality: for each Commercial Break, there are many Episodes; for each Episode, there are many Commercial Breaks. Participation: Commercial Break mandatory, Episode optional. *

Publicizing = * Relationship. Keeps track of which product is advertised by a particular piece of commercial copy. Cardinality: for each Commercial Copy, there is one Product; for each Product, there are many Commercial Copy(s). Participation: Commercial Copy mandatory, Product optional. *

Purchase Decision = * Data flow *
Programme Name + Programme Type + Programme Description
+ Programme Duration + Programme Price
+ Programme Episodes + {Opposition Programme}
+ {Performer Name} + {Producer Name} + (Director Name)
+ Supplier Name + {Episode Scheduled Date
+ Episode Start Time + Episode End Time}

Ratecard = * Data flow. Prices, moveability, and preemption rules of time available for sale. *
Rate From Date + {Rate Spot Duration + {Rate Segment Day
+ {Rate Segment Start + Rate Segment End
+ {Rate Moveability + Spot Price}}}}

Ratecard File = * Data store. File kept by the Research Department. *
{Ratecard}

Ratecard Period = * Entity. Period during which given rates apply. *
<u>Rate From Date</u>

Ratecards = * Data store. File kept in the Sales Department. There is a duplicate file in the Commercial Booking Department. *
{Ratecard}

*Ratecard Segment = * Entity. Continuous band of time defined on ratecard. *
Rate Segment Day + Rate Segment Start + Rate Segment End*

*Rate From Date = * Data element. The commencement date for a ratecard period. Format: Day/Month/Year. **

*Rate Moveability = * Data element. Rate moveability as defined in the ratecard. *
[Fixed: fixed on a nominated day | Broad: moveable within a specified segment on a nominated day | ROD: run-of-day, moveable to any similarly priced segment on a nominated day | ROW: run-of-week, moveable to any similarly priced segment during a week]*

*Rate Segment Day = * Data element. Day(s) of week on which this segment of time occurs. *
[Weekday | Saturday | Sunday]*

*Rate Segment End = * Data element. The end time for a ratecard segment. **

*Rate Segment Start = * Data element. The start time for a ratecard segment. **

*Rate Setting = * Relationship. Keeps track of the ratecard segments applicable to a ratecard period. Cardinality: for each Ratecard Segment, there is one Ratecard Period; for each Ratecard Period, there are many Ratecard Segments. Participation: Ratecard Segment mandatory, Ratecard Period mandatory. **

*Rate Spot Duration = * Data element. Duration of spots as defined in the ratecard. Units: seconds. *
[10 | 20 | 30 | 40 | 50 | 60]*

*Rating = * Entity *
Rating Date + Audience Type*

*Rating Date = * Data element. The date to which a rating refers. **

Rating Time = * Data element. Time to which an actual rating refers. Ratings are taken every minute. Units: hrs/mins/secs. *

Recommended Rate = * Data flow. The rate recommended for a spot that is in danger of preemption. *
Rate Moveability + Spot Price

Replacement Spot = * Data flow. A spot to replace another spot that has been preempted. *
Product Name + Spot Number + Spot Duration + Spot Price + Rate Moveability

Representation End Date = * Data element. Date an agency ends representing a product. *

Representation Start Date = * Data element. Date an agency starts representing a product. *

Representing = * Relationship. Keeps track of which agency is responsible for a product. Keeps track of historical relationships between agencies and products for collecting bad debts. Cardinality: for each Advertising Agency, there are many Products; for each Product, there are many Advertising Agency(s). Participation: Advertising Agency mandatory, Product mandatory. *
Representation Start Date + Representation End Date

Required Spot Duration = * Data element. Required length of commercial spots for a campaign. See Spot Duration for values. *

Revenue Reports = * Data flow. Computer reports used by management to help set sales targets. *
Breakchart Date + Daily Revenue + {Spot Number + Spot Rate + Product Name + Spot Transmitted Time + Spot Price}

Rule Effective Date = * Data element. The date that a new rule from the Broadcasting Board must be applied to all Piccadilly's programme and commercial transmissions. *

Sales Executive = * Entity *
Sales Executive Name + Sales Executive Address
+ Sales Executive Start Date

Sales Executive Address = * Data element *

Sales Executive Name = * Data element *

Sales Executive Register = * Data store. File of sales executives' responsibilities kept by the Sales Department. *
{Sales Executive Name + {Agency Name + Servicing Start Date
+ Servicing End Date}}

Sales Executive Start Date = * Data element. The date a sales executive starts working for Piccadilly. *

Sales Target = * Entity. The revenue that Piccadilly aims to make within a specified period. *
Sales Target From + Sales Target To + Sales Target Amount

Sales Target Amount = * Data element. The amount of a sales target. Units: pounds sterling rounded up to the nearest thousand. *

Sales Target File = * Data store *
{Sales Target Instructions}

Sales Target From = * Data element. The start date for a sales target. *

Sales Target Instructions = * Data flow. Set by Piccadilly management. *
Sales Target From + Sales Target To + Sales Target Amount

Sales Target To = * Data element. The end date of a sales target. *

*Scheduling = * Relationship. Keeps track of the scheduling decisions made concerning episodes, commercial breaks, and broadcasting rules. Cardinality: for each instance of one Episode and one Broadcasting Rule, there are many Commercial Breaks; for each instance of one Commercial Break and one Broadcasting Rule, there are many Episodes; for each instance of one Commercial Break and one Episode, there are many Broadcasting Rules. Participation: Broadcasting Rule optional, Commercial Break optional, Episode optional. **

*Seconds Available = * Data element. Derivable: Break Duration minus total of Spot Durations for all Commercial Spots that have an Occupying relationship with this Commercial Break. Units: seconds. Value range: 0 to 180. **

*Seconds Sold = * Data element. Derivable: sum of Spot Durations for all Commercial Spots that have an Occupying relationship with a Commercial Break. **

*Selected Ratings = * Data flow. Selected records from the television ratings report that are stored by the Research Department. **
Television Ratings Report

*Selected Spots = * Data flow. Spots selected by an agency as part of a campaign. **
Agency Name + Campaign Number + Product Name
+ Campaign Start Date + Campaign End Date + {Spot Number
+ Spot Duration + Spot Price + Rate Moveability
+ ([Breakchart Date + Break Start Time | {Breakchart Date}])}

*Selling = * Relationship. Keeps track of which sales executive is responsible for selling to an advertising agency. Cardinality: for each Advertising Agency, there are many Sales Executives; for each Sales Executive, there are many Advertising Agency(s). Participation: Advertising Agency mandatory, Sales Executive mandatory. **
Servicing Start Date + (Servicing End Date)

Servicing End Date = ✳ Data element. The date a sales executive stops servicing an advertising agency. ✳

Servicing Start Date = ✳ Data element. The date on which a sales executive starts servicing an advertising agency. ✳

Slotted Spot = ✳ Data flow. Relocation of a spot due to its place being taken by another spot at a higher rate. ✳
Breakchart Date + Break Start Time + Spot Number
+ Spot Price + Rate Moveability + Spot Booking Agreement

Spot Booking Agreement = ✳ Data element grouping. The conditions under which a spot is sold. ✳
[Spot Date Agreed + Spot Break Agreed | Spot Date Agreed
+ Spot Segment Agreed | Spot Date Agreed | Spot Week
Agreed + {Spot Segment Agreed}]

Spot Break Agreed = ✳ Data element. Agreed break in which a spot must be transmitted. ✳

Spot Cancellation = ✳ Data flow ✳
Agency Name + Product Name + Campaign Number
+ Spot Number

Spot Date Agreed = ✳ Data element. Agreed date on which a spot must be transmitted. ✳

Spot Duration = ✳ Data element. Duration of a commercial spot. Units: seconds. ✳
[10 | 20 | 30 | 40 | 50 | 60]

Spot Measuring = ✳ Relationship. Keeps track of the actual ratings for a commercial spot. Cardinality: for each Commercial Spot, there are many Actual Ratings; for each Actual Rating, there is one Commercial Spot. Participation: Actual Rating mandatory, Commercial Spot optional. ✳

*Spot Number = * Data element. Identifier for a commercial spot. **

*Spot Price = * Data element. Ratecard price for a spot rate. Units: pounds sterling. Value range: >= 150 and <= 25000. **

*Spot Rate = * Entity **
Rate Spot Duration + Spot Price + Rate Moveability

*Spot Rate File = * Data store. File in the current computer system. **
{Ratecard}

*Spot Segment Agreed = * Data element. Agreed segment in which a spot will be transmitted .**

*Spot Sticker = * Data flow **
Spot Number + Spot Duration + Spot Price + Rate Moveability
+ Spot Booking Agreement + Product Name
+ Campaign Number

*Spot Transmitted Date = * Data element. The breakchart day on which a commercial spot is transmitted. **

*Spot Transmitted Time = * Data element. The time when a commercial spot is transmitted. Units: hrs/mins/secs. **

*Spot Upgrade Request = * Data flow. Increase in rate to avoid preemption. **
Agency Name + Product Name + Campaign Number
+ {Spot Number + Spot Duration + Rate Moveability}

*Spot Week Agreed = * Data element. Agreed week during which a spot must be transmitted. **

*Spots Transmitted = * Data flow **
Spot Transmitted Date + {Agency Name + {Campaign Number
+ {Spot Number + Spot Transmitted Time}}}

*Suggested Campaign = * Data flow. Suggestions to an agency about the makeup of an advertising campaign. *
Agency Name + Product Name + Campaign Number
+ Campaign Start Date + Campaign End Date
+ {Required Spot Duration + Spot Price + Rate Moveability
+ ([Breakchart Date + Break Start Time | {Breakchart Date}])}
+ Target Rating Percentage + Campaign Predicted Rating*

*Suitable Break = * Data flow. Same definition as Moveable Spots. **

*Suitable Time = * Data flow *
Agency Name + Product Name + Campaign Start Date
+ Campaign End Date + {Breakchart Date + Break Start Time
+ Seconds Available + Current Rate + Audience Type
+ Average Break Predicted Rating}*

*Supplier Name = * Data element. Identification for a programme supplier. **

*Supplying = * Relationship. Keeps track of the programmes that are supplied by a programme supplier. Cardinality: for each Programme Supplier, there are many Programmes; for each Programme, there is one Programme Supplier. Participation: Programme mandatory, Programme Supplier optional. **

*Target Audience = * Data element. The audience at which a campaign is aimed. See Audience Type for values. **

*Target Rating Percentage = * Data element. Percentage rating aimed for by a campaign. Integer. Value range: > 0 and <= 100. **

*Television Company Name = * Data element **

*Television Ratings Report = * Data flow. Statistical analysis of programmes and types of viewers showing ratings that are measured every 5 minutes during the viewing day. *
{Rating Date + {Programme Name + Rating Time
+ {Audience Type + Actual Rating Percentage}}}*

*Time Sold = * Data flow **
{Breakchart Date + {Break Start Time + Break Sold Value + {Seconds Sold + Rate Moveability}}} + Total Sold Value

*Total Available Price = * Data element. Derivable: ((Available Seconds/30) x Spot Price for fixed spot in Ratecard Segment corresponding to Commercial with Spot Duration equal to 30 secs). **

*Total Sold Value = * Data element. Derivable: sum of Break Sold Value for all the Commercial Breaks within a specific period. **

*Transmission = * Relationship. Keeps track of when an episode of a programme is transmitted. Cardinality: for each Episode, there is one Breakchart Day; for each Breakchart Day, there are many Episodes. Participation: Episode optional, Breakchart Day optional. **
Programme Transmission Time + Programme Transmission Date

*Transmission Log = * Data store **
Spot Transmitted Date + {Spot Number + Spot Booking Agreement + Spot Price + Rate Moveability + Product Name + Spot Transmitted Time}

*Transmission Times = * Data flow **
Spot Transmitted Date + {Spot Number + Spot Booking Agreement + Spot Price + Rate Moveability + Product Name + Spot Transmitted Time}

*Transmission Transaction = * Data flow **
Spot Transmitted Date + {Spot Number + Spot Booking Agreement + Spot Price + Rate Moveability + Product Name + Spot Transmitted Time}

*Transmitting = * Relationship. Keeps track of when a spot is transmitted. Cardinality: for each Commercial Spot, there is one Commercial Break; for each Commercial Break, there are many Commercial Spots. Participation: Commercial Spot optional, Commercial Break optional. **
Spot Transmitted Time + Spot Transmitted Date

*Unavailable Campaign = * Data flow. Piccadilly has no time available to suit the campaign requirements. **
Agency Name + Product Name + Target Audience
+ Target Rating Percentage + Campaign Duration
+ Campaign Start Date + Campaign End Date
+ Required Spot Duration

*Unavailable Slot = * Data element grouping. A suitable slot cannot be found for this spot at this price. The recommendation is to upgrade the spot to the next higher price. **
Spot Number + Spot Price + Spot Duration

*Upgrade Confirmation = * Data flow. Agreement with an agency to upgrade the rate of a spot. **
Agency Name + Campaign Number + Product Name
+ Spot Number + Spot Duration + Spot Price
+ Rate Moveability

*Upgraded Rate = * Data flow **
Spot Booking Agreement + Spot Price + Rate Moveability

*Upgrade Unavailable = * Data flow. Message sent to an agency when it is not possible to upgrade a preempted spot. **
Spot Number + Spot Duration

REVIEW: IDENTIFYING EVENTS 3.7

Before You Reached Here ...
You have identified and listed all the events for the Piccadilly system. This exercise was given in Chapter 1.8 *Identifying Events.*

The Context Is Your Guide

The problem statement in Chapter 1.8 suggested that the Piccadilly context diagram from the current physical model (Figure 3.6.1) is the primary input to building the event list. Figure 3.7.1 repeats that diagram except that each boundary data flow is connected to an event by its number in the event list in Figure 3.7.2.

Piccadilly Event List

By going through the remainder of the context diagram and referring to the background material on Piccadilly as well as the lower-level diagrams, we came up with the event list in Figure 3.7.2.

Your task was to name all the events. Let's start with the data flow CAMPAIGN REQUIREMENTS, which is tagged to event 1. Since the flow comes from the terminator ADVERTISING AGENCIES, it's an external event. Something happens in the agencies to cause this data flow to enter the Piccadilly context. We suggested that descriptive names for external events consist of the terminator's name, and the reason that the terminator sends the system the data.

Following our suggestion and using the singular form, you'd name the first part of the event name "Agency." Next consider the reason an agency sends CAMPAIGN REQUIREMENTS to Piccadilly. Remember, Stamford Brook told you, "Our clients are the advertising agencies, which are hired by companies that want to run advertising campaigns for their products." So the reason that agencies send CAMPAIGN REQUIREMENTS to Piccadilly is they want to run a campaign. This means the event should be called *Agency wants to run a campaign.* So much for the first one. You should give the other events similarly descriptive names. (Ours are in the list in Figure 3.7.2.)

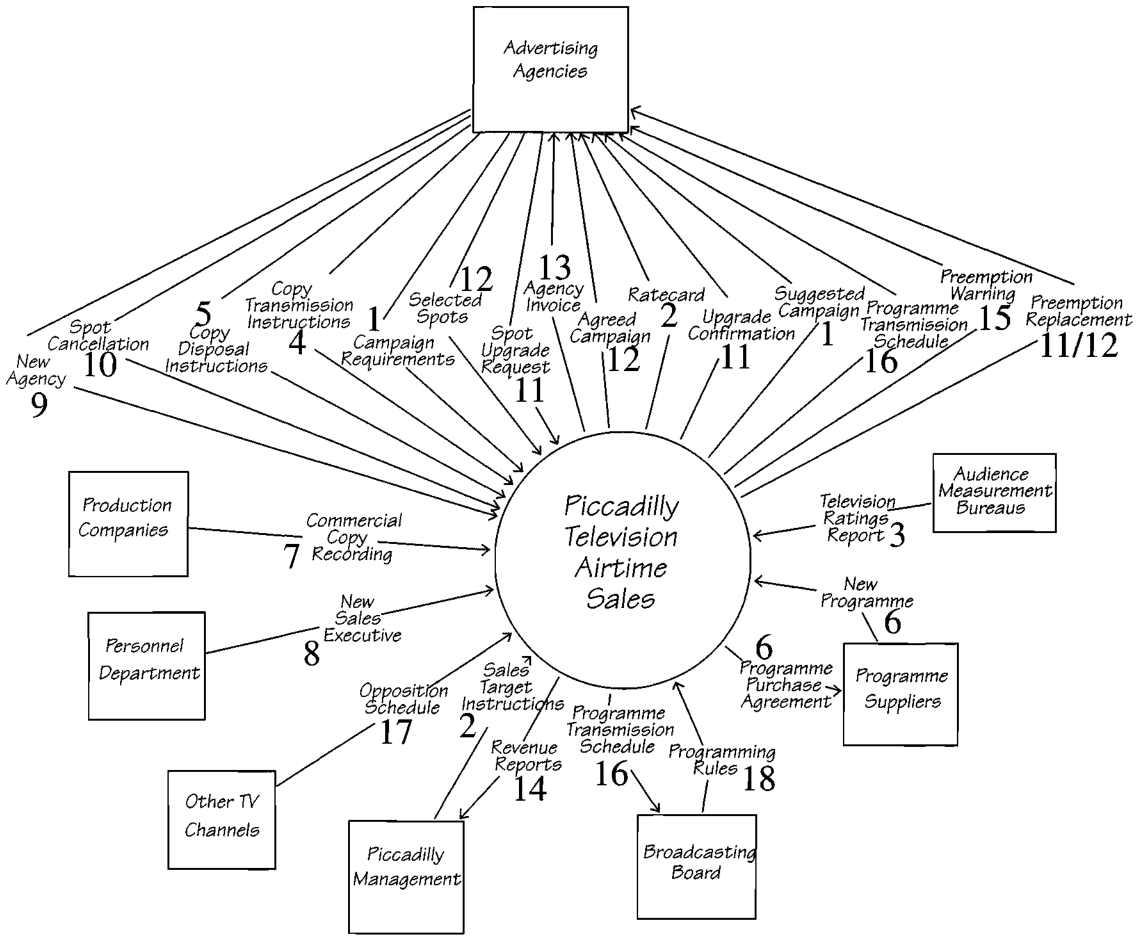

Figure 3.7.1: *Piccadilly context diagram with tagged data flows. The numbers correspond to those in the event list in Figure 3.7.2.*

Event Name	Associated Data Flows
1. Agency wants to run a campaign	CAMPAIGN REQUIREMENTS (IN) SUGGESTED CAMPAIGN (OUT)
2. Management sets a sales target	SALES TARGET INSTRUCTIONS (IN) RATECARD (OUT)
3. Bureau prepares TV ratings	TELEVISION RATINGS REPORT (IN)
4. Agency decides the transmission instructions for a commercial	COPY TRANSMISSION INSTRUCTIONS (IN)
5. Agency decides a commercial is outdated	COPY DISPOSAL INSTRUCTIONS (IN)
6. Supplier wants to sell a new programme	NEW PROGRAMME (IN) PROGRAMME PURCHASE AGREEMENT (OUT)
7. Production company makes a commercial	COMMERCIAL COPY RECORDING (IN)
8. Personnel hires a sales executive	NEW SALES EXECUTIVE (IN)
9. New agency wants to do business	NEW AGENCY (IN)
10. Agency cancels a spot	SPOT CANCELLATION (IN)
11. Agency wants to upgrade a spot	SPOT UPGRADE REQUEST (IN) UPGRADE CONFIRMATION (OUT) PREEMPTION REPLACEMENT (OUT)
12. Agency chooses spots for a campaign	SELECTED SPOTS (IN) AGREED CAMPAIGN (OUT) PREEMPTION REPLACEMENT (OUT)
13. Spots are transmitted	AGENCY INVOICE (OUT)
14. Time to analyze revenue	REVENUE REPORT (OUT)
15. Time to analyze the breakchart	PREEMPTION WARNING (OUT)
16. Time to finalize new programme schedule	2:2 {PROGRAMME TRANSMISSION SCHEDULE} (OUT)
17. Another channel sets a schedule	OPPOSITION SCHEDULE (IN)
18. Broadcasting Board makes rules	PROGRAMMING RULES (IN)

Figure 3.7.2: *Event list for Piccadilly.*

You were also asked to annotate your list with all the data flows connected with each event response. The current physical model for the Sales Department (Figure 3.6.5) reveals that SUGGESTED CAMPAIGN is output from this event response. That gives us

1. *Agency wants to run a campaign*	*CAMPAIGN REQUIREMENTS (IN)* *SUGGESTED CAMPAIGN (OUT)*

Temporal events take place because the system has a contract with a terminator to provide information at a certain time. You begin temporal event names with "Time to," and you add whatever the system is expected to do. For example, the information about the Programme Transmission Department in Chapter 1.4 *The Piccadilly Organization* told you, "Every quarter, finalized programme transmission schedules are sent to the Sales, Commercial Booking, and Research departments, as well as to the Broadcasting Board for its review. The version of the schedule that Perry sends to all of the agencies highlights the new high-rating programmes in the hope that it will encourage them to book their spots early."

This description says that an event occurs whenever it is *Time to finalize new programme schedule,* and the associated output data flow is, naturally enough, PROGRAMME TRANSMISSION SCHEDULE. This is in our list as event 16. Notice the annotation 2:2 {PROGRAMME TRANSMISSION SCHEDULE} to indicate that two copies of the schedule leave the context of the system as a result of event 16. Your context diagram shows that one copy is sent to the advertising agencies and another is sent to the Broadcasting Board. We know there are three other copies of the schedule that are sent to internal Piccadilly departments, but you are not concerned with these internal flows when making your event list. For the moment you are focusing only on the context flows for each event because this will provide you with minimally connected subsystems, one for each event. Later, when you model the details of each event response, you will deal with these internal flows.

Note that we are only dealing with the events that are caused by the flows in the context diagram. These flows concentrate on activities that are fundamental to the system. If your work in event partitioning caused you to identify other potential context flows and events that are concerned with maintaining the data, congratulations! You have exceeded requirements. We'll come back later to discuss these other events.

✚ Ski Patrol

We anticipate you might have had problems with a few things: first, the names. The event names you choose should be descriptive enough to indicate the likely response that the system makes. If the name is too general, or it only describes the data flow's arrival, your users may well say, "So what?" and be unable to confirm the information. On the other hand, if you have good, descriptive names, the next part of the Project—building event-response models—will be a lot easier. Take a few moments now to review your event names and to satisfy yourself that they cannot be improved.

A difficult part of this exercise is deciding exactly what is an event, and what is part of an event. Think of an event response as a chain reaction that continues until all the resulting data flows have reached data stores or terminators.

The response may end before you expect. Take, for example, event 1 *Agency wants to run a campaign.* The incoming flow CAMPAIGN REQUIREMENTS triggers the sales executive into action. When the response generates the flow SUGGESTED CAMPAIGN, the action finishes. Because SUGGESTED CAMPAIGN goes to a terminator, and the system has to wait for the terminator's action, that's the end of the response. The executive now has to wait until the agency decides what it's going to do. When the terminator does act, it is a separate event, and has its own response. The agency contact may call back soon to say that the campaign will run, or the agency may take a few days to call back. (When the agency responds, it is event 12.) The agency may never call back.

A response may generate some data flows that you didn't expect. We show event 12 *Agency chooses spots for a campaign* with these data flows:

SELECTED SPOTS (IN)
AGREED CAMPAIGN (OUT)
PREEMPTION REPLACEMENT (OUT)

Why is PREEMPTION REPLACEMENT included as part of the response to this event? Why isn't it the response to a separate event *Spot is preempted?* To answer this, let's look at what happens when the flow SELECTED SPOTS arrives at Piccadilly (see Figure 3.7.3).

The agency tells the sales executive which spots are wanted for a campaign. The executive tells the Commercial Booking people about the agreed campaign so that they can put the spots on the breakchart. Sometimes, placing new spots on the breakchart causes spots from another campaign to be preempted. When this happens, the Commercial Booking people tell the appropriate sales executives, who each tell the agency concerned about the preemption and recommend a preemption replacement. In this case, PREEMPTION REPLACEMENT is part of the chain reaction when the system responds to an agency choosing spots for a campaign.

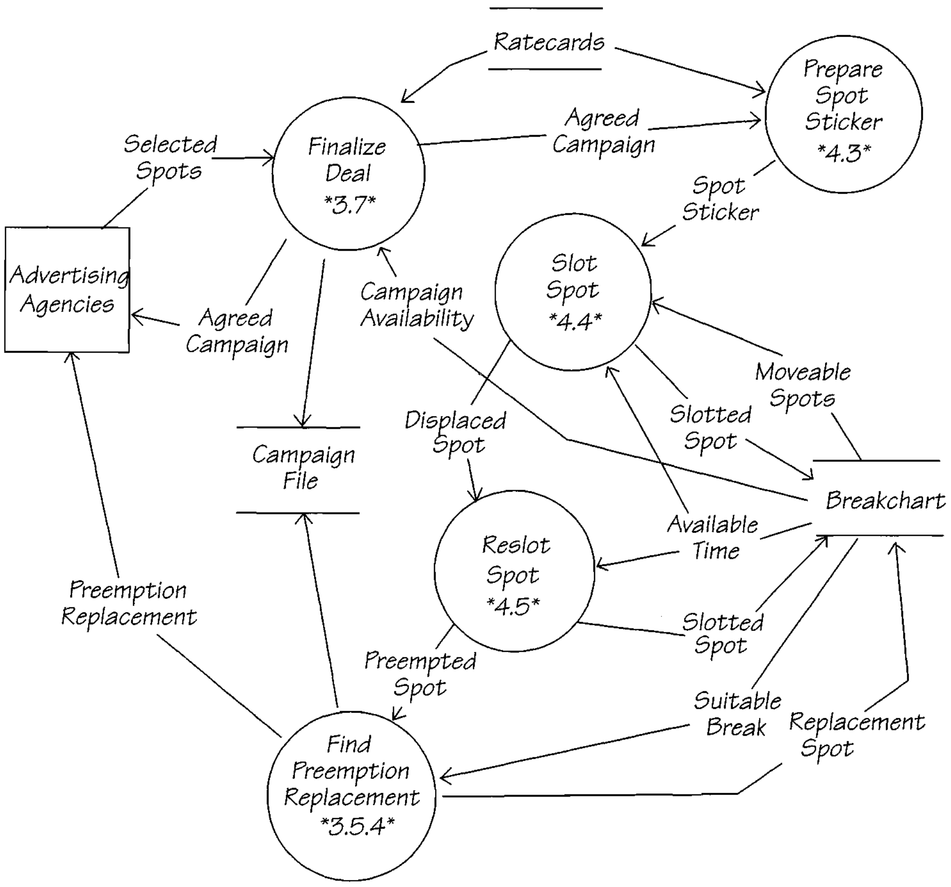

Figure 3.7.3: Event-response model for event 12 **Agency chooses spots for a campaign.** *The numbers in the bubbles are those of the process in the current physical model. They are included to make it easier for you to relate this model to the diagrams in the current physical model.*

Notice that the response to event 11 *Agency wants to upgrade a spot* can also produce the data flow PREEMPTION REPLACEMENT. When a spot is upgraded, it may displace another spot, which in turn may displace another, and so on. If there is no more room on the breakchart for the last displaced spot, it is preempted, and the chain reaction results in a PREEMPTION REPLACEMENT.

Don't be tempted to group similar processes and call it one event. For example, all of the agency's communications about spots might be lumped together under the name *Agency changes its spots.* This event would cover events 10, 11, and

12. Even if the agency sent the same type of message (say it used the same printed form for all three events), the response the system would make is different in each case. If the events require different responses by the system, they are different events. The result of following these guidelines is manageably sized subsystems, one for each event. Later on in the Project, you will model each event response in detail and you will model the inter-event dependencies.

When you are satisfied with your efforts, proceed to your next assignment.

Trail Guide

● ■ ◆ ❄ All trails: Return to Chapter 1.8 *Identifying Events,* to "A Strategic Point" heading, where there is some more information you'll need before proceeding.

INDEX