EXHIBIT A

26 March 2010

Expert Report of Donald J. Reifer

Oracle USA, Inc., et al. v. SAP AG, et al.

Designated Highly Confidential
Pursuant to Protective Order

Donald J. Reifer, President
Reifer Consultants, Inc.
14820 N. Dragons Breath Lane
Prescott, AZ 86305-5644

I. INTRODUCTION AND SUMMARY OF OPINIONS

I, Donald J. Reifer, submit the following report in the case Oracle USA, Inc. et al. v. SAP AG et al., Civil No. 07-CV-1658 (N.D. Cal.), on behalf of SAP AG, SAP America, Inc., and TomorrowNow, Inc. (collectively, "Defendants"). I am the founder and President of Reifer Consultants, Inc., a company that specializes in the areas of software estimation and measurement. I am also a Visiting Associate at the University of Southern California's Center for Systems and Software Engineering where I have been a member of the COCOMO cost estimating model team for over a decade.

The Defendants engaged my expert services to review and evaluate the report of Mr. Paul Pinto ("Pinto Report"). Mr. Pinto apparently was retained by Oracle USA, Inc., Oracle International Corporation, Oracle EMEA, Ltd. and Siebel Systems, Inc. (collectively "Plaintiffs") to estimate the costs associated with development of certain suites of products that they allege were accessed, copied, and used by Defendant TomorrowNow, Inc. ("TN"), through actions specified in the Fourth Amended Complaint. Mr. Pinto purported to use two methodologies to estimate those software development costs: COCOMO II and Function Point Analysis ("FPA"). Mr. David Garmus, a Certified Function Point Specialist, is addressing the FPA portions of the Pinto Report. I am addressing the COCOMO II portions of the Pinto Report. I have reached the opinions expressed in this report based on my experience and detailed review and analysis of the materials provided to me in this matter.

I reviewed and assessed the findings and conclusions expressed in the Pinto Report for accuracy, currency, correctness, reasonableness, and soundness. As part of my review and assessment, I consulted generally available materials and conducted a series of independent evaluations to try to confirm Mr. Pinto's findings and conclusions. I reviewed the Pinto Report and materials that accompanied it. I also analyzed the actual Oracle suites of products which Mr. Pinto examined when estimating software development costs, including:

- JD Edwards EnterpriseOne, Version 8.12
- PeopleSoft 8.8 Customer Resource Management (CRM)
- PeopleSoft 8.8 Enterprise Performance Management – rev 1 (EPM)
- PeopleSoft 8.4 Financial Supply Chain Management – rev 1 (FSCM)
- PeopleSoft 8.8 Human Resources Management System (HRMS)

- PeopleSoft 8.0 Student Administration

In his report, Mr. Pinto places a value on the development costs for these suites of products, as well as Siebel and JD Edwards World suites of products. As a software-valuation expert, my focus is on Mr. Pinto's use of the COCOMO II model to develop cost estimates and the parameters he uses to calibrate it. Mr. Garmus will address Mr. Pinto's use of function points in a separate report. I will review and take issue with Mr. Pinto's methodology when and as it potentially impacts or influences the proper use of the COCOMO II model.

I find Mr. Pinto's COCOMO results to be questionable based on the following five criteria that I established for the purpose of my review and assessment:

- **Accuracy** – As I will discuss later in this report, Mr. Pinto's use of an outdated version of the COCOMO model and his misinterpretations of various experts' results, including my own, lead to inaccurate conclusions particularly regarding estimates for development costs for Plaintiffs' suites of products. As an example, one of Mr. Pinto's most questionable findings was his estimates for JD Edwards World and Siebel suites of products because he uses inference instead of hard measurement data to size the programs. To generate accurate size estimates, Mr. Pinto should have sized these suites using code-counting utilities.
- **Correctness** – Mr. Pinto's estimates are clouded by numerous mathematical mistakes that he makes in simple calculations. These mistakes are highlighted in the latter part of this report. As another example of his lack of attention to correctness and detail, when Mr. Pinto used the COCOMO II model to develop estimates for PeopleSoft, he incorrectly rated "Complexity" without looking at the code to characterize it properly. For this case, he rated the complexity of the code "Very High (VH)." Such a high rating is often used to characterize the software developed for the real-time avionics system for a fighter aircraft, not business applications. Incorrect ratings like this cause me to question the correctness of all of his cost estimates.
- **Currency** – Mr. Pinto does not use the most current versions of software estimating methods and tools in his analysis. For example, he uses an outdated and unsupported version of the COCOMO II model (COCOMO II.1997) to develop his software cost estimates. Use of this version of the model generates estimates that are much higher than

those produced by the current release, COCOMO II.2000. His use of the wrong version of the COCOMO II model led him into drawing incorrect conclusions. The University of Southern California (USC) website and the textbook used to describe COCOMO II [BOE01], of which I am an author, both support the selection and use of the current COCOMO II.2000 version of the model.

- **Reasonableness** – Many of his results do not seem reasonable. For example, to price the resulting estimates, Mr. Pinto develops a labor rate based on a labor mix that assumes sixty percent management and support personnel to forty percent technical workers. Besides biasing his labor rates high, this mix leaves in question whether the software work contemplated could have feasibly been completed in a reasonable time period with such a high management overhead.
- **Soundness** – The manner in which Mr. Pinto used the methodologies, COCOMO II and FPA, can also be questioned. For example, he emphasizes the need to use two different approaches to develop independent estimates primarily so one can be used to cross-check the results of the other. However, he then elects to build both of his estimates on the same size basis, source lines of code. By doing this, any claims of independence evaporate. He also builds his cost estimates on the case that time is of the essence and that the software to be developed must be completed in two years. However, his own cost model runs show that the development could not be completed in two years and that the most feasible schedule possible with his settings for the model is on the order of five to eight years. If he had used the COCOMO II properly, he would have determined that his proposed two-year schedule is unachievable.

For the purpose of assessing the accuracy, correctness, currency, reasonableness, and soundness of Mr. Pinto's conclusions, I performed a COCOMO II analysis to size various suites of products that Mr. Pinto analyzed. I ran the COCOMO II.2000 version instead of the outdated COCOMO II.1997 version, increased (corrected) the staff hours assumed per staff month of effort from 144 to a more-appropriate 152, used the more-appropriate labor rates that I developed that assumed personnel costs for staff working in India were lower, updated (corrected) the code count estimates, and set the scale and cost drivers properly. The range of the costs that I developed after these corrections were made – as a check on his analysis and conclusions – was about eighteen to twenty-five percent of Mr. Pinto's depending on what goals for the schedule were established even when I used his questionable size estimates for two of the suites of products in question. It is also important to note that these model runs, which are summarized in Table 1, verified that the two-year schedule estimates that Mr. Pinto claimed was needed for developing the new versions for the four Oracle suites of products under consideration was not possible even under the most optimistic conditions. These results make me question the accuracy, correctness, currency, reasonableness, and soundness of the findings and conclusions that Mr. Pinto presents in his report.

Estimate	Cost ($)			Duration (Months)		
	OPT	LIKELY	PESS	OPT	LIKELY	PESS
Pinto- COCOMO.1997 + 144 Hours/Staff-Month	?	$1,477.3M	?	?	?	?
Reifer – COCOMO.2000 + 152 Hours/Staff-Month	$963.0M	$1,203.7M	$1,504.6M	91.9	98.4	105.3
Reifer – above + new parameter settings	$573.8M	$717.5M	$894.3M	35.4	37.8	40.4
Reifer – above + new labor rate + correct size	$302.8M	$378.6M	$471.6M	35.4	37.8	40.4
Reifer – above + optimal development schedule	$211.8M	$264.7M	$330.8M	47.2	50.4	53.8

Table 1: Summary and Comparison of COCOMO Model Runs

<u>Legend</u>

OPT – optimistic	LIKELY – most likely	PESS – pessimistic

II. QUALIFICATIONS OF EXPERT WITNESS

a. <u>Background</u>

A copy of my curriculum vitae is provided in Appendix A. I have worked in the software field for over forty years. My early career was spent in aerospace companies as software engineer and manager. I started as a software engineer where I developed software and its documentation. I then progressed into management leading teams that were charged with developing software for projects. I moved to other companies to take on more responsibility. I managed large software projects of national importance like the defense portion of the space transportation system (Space Shuttle) as part of these assignments.

During the next part of my career, I worked as a consultant to organizations interested in using metrics and models for managing large software projects. My measurement vision was

Code Category	% Design Modified	% Code Modified	% Integration Modified	Relative Percent Effort
New – all original	N/A	N/A	N/A	100%
Adapted – existing software that is changed or modified	0 to 100%	0 to 100%	0 to 100%	0 to 100%
NOMINAL	40%	40%	60%	46%
Reused – existing software that is used as-is/calls counted	0%	0%	0 to 100%	30% (at most)
COTS – requires glue code wrappers that are counted	Count the wrapper code as new and add effort needed to test wrapped COTS package			

Table 2: Counting Conventions for Equivalent Source Lines of Code

In other words, the size for a software package is most often never all new source lines of code. It is smaller because of these reuse considerations.

Mr. Pinto assumed that all of the suites of products under consideration would have to be redeveloped as new code. This assumption is just not true for most of the applications that I have been associated with. Instead, Mr. Pinto should have grouped the software into new, modified, reused, generated and COTS categories and used the Software Engineering Institute (SEI) counting standards that he referenced to address these different types of software [ORCLX-PIN-000017] as these different groupings of software were used to develop the suites of products in a manner similar to that shown in Figure 1 to calculate source lines of code.

The net results of Mr. Pinto's failure to use such practices as he grouped the software are that his estimates of source lines of code are highly suspect and his size estimates seem biased high. This in turn biases Mr. Pinto's COCOMO II estimates high.

- **Mr. Pinto's Step 2: Count the Number of Source Lines of Code**

I next tried to acquire copies of the specialized counting utilities that Mr. Pinto developed to tally source lines of code. My goal was to replicate his analysis as I tried to understand how he counted source lines of code assuming that all of the code was considered new code. While Mr. Pinto infers that calculating source lines of code is simple [Pinto Report, p. 15], the SEI manual that he relied on to provide counting conventions refutes his claim. Counting lines of code is difficult and requires more powerful tools than Mr. Pinto developed to deal with the many nuances that he acknowledges may be present in the code that the counters must handle.

Why Mr. Pinto developed his own source lines of code counters puzzled me. Powerful tools, frequently used by industry, that perform the task exist and can be acquired for free from sites like those at the University of Southern California (see the tools section of http://sunset.usc.edu). When investigating Mr. Pinto's counters more closely, one sees that while they count the code, they do not do so in a manner that fully complies with the standards and conventions defined by the SEI. Many of the nuances that Mr. Pinto acknowledges that are present like embedded constants in the C programming language were just overlooked by his utilities. [Pinto Report, pp. 15 – 16]

To understand the impact of these counts, my assistant and I developed a set of utilities that replicated the code for Mr. Pinto's counters as described in ORCLX-PIN-000067 for the C programming language (including headers) running on a PC running Windows Vista. I then had my assistant download the C source code for a piece of public domain software for a flight simulator called FlightGear (http://www.flightgear.org). I next had him count the code for the main routine using the Pinto utilities and the freely available USC developed language code counters called Unified CodeCount (UCC) (see the download section of http://sunset.usc.edu). The results of this counting experiment are provided in Table 3 [see SAP-DJR-000003 for summary]. These differences lead me to question both the accuracy and correctness of Mr. Pinto's counts and his customized counting utilities.

SLOC Counting Tool	Total Number Lines	Total Blank Lines	Total Comment Lines	Total Physical SLOC	Total Logical SLOC	Total Number Files
Pinto Code Counter[1]	58,739	9,687	11,941	37,111	30,215	199
USC Code Counter	58,752	9,687	12,086	36,979	27,585	199
DIFFERENCE	- 13	0	- 145[2]	132	- 2,630	0

Table 3: Results of Code Counting Experiment using FlightGear

Notes

[1] This is a counter that follows Mr. Pinto's parsing rules as described in ORCLX-PIN-000066 and replicated his code as described in ORCLX-PIN-000067.

[2] The difference in comment lines is primarily the number of embedded constants in the count.

The main difference in Logical Source Lines of Code ("SLOC") calculation occurred due to how embedded comments were counted by Mr. Pinto's utility software. There was also some confusion over how Mr. Pinto counted compiler directives and data declarations.

To verify whether this error consistently existed in the JD Edwards code, my assistant and I developed a second set of counters for the Java J2EE programming language following the parsing rules described in ORCLX-PIN-000076 and replicating the code described in ORCLX-PIN-000077 to run on my Windows/Vista PC platform. We then extracted three C and two Java J2EE routines from the JD Edwards EnterpriseOne code library and ran them through our versions of the Pinto utilities and USC UCC counter. The results, which are summarized in Table 4, verify that an error of nine and one half percent exists for all of the code inspected [see SAP-DJR-000004 for summary including file list].

Definition Checklist for Source Statements Counts

Definition name: __Logical Source Statements___ Date:________________

________________(basic definition)__________Originator:_COCOMO II____

Measurement unit:	Physical source lines						
	Logical source statements			√			
Statement type	Definition	√	Data Array			Includes	Excludes
When a line or statement contains more than one type, classify it as the type with the highest precedence.							
1 Executable			Order of precedence		1	√	
2 Non-executable							
3 Declarations					2	√	
4 Compiler directives					3	√	
5 Comments							
6 On their own lines					4		√
7 On lines with source code					5		√
8 Banners and non-blank spacers					6		√
9 Blank (empty) comments					7		√
10 Blank lines					8		√
11							
12							
How produced	Definition	√	Data array			Includes	Excludes
1 Programmed						√	
2 Generated with source code generators							√
3 Converted with automated translators						√	
4 Copied or reused without change						√	
5 Modified						√	
6 Removed							√
7							
8							
Origin	Definition	√	Data array			Includes	Excludes
1 New work: no prior existence						√	
2 Prior work: taken or adapted from							
3 A previous version, build, or release						√	
4 Commercial, off-the-shelf software (COTS), other than libraries							√
5 Government furnished software (GFS), other than reuse libraries							√
6 Another product							√
7 A vendor-supplied language support library (unmodified)							√
8 A vendor-supplied operating system or utility (unmodified)							√
9 A local or modified language support library or operating system							√
10 Other commercial library							√
11 A reuse library (software designed for reuse)						√	
12 Other software component or library						√	
13							
14							

Figure 1: SLOC Definition Checklist

SLOC Counting Tool	Language	Total Number Lines	Total Blank Lines	Total Comment Lines	Total Physical SLOC	Total Logical SLOC	Total Number Files
Pinto	C[1]	779	10	230	539	528	3
	Java[1]	156	8	43	105	95	2
USC	C	779	10	245[2]	524	478	3
	Java	156	8	55[2]	104	86	2

Table 4: Results of Code Counting Experiment using Five Routines from the JD Edwards EnterpriseOne Software Applications Package

<u>Notes</u>
[1] These are utilities that count C and Java code following the rules and replicating the code as Mr. Pinto's describes in ORCLX-PIN-000066, PIN-000067, PIN-000076 and PIN-000077.
[2] The difference in comment lines is the number of embedded constants in the count.

While seemingly small, a nine and one half percent error in counts is significant when working with numbers of this magnitude. For the C and Java programming language code in the JD Edwards EnterpriseOne suite, this error means that the code count in Mr. Pinto's Table 5 should be reduced by 738,605 source lines of code (using 7,774,791 SLOC as the base count). I will address this error in Section VII of this report.

Because of the impact, I went a step further. As summarized in Table 5, I counted a larger sample of the C code in the JD Edwards EnterpriseOne suite to assess whether this error propagated throughout it. As noted in the summary, the error for C code including the headers was 14.5% when I compared the USC versus Pinto counts [see SAP-DJR-000005 for summary]. I use these results to correct the C and Java sizing source lines of code counts later in this report when I develop an independent cost estimate for this suite, which I develop in order to point out the various, substantial errors in Mr. Pinto's analysis and conclusions.

Language	No. Programs	USC Count	Pinto Count[1]	% Difference
Header	836	153,172	153,205	0.02
C	728	779,139	937,620	16.9
	TOTAL	932,311	1,090,825	14.5

Table 5: Results of Code Counting for JD Edwards EnterpriseOne Package

<u>Notes</u>
[1] These are C language counting utilities that replicate Mr. Pinto's code and follow the rules provided in ORCLX-PIN-000066 and ORCLX-PIN-000067.

d. <u>Imprecise Counts of Source Lines of Code</u>

My assistant and I spent a great deal of effort trying to validate Mr. Pinto's logical SLOC counts because they represent the core basis of his estimating methodology. Mr. Pinto developed a number of specialized utility software routines to develop his line counts. These programs counted lines in the C, COBOL, Java J2EE, SQC, SQL and other programming languages (see ORCLX-PIN-000066 to ORCLX-PIN-000085) supposedly using guidelines developed by the SEI (see ORCLX-PIN-000017) to guide their development.

Mr. Pinto then used these code counters to develop his sizing estimates for the JD Edwards EnterpriseOne and PeopleSoft suites of products, provided in his report as ORCLX-PIN-000024 to ORCLX-PIN-000062. As previously mentioned, I used the FlightGear open-source software program as a benchmark to determine whether or not these size estimates adhered to the SEI conventions. My assistant and I ran the USC–developed utility (which is heavily used by industry) and the utilities that we developed replicating Mr. Pinto's code counters side-by-side and encountered a nine and one half percent error for C programming language code stemming from issues primarily in how Mr. Pinto's counter counted embedded comments. We then developed a counter for the Java programming language. Afterwards, we extracted five routines (three C and two Java) from the JD Edwards EnterpriseOne suite of products and counted them using both counters. The counter runs verified that the nine and one half percent error found earlier in FlightGear was present in the C and Java code used by the JD Edwards EnterpriseOne suite of products. We next ran the counters for over one million lines of C code and verified that the Pinto counts were high by a factor of fourteen and one half percent. This is a significant error. [See SAP-DJR-000005]

e. <u>Double Counting Documentation Costs</u>

Mr. Pinto builds an elaborate model for estimating documentation costs which he states are not addressed by the COCOMO II cost model estimates. That is incorrect. Referring to Dr. Barry Boehm [BOE01], the costs for most normal technical and user documentation generated as a product of each the development tasks encompassed in the scope of the software cost model is fully addressed in the COCOMO II estimates. Normal user documentation includes user, operator, reference, and maintenance manuals. It is important to note that these conclusions were developed by analyzing data from hundreds of software projects completed by hundreds of

I also ran the model with the new labor rates that I developed and made the size correction of nine and one half percent for Java and fourteen and one half percent for C code that I noted earlier in the report. As expected, these changes (corrections) lowered the software development costs for this suite of products considerably.

i. Mr. Pinto's Results

Mr. Pinto's assumptions for use with the COCOMO II.1997 version of the model are summarized along with his results in Table 21. ("Corrected" refers to the corrections of the mathematical errors he made in his calculations.) The left column notes the correct expansion of 15,491 staff-months by 144 hours per staff-month. The bottom row shows a second error by Mr. Pinto when he multiplied 2,230,746 hours by $145.72 per hour.

JD Edwards EnterpriseOne	
Source Lines of Code	7,775K

JD Edwards EnterpriseOne					
Scale Factors	PREC	FLEX	RESL	TEAM	PMAT
Rating	H	H	H	H	H

Cost Driver Ratings									
RELY	DATA	DOCU	CPLX	RUSE	TIME	STOR	PVOL	ACAP	PCAP
H	H	H	H	H	N	N	N	VH	VH
PCON	APEX	PLEX	LTEX	TOOL	SITE	SCED			
VH	VH	VH	VH	H	H	H			

JD Edwards EnterpriseOne (With mathematical errors noted)		Corrected
Estimated Effort (Staff-Months)[1]	15,491	
Estimated Duration	?	
Person Hours[2]	2,230,746	2,230,704
Average Blended Rate	$145.72	
Estimated Cost	$325,061,334	$325,058,187
Corrected	$325,064,307	

Table 21: Summary of Factors Used and Results Achieved with COCOMO II.1997 By Mr. Pinto with Mathematical Errors Highlighted

Notes
[1] Using the unsupported and outdated COCOMO II.1997.
[2] Using 144 staff hours/staff-month of effort.

ii. Corrected Results

My corrected estimate uses Mr. Pinto's original assumptions for size, labor rates and model parameter ratings. I elected to use the COCOMO II.2000 version of the cost model because it is

Section VI, Paragraph f, the results of rerunning the model with correct values are summarized in Table 27.

JDE EnterpriseOne (With Model Parameters and Rates Changed)	
Estimated Effort (Staff-Months)[1]	8,377.4
Estimated Duration	36.5
Person Hours[2]	1,273,365[3]
Average Blended Rate	$91.69
Estimated Cost	$116,754,837[3]

Table 27: JD Edwards EnterpriseOne Estimate Using New Labor Rates

Notes
[1] Using the COCOMO II.2000 model instead of the COCOMO II.1997 assumed by Mr. Pinto.
[2] Using 152 hours/staff-month instead of the 144 hours/staff-month assumed by Mr. Pinto.
[3] Rounded

The estimated results are $208,306,497 less than Mr. Pinto developed for his “most likely” case when I apply these new labor rates to develop the estimate.

As previously reported, I found that the size estimates for the C programming language was off by fourteen and one half percent and the Java programming language by nine and one half percent. If we reduce the size of the estimate from 7,774,791 SLOC to 6,690,878 SLOC to account for these mistakes and run the model one more time with the new size, corrected cost/scale drivers, corrected hours/staff-month, and new rates, we get the results summarized in Table 28. I reduced the C size by 1,001,394 and the Java size by 82,519 source lines of code based on the size counts Mr. Pinto provides in Table 5 of his report (on page 17).

JD Edwards EnterpriseOne			
	Optimistic	**Most Likely**	**Pessimistic**
Estimated Effort (Staff-Months)[1]	5,768.3	7,210.4	9,012.9
Estimated Duration	32.6	34.9	37.3
Person Hours[2]	876,782[3]	1,095,981[3]	1,369,961[3]
Average Blended Rate	$91.69	$91.69	$91.69
Estimated Cost	$80,392.142[3]	$100,490,498[3]	$125,611,724[3]

Table 28: JD Edwards EnterpriseOne Estimate using Corrected Size Estimate for C Code

Notes
[1] Using the COCOMO II.2000 model instead of the COCOMO II.1997 assumed by Mr. Pinto.
[2] Using 152 hours/staff-month instead of the 144 hours/staff-month assumed by Mr. Pinto.
[3] Rounded

Siebel (With Model Parameters and Rates Changed)			
	Optimistic	Most Likely	Pessimistic
Estimated Effort (Staff-Months)[1]	2,695.7	3,369.6	4,212.0
Estimated Duration	40.9	43.8	46.9
Person Hours[2]	409,746	512,179	640,224
Average Blended Rate	$103.15	$103.15	$103.15
Estimated Cost	$42,265,300	$52,831,264	$66,039,106

Table 52A: Siebel Estimate with Optimal Schedule

Notes

[1] Using the COCOMO II.2000 model instead of the COCOMO II.1997 assumed by Mr. Pinto.
[2] Using 152 hours/staff-month instead of the 144 hours/staff-month assumed by Mr. Pinto.
[3] Rounded

The "most likely" estimated results for this new case are $ 204,474,876 less than Mr. Pinto developed for his "most likely" case. This, too, is a considerable difference in the estimates. Needless to say, these results, too, confirm that Mr. Pinto's results are not accurate, correct, reasonable, or sound.

VIII. RESULTS

Based on my experience in the field of parametric cost estimation and the use of the COCOMO II.2000 cost estimation model, and after reviewing the materials noted in this case, including those contained in the Pinto Report and the materials provided with it, and developing independent estimates, my conclusions are as follows:

a. Summary of Analysis

I am highly confident that the software development costs associated with the JD Edwards EnterpriseOne Version 8.12 suite of products should be considerably less than that valued by Mr. Pinto. My independent estimate, as described in this report, would be in the range of between $80.4 and $125.6 million based on my assumptions, corrections, updated size estimates, and labor rate calculations. The most likely cost would be $100.5 million. Moreover, when the schedule is not artificially compressed (as proposed by Mr. Pinto), the most likely cost for this suite of products would be $70.3 million.

I am highly confident that the software development costs associated the PeopleSoft Version 8.X suite of products should be considerably less than that valued by Mr. Pinto. My independent estimate, as described in this report, would be in the range of between $125.9 and $195.1 million based on my assumptions, corrections, and labor rate calculations. The most likely cost would be $157.4 million. Moreover, when the schedule is not artificially compressed (as proposed by Mr. Pinto), the most likely cost for this suite of products would be $110.0 million.

I am highly confident that the software development costs associated the JD Edwards World suite of products should be considerably less than that valued by Mr. Pinto. My independent estimate, as described in this report, would be in the range of between $36.1 and $56.5 million based on my assumptions, corrections, and labor rate calculations. The most likely cost would be $45.2 million. Moreover, when the schedule is not artificially compressed (as proposed by Mr. Pinto), the most likely cost for this suite of products would be $31.6 million.

I am highly confident that the software development costs associated with the Siebel suite of products should be considerably less than that valued by Mr. Pinto. My independent estimate, as described in this report, would be in the range of between $60.4 and $94.4 million based on my assumptions, corrections, and labor rate calculations. The most likely cost would be $75.5 million. Moreover, when the schedule is not artificially compressed (as proposed by Mr. Pinto), the most likely cost for this suite of products would be $52.8 million.

In total, I am highly confident that the software development costs associated with the four suites of products cited, as described in this report, based on my assumptions and corrected labor rate and size calculations could be accomplished in about three years for between $302.8 and $471.6 million with a with a most likely cost of $378.6 million. The most likely cost when the schedule is not artificially compressed (as proposed by Mr. Pinto) is $264.7 million.

The estimates developed during my analysis are summarized in the Table below.

Based on COCOMO II.2000 model runs using Mr. Pinto's numbers, I have further concluded that his cost estimates are high and that the 24 month schedule he assumed is infeasible. The quickest possible delivery schedule is 35.4 month assuming the most optimistic assumptions possible with the COCOMO II.2000 model. The quickest and most optimistic possible schedule (without undue compression, as proposed by Mr. Pinto) is 47.2 months.