

Experience with the COCOMO II Application Point Model

Richard D. Stutzke
Science Applications International Corp.
6725 Odyssey Drive
Huntsville, AL 35806-3301

(256) 971-6224 (office)
(256) 971-6550 (facsimile)

Richard.D.Stutzke@saic.com

25 October 2000

Presented at the 15th International Forum on COCOMO and
Software Cost Modeling, Los Angeles, 24-27 October 2000.

Abstract

Many modern software development projects use application composition methods. These methods use an Integrated Computer-Aided Software Engineering (ICASE) environment and are typically small, requiring only a few people for several months. Many of these projects also utilize integrated product development teams who employ Rapid Application Development (RAD) to quickly evolve a series of product versions. For such projects, sizing the product in terms of lines of code is not practical. Instead new size measures are needed to estimate the development effort. Boehm and his colleagues have proposed a measure called Application Points [Boehm, 2000].

In August 1999 our organization had an opportunity to use the COCOMO II Application Point Model to estimate the effort for an internally funded rapid application development project. This project produced a cost tracking and invoicing system. The project was schedule constrained and used Joint Application Development (JAD) techniques to elicit the system's requirements from subject matter experts representing multiple departments within the organization (finance, accounting, contracts, project control, and project management).

During the project we collected the effort expended on various activities. In this paper we will use the collected information to compute the actual productivity of the staff and to compute the distribution of effort by activity (development, test, management, etc.) By comparing the observed productivity with the predictions of the Application Point Model, we can get an initial data point to gauge the accuracy of the model. We will also suggest some possible improvements to the Application Point Model that might make it easier to use and which might lead to increased accuracy.

1.0 Introduction

We recently completed the first phase of a rapid application development project that utilized modern database and graphical user interface technology. The project produced a two-tier application using Microsoft Access and SQL server. We collected some data on this project and thought it would be a good opportunity to investigate the utility of the application point model that is defined as part of the COCOMO II model described in [Boehm, 2000]. We also did comparisons with the original source of COCOMO's application points, namely the object points defined by Banker and coworkers [Banker, 1994]. We also made comparisons with function and feature points.

In this paper we first describe the product and the process we used to develop the product. We then present the estimates we made for the project and actual measured data. We will discuss the observed productivity and speculate on the possible implications of the results.

2.0 Project Description

The product was a project management system for government contracts. The primary function of the system is to manage information associated with task orders, including tracking costs and producing various analysis reports. The system is also used to generate a monthly invoice for the customer and a very complicated and detailed cost and status report. The system provides some access to the general project information such as points of contacts, telephone numbers, etc. Future versions of the system will provide Web access to both contractor and customer users.

The architecture that we chose for the system was standard commercial technology. The system maintains its data behind a firewall on a data server. A Web server resides outside of the firewall and provides users with Web access to various reports. For the first release of the system, the Web server was not implemented. The first release was designed strictly to support the back office functions performed by the staff of project controllers and task managers. The data server uses Microsoft SQL Server 7.0. To promote extensibility of the product for future releases and to support the validation of the data to maintain database integrity, we expended considerable effort to develop a normalized database design. The system has fairly simple interfaces with other corporate systems. Specifically, only two tables of data actually needed to be downloaded in order to do cost tracking and invoicing. Access to the functions of the system is based upon a user's assigned role and assignments to specific task orders. For the first release of the system, security was rather simplistic since the first release supported the trusted power users who work in the back office and perform the majority of the business functions. These users needed high-speed access and so were provided with Win32 applications. (One discovery we made during the project was that current Web technology does not support the high capacity and resolution that is needed to support the amounts of data contained in complicated financial reports.)

The project chose to use Joint Application Development (JAD) supported by Rapid Application Development (RAD). The reason was that several application domains were

involved. These included task order management, contract management, subcontract management, project control, finance, and other accounting functions. The schedule was ambitious (see next paragraph) and so the team chose to use Microsoft Access to prototype the user screens since they already knew how to use this technology. They also used Microsoft Excel to format reports since they also understood this technology. The project had a small dedicated team of excellent developers. Because they were dealing with new technology, however, the Chief Engineer of the project also served as a full-time expert to provide training and mentoring. (The Chief Engineer was one of the four developers.)

The project was schedule driven and the first release was required in 6.5 months. The project kickoff took place on 15 August 1999 with a target release date for Release 1 of 1 March 2000. This is an elapsed time of 200 calendar days. The project experienced some difficulties (see discussion below) and the first release actually became operational on 13 April 2000, 243 calendar days after project kickoff. This is a 22 percent schedule overrun, not too bad by typical standards. Another factor complicating this project was that it was internally funded. This made it difficult to obtain a steady source of funds. It also meant that it was difficult to get internal stakeholders to participate in some of the Joint Application Development meetings and to get them to commit to firm requirements. Because they were internal customers, it seemed that they always had more pressing fires to fight. This is another example of the usual case of being too busy fighting alligators to worry about draining the swamp.

Table 1 shows the project chronology. It shows some of the major events that occurred during the course of the project. The project team realized that a good database schema was the key to success. The first information model was completed approximately five weeks after project kickoff. This was a fairly robust model and was approximately 80 percent complete. During the course of the six-month project, however, the database schema was revised approximately eight times as new requirements were articulated by the users and as misunderstandings and miscommunications came to light and were resolved. For example, there were some redesigns of the system that occurred in January after the users had reviewed some of the prototype screens in November and December. In the team's opinion, this project had an unusually high degree of requirements volatility and growth compared to projects performed with commercial customers. Commercial contracts tend to impose penalties for excessive changes.

Table 1: Project Chronology

Date	Item
04aug99	Published "Product Overview"
05aug99	Defined "Project Goals"
15aug99	Project Kickoff
22sep99	Completed information model
23sep99	Started prototyping tables and "bare bones" screens in Access (for user feedback)
01nov99	Added Contracts/Subcontracts as stakeholders
18nov99	Major review of all screens and logic by SMEs from all functional areas
06jan00	Discovered that the billing rules supplied by the SMEs were incorrect. (This necessitated a complete redesign of 6 data tables for rate loading.)
25jan00	Completed redesign of the data tables for rate loading (now 20 tables)
31jan00	Began independent testing (Build 1)
01mar00	Target date for Release 1.0
13apr00	Release 1.0 began operation (Build 8)

3.0 Initial Estimate

Before preparing our initial project estimate, we examined two other systems developed by other organizations. These two systems performed similar functions to those of our planned system. We talked with the developers of these systems to get an idea of the number of screens, reports, and data tables that would be required. We wanted a highly modular, normalized database to facilitate the reuse and reconfiguration of the tool for other projects. This led to an increase in the number of tables needed.

Using the definitions for the application point model defined in [Boehm, 2000] we decided that the following values would be used to prepare our estimate. First, the "number and source of data tables" was assumed to be "8+". The "number of views contained" was assumed to be less than 3, giving a rating of Medium for the screens. We rated the difficulty of the reports as Medium as well. Since the application was being built from scratch, there was no existing software to be reused and so reuse was not a consideration.

Table 2 shows the initial size estimates for the application. The left-hand column shows the objects that were identified. The second column shows the number of objects. The third column is the weights taken from the application point model. The fourth column gives the computed number of application points. The initial number of reports was estimated based upon the other systems. Unfortunately, as the project progressed, considerable amounts of effort were expended on defining (and revising) the database schema and the screens used to enter and display data. This caused the team to reduce the number of reports implemented in Release 1. Only the six contractually required

reports were addressed in the first release. We'll see this in later charts. By applying the application point estimation procedure described in [Boehm, 2000], we estimated that the product's size would range from 504 to 524 new application points (NAPs).

Table 2: Initial Estimates

Object	Number	Weight	Application Points
Screens	57	2	114
Reports	70	5	350
3GL	4-6	10	40-60
		Total =	504-524

We had good developers but they were learning several new domains as previously noted. (They did know Microsoft Access, Excel and Visual Basic.) We decided that the ratings were Low to Medium for their skill level. We were using the very latest Microsoft Windows technology (SQL Server, etc.). We decided based on past experience with Microsoft products, that we would rate the ICASE maturity as Low to Nominal. Using these two values we obtained an estimated productivity of 7 to 13 new application points per person-month. Assuming 152 person-hours per person-month, we then calculated a range of values for the development effort. Using the higher productivity gave approximately 5900 person-hours. Using the lower value of productivity gave a value of approximately 11,400 person-hours

4.0 Observed Data and Revised Estimate

Figure 1 is a plot showing the various size measures for the eight builds that were produced during the course of the project leading up to the first release. (The system was operational on 13 April 2000. (Actually there was a revision to Build 8 created on 14 April to correct a minor bug.)

4.1 Actual Size

Using the data from this last build, we actually produced 141 screens, six reports and estimated that we had 14 3GL algorithms. Table 3 shows the updated estimates using the application point method based upon the actual measured numbers of screens and reports. The total is 452 new application points. Using the same two productivity values as before, we obtain an estimated effort of 5285 to 9815 person-hours.

Figure 1: Product Builds By Date

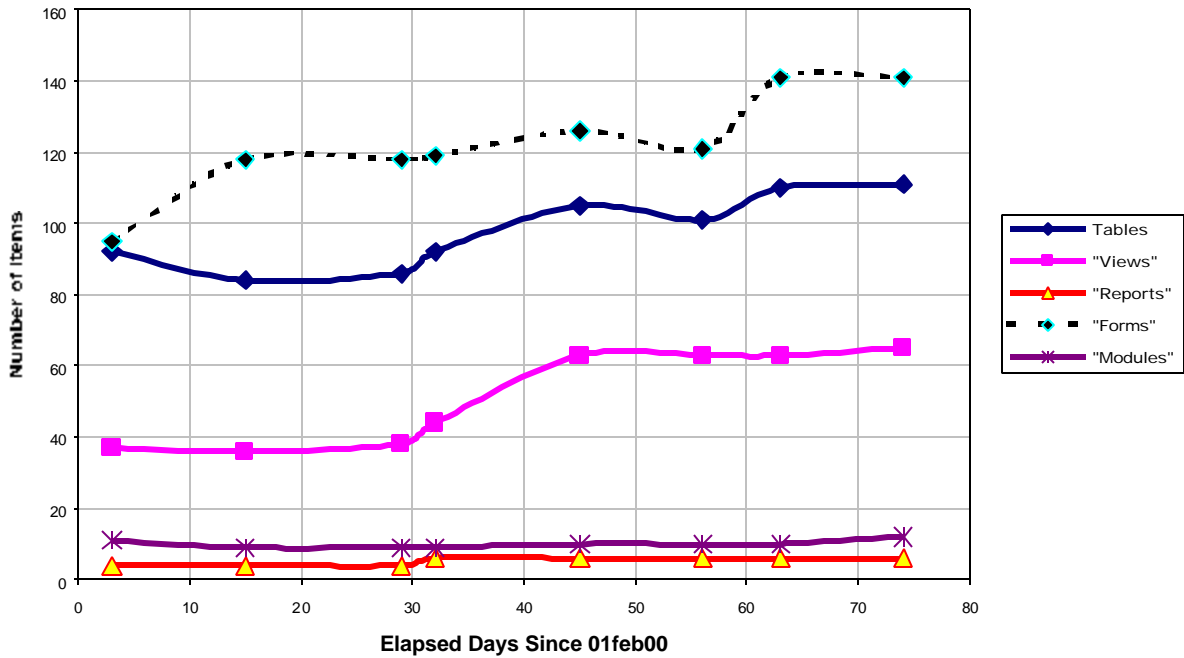


Table 3: Updated Estimates

Object	Number	Weight	Application Points
Screens	141	2	282
Reports	6	5	30
3GL	14	10	140
		Total =	452

4.2 Actual Effort Expended

Table 4 shows the effort data that we collected to prepare Release 1. This data was collected over the period 15 August 1999 through 14 April 2000. Unfortunately, we did not define a detailed Work Breakdown Structure at the start of this project and so the data is somewhat coarse. The total effort for the project was approximately 6557 person-hours. The effort expended by the subject matter experts (SMEs) amounted to approximately five percent of the total effort. If we remove the effort for the subject matter experts, the development team actually expended 6215 person-hours.

Table 4: Release 1 Effort (15Aug99 – 14 Apr00)

Activity	Total Effort (phrs)	%	%
SMEs	341.75	5.2	---
DEV (includes (CM of EDL & Sys. Admin)	4843.50	73.9	77.9
TEST	722.50	11.0	11.6
UM, Help, Training*	109.00	1.7	1.8
MGT, QA	540.00	8.2	8.7
Total	6556.75		
Total W/O SMEs	6215.00		

As shown in Table 4, we have subdivided the actual effort into development, test, manuals and training, and management and quality assurance. The development effort accounted for 78 percent of the total effort expended. Test was the next largest area, accounting for 12 percent of the effort. Management and quality assurance was about 9 percent. One reason that the user manuals, writing the help files, and preparing user training courses was so small (only two percent) was that we were only planning the training and collecting information during the time that Release 1 was being produced. The reason is that the first release was only to be used by people who were part of the test team and so already knew how to use the tool. The final course materials were scheduled for completion during a later release when “outsiders” would begin using the product. This means that the reported percentage of effort for the user manual and user training activities is unnaturally low compared to the amount for a normal complete product.

Table 5 shows our analysis of the development effort that comprised the bulk of the effort expended on the project. This was based on interviews of the developers. The left column shows the type of object produced. The second column shows the number of each object that was produced. The third column shows the approximate effort expended, based on discussions with the development team plus some analysis of existing accounting data. Particularly noteworthy here is that there are two reports which are extremely complex. The invoice required approximately 500 person-hours of effort to complete. This was large compared to the effort expended for the other reports. The monthly Cost and Performance Report (CPR) is more complex than the invoice, but was not substantially completed when Release 1 became operational. Thus, we have included it in the count of “simple reports” here.

Table 5: Analysis of Development Effort

Object	Number	Effort (phrs)
Reports (simple)	6	500-600
Reports (complex)	1	400-600
Data Base Tables	110-180	~2000
Screens	141	3100-3200
	Total =	6200

The development team expended a large amount of effort to prepare the database tables. The database tables that were counted included both normal tables (which contain actual data) plus lookup tables that are used to generate drop-down menus etc. (The number of tables was approximately evenly split between normal data and lookup tables.) Our original supposition was that lookup tables would be quite easy to develop. The development team says that this is not always the case because even simple tables may be involved in complicated logic. (See next paragraph.) The design team believes that the effort in developing these two types of tables is approximately equal. There are also 67 temporary tables (database “views”) that are constructed by joining rows and columns from multiple tables in order to create a set of data which can be used to facilitate performing complicated calculations or displaying certain reports. We have rounded the effort off to 6200 person-hours.

Our team found it difficult to gauge the difficulty of developing data tables, screens, and reports at the start of the project. Some screens are quite simple, being used for the simple input and output of loosely coupled data. A good example is an employee record (name, address, phone number, etc.). Other screens, however, appear simple but actually invoke large amounts of code (action routines, stored procedures, etc.) that perform complex processing. One example was the screen used to import data from the Corporate Management Information System. The user only specifies the type of data and the end date of the accounting period, yet the importation process that occurs involves extensive data validation and cross checking, touching many tables. It is not always easy to identify such complex processing (“3GL algorithms”) at the start of the project or, if they are identified, to gauge their scope and complexity. Similarly, some data tables may have many fields but require very simple processing and so are easy to implement. For example, the personnel data table in our system is of this type (31 fields). On the other hand, we encountered groups of several tables, each with only a few fields, that are very closely coupled and subject to complex rules designed to enforce the integrity of the data. (These are limitations on valid data values and the policies governing the multiplicity, conditionality, and fidelity of relations.) We had a group of nine tables (five were lookup tables) that defined the rules to load costs. It took the team approximately 100 person-hours to design these tables. The code that performed the loading calculations took hundreds of person-hours more. So simply counting the numbers of tables is not a good basis for estimating development effort.

Estimators need some reliable way to estimate the “size” of objects such as screens, reports, tables, interfaces, and algorithms *early* in a project. This is hard because the “size” reflects the relative degree of development difficulty and so the amount of effort needed. Unfortunately, the difficulty (effort) for a particular type of object changes depending on the particular technology and tools being used (Microsoft Access, Oracle, Visual Basic, Powerbuilder, Delphi, etc.) This makes it hard to define a basic size measure that is independent of technology. (Perhaps this dependence could be factored into the productivity. This just transfers the complexity into another parameter.) To gain some insight into possible approaches, we briefly examined function and feature points.

5.0 Comparative Analysis

We compared our observed data with results from various estimating methods.

5.1 Comparison with COCOMO Application Points

Based on the work reported here, it appears that the application point model did indeed predict a total effort that was at least in the ballpark. On the other hand, this was a bit fortuitous because we produced over twice as many screens and one-tenth the number of reports compared to our initial estimate. In addition, there was a large amount of volatility that occurred during this project, as well as a growth in the overall scope by a factor of 3 to 4.

We recommend that the application point estimation model be reexamined. To improve its applicability to projects developing large databases, the cost drivers should be extended to include data objects. In addition, the method needs to provide more precise definitions of the cost drivers and their rating scheme so that estimators can assign ratings in a consistent way. Lastly, the weights associated with the various cost drivers need to be reevaluated in light of the findings described earlier in this paper.

5.2 Comparison with Banker's Data

Table 6 compares our computed production coefficients, based on the data shown in Table 5, with values shown in Table 3 of [Banker, 1994]. The left column shows the object being produced. The second column shows our estimated production coefficients (in person-hours per item produced) based on the data shown in Table 5. The third column shows the values from Banker et al, expressed in person-hours instead of person-days. Our estimates for producing a screen are slightly higher (approximately 50 percent) than those of Banker. On the other hand, we did not specifically count 3GL modules and rule sets. We believe that the rule sets and possibly some of the 3GL routines are included in what we consider to be action routines which are constructed as part of building a screen. This would cause our production coefficients for screens to appear to be higher. Our production coefficient for producing simple reports is somewhat larger than that shown by Banker. There is no equivalent for the anomalous complex report (the invoice). It is possible that some of their effort for 3GL modules is included in our estimates of the complexity of producing the report, thereby accounting for the higher production coefficient that we observed. Note also that Banker et al did not estimate any effort for producing data tables. We will return to this point later.

Table 6: Comparison of Production Coefficients (phrs/item)

Object	Our Estimates	Banker et al (Table 3)
Screen	22-23	16
Reports (simple)	60-100	40
Reports (complex)	500	---
3 GL Modules	---	80
Rule Set	---	24
Data Tables	11-18	---

5.3 Comparison with Feature and Function Points

Table 7 shows a comparison of our data with feature point estimates. We thought that it would be interesting to look at this because function points and feature points both attempt to estimate effort associated with the design of internal and external logical files. In other words, these methods account for engineering the database schema which was ignored by the object point method described by Banker et al. In Table 7, the first column shows the name of the function point item. The second column shows the name of our items that correspond to the function point items. The third column shows the number of items of the type that we produced. The fourth column shows the weight in feature points. The next two columns show the total number of feature points, with the second column merely being an aggregation of the two data items (internal logical files and external interface files). We estimate a total of 699 feature points for the application based upon the number of items that we actually counted. Note that we increased the number of reports from 6 to 8 to account for the two complicated reports. The first was the invoice mentioned previously. The second was the complex cost and performance report. This was an attempt to compensate for the fact that these reports would take extra effort. The next-to-last column shows the percentage allocation of product size in percent based upon the estimated number of feature points. The last column shows our allocation of effort to construct those items. If we assume that the production coefficients are the same constant value for all types of objects, then the size is directly proportional to the effort. This means we can directly compare the percentages for the size in feature points to the effort values that we observed. We immediately see that the feature point method seems to greatly overestimate the amount of effort required to produce the inputs (80 percent versus 50 percent). It also appears to underestimate the effort required to generate the outputs. This apparent underestimation might be due, however, to the two complex reports. We prepared a similar table using function points and obtained essentially the same percentages.

Table 7: Comparison with Feature Points

FP Item	Our Item	# Item	Weight	Product (Raw FPs)	Aggregated	FP %	Our Effort %
Input	Forms	141	4	564	564	80	50-52
Output	Reports	8+	5	40	40	6	16-18
Query	(Included in forms)	0	4	0	0	0	0
ILF	Data Group	9	7	63	77	11	32
EIF	Data File	2	7	14			
Algorithm	Algorithms	6	3	18	18	3	0
				699	699	100	100

5.4 Discussion

Table 8 shows a comparison of the estimated effort obtained using the application point and function point models to our data. The left column shows the type of design activities that were performed. The center column shows the results for the object and application point estimation. The last column shows the result from the function and feature point estimation. The first row in Table 8 shows the effort for designing the database. The object point and application point methods totally ignore the effort associated with the data base design activity. Compared to the predictions from function and feature points, it appears that they are predicting data base design effort values which are two to three times lower than we observed. This indicates that it might be desirable to increase the weights for internal logical files and external interface files for product such as the one we developed.

Table 8: Comparison of Estimated Effort to Our Data

Type of Design Activity	Object (& Application) Points	Function/Feature Points
Data Base (data groups)	Totally Ignored	About 2-3 x low
Screen (form, display)	Comparable with our data (especially if include action routines)	About 60% too high
Reports	About 1.5-2.5 x low	About 3x low

The second row in Table 8 shows the results for producing screens. Screens consist of forms and displays. For the object point and application point methods, the estimated effort was comparable with that shown by our data. We think this is especially true if we realize that the effort that we measured for producing a screen included action routines which are equivalent to the rule sets and the 3GL modules identified in the object point and application point methods. For the function and feature points, the estimated effort

appears to be approximately 60 percent too high. This suggests that the weights used for producing inputs should probably be reduced.

The third row in Table 8 shows the results for designing reports. The object point and application point methods gave values that appeared to be lower than we observed by factors of 1.5 to 2.5. The function in feature point estimation method appeared to give values that are about three times lower than our observed values.

5.5 Some Speculations

Examining the data in the Tables 7 and 8 leads to some interesting speculations. First, regarding the object point and application point methods, they appear to focus primarily on the effort associated with the construction of the user interface. We suspect that these two methods tacitly assume that the data repository already exists. This is true for many legacy systems where the back office mainframe computer has existed forever. It is a large complicated database that no one has the time or temerity to reengineer. Often these large mainframe systems contain many flat files and are not relational. The bottom line is that there is no effort expended by the design team to design and implement the database portions of these systems because the database already exists.

As for function and feature points, it appears that the weights for inputs are too high. This might result because using modern development tools such as Access, Visual Basic, Delphi, and PowerBuilder; it is much easier and quicker to define and inputs screen and its associated action routines. That is, it is simpler today to perform the design and implementation activities than it was in the old days of COBOL programming when some of the weights were originally defined for function and feature points.

Also for function and feature points, the weights assigned for designing and implementing outputs and logical files appear to be low. Our data on reports is rather limited and may be biased by the one especially complicated report (the invoice) that we actually implemented for Release 1. We are, however, more confident in our conclusion for the effort required to engineer the database objects. We do believe that relational databases are harder to design than systems that use flat files of data. We thus think that the effort (or weight) to produce such data objects should be increased. This means that the weights should be increased in the function and feature point methods.

5.6 Caveats

There are a number of caveats which may influence our conclusions and we wish to note them here. First, our data includes the effects of significant growth in product scope, as well as substantial requirements volatility. The original estimate for the number of tables in the system was approximately 25 to 30. As our data show, we ended up with approximately 111 tables (excluding the additional 67 view tables). This is an increase by a factor of approximately 3 to 4. In addition, there was also significant volatility that occurred during the effort for the reasons discussed at the beginning of this paper. There were at least eight revisions of the database schema, most tied to the release of a new build.

The other complicating factor is that this first release produced unusual reports. First, the number of reports was low. The full complement of reports that one might expect to be defined in a system of this type were not produced due to lack of time. The first release simply produced the six reports required by the contract plus the invoice. Second, one of the reports that was produced, the invoice, was exceptionally difficult. It alone accounted for approximately 500 person-hours of effort. Because of the small number of reports and the one anomalous report it is difficult to draw any good conclusions concerning the weights to be used for reports in the object point, application point, function point, and feature point counting methods.

6.0 Acknowledgements

The author would like to thank the members of the Odyssey Design Team for sharing their data and insights on their product and development process. The team members who produced Release 1 were Boyce Beech, Robin Brasher, Trent Hunt, and Ken Powers.

References

- [Banker, 1994] "An Empirical Test of Object-Based Output Measurement Metrics in a Computer Aided Software Engineering (CASE) Environment", Rajiv D. Banker, Robert J. Kauffman and Rachna Kumar, *Journal of Management Information Systems*, Vol. 8, No. 3, Winter 1992, pp. 127-150.
- [Boehm, 2000] "Software Cost Estimation with COCOMO II", Barry W. Boehm, Bert Steece, Donald Reifer, Ray Madachy, Ellis Horowitz, Bradford K. Clark, Sunita Chulani, A. Winsor Brown, and Chris Abts, Prentice-Hall, 2000.