EXHIBIT 4

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

TEXT REMOVED - NOT RELEVANT TO MOTION

II. QUALIFICATIONS OF EXPERT WITNESS

a. Background

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 embraced by my many clients and I helped found a consulting company dedicated to using metrics and models to improve the practice of software management. To implement this vision, I developed a family of software cost estimating models called SoftCost which I marketed and supported worldwide. My consulting company, Reifer Consultants, Inc., grew as we helped commercial clients put metrics and measurement to work for management.

In my mid-career, I was asked to serve our nation by taking a leadership position in the Defense Department. As part of this assignment, I deactivated my consulting business to rid myself of any conflicts of interest. During my two years in government, I led software projects of national importance like the Ada programming language and the software reuse initiative. I also participated in the Corporate Information Management initiative which was aimed at streamlining the way the Defense Department handled business applications software like that used for payroll processing, information management, and logistics.

I reactivated my consulting company after completing my government assignment and refocused my efforts on helping clients implement empirical methods and models. When asked by Dr. Barry Boehm for help, I joined the COCOMO team at the University of Southern California. I also worked with other clients on topics of interest which ranged in scope from pursuing research in malware detection and code obfuscation to developing new software cost models like that which I devised for estimating web costs using a concept for sizing based on function points named web objects [REI02].

I have considerable expertise in the fields of software cost estimating and productivity improvement. As part of the management consulting practice which I have led for over thirty years, I have lectured, consulted and taught the topics of software cost estimating and business case formulation worldwide.

From 1985 to 1993, I developed and marketed the SoftCost family of software estimating models worldwide. These estimating models were used by more than twenty organizations, including the U. S. Government and commercial organizations to develop software cost and schedule estimates. I consulted and taught software cost estimating techniques to my clients during the period and developed mathematical formulas aimed at improving the usability and accuracy of the cost models. Based on this work, I received the prestigious Frieman Award from the International Society of Parametric Analysts (ISPA), a professional society dedicated to cost estimation, for my contributions to the field of software parametric estimating in 1991.

From 1996 to present, I have served as a Visiting Associate at the Center for Systems and Software Engineering at the University of Southern California (USC/CSSE) where I am a senior member of the COCOMO II project team. COCOMO II is the world's most widely used cost estimating model. From 1996 to 2000, I helped to calibrate the COCOMO II model and its cost and scale drivers. I also interfaced with the organizations supplying the data and checked it to ensure its correctness and usability. In 2000, I co-authored a book on COCOMO with Dr. Boehm and other members of the USC team entitled "Software Cost Estimation with COCOMO II." From 2000 until the present, I have been working as a member of the COCOMO II team.

I have prepared independent software effort and duration estimates for clients in commercial industry and government. These have been used as benchmarks against which competitive bids have been compared and their reasonableness ascertained. I also teach a course on software cost estimating with COCOMO II.2000 for selected clients. The next offering of this class will be in April 2010 for Tecolote Research, a government support contractor specializing in software cost estimating. I continue to be asked to perform consulting services and teach professionals how to properly calibrate and use software cost models, especially COCOMO II.2000.

I am also leading a joint Army and Air Force Study that is investigating ways to improve how software maintenance is estimated and budgeted by both their headquarters staffs and software life cycle support centers. These centers employ over ten thousand programmers who are involved in updating, repairing, optimizing, and improving software being used for both Information Technology and weapons system applications ranging from ERP (Enterprise Resource Planning) systems used in-house to military aircraft and missile systems.

TEXT REMOVED - NOT RELEVANT TO MOTION

TEXT REMOVED - NOT RELEVANT TO MOTION

c. <u>Use of Empirical Data</u>

Whenever possible, I also tried to use empirical data to substantiate my ratings of COCOMO II model parameters, and its scale and cost drivers. For example, I looked at the source code referenced in the Pinto Report to try to fairly judge its size and complexity. When reviewing the source code, I used the COCOMO model's five part complexity driver rating scheme to make my findings and determinations [see Table 2.19 in [BOE01]]. I next reviewed the source code referenced by Mr. Pinto to determine whether there was a high degree of reuse present in the software as claimed. I looked for code in the source code that called reusable modules to make this determination and finding [REI04]. Finally, I reviewed Mr. Pinto's source lines of code counters and parsing rules. I had my assistant from USC develop software to assess how well these counters worked using an open source program called FlightGear which served as an independent benchmark. Mr. Pinto's size estimates were higher than expected because he did not fully comply with the code counting rules which he references from the Software Engineering Institute [ORCLX-PIN-000017]. Because size drives cost in the COCOMO model, inaccurate size estimates lead to incorrect effort and duration estimates.

During my career, I have found it useful to rely on "hard data," whenever it was available, to make my determinations and findings. While it may be easy for skilled professionals to debate expert opinions, it is much more difficult to argue the facts when discovered through such detailed analysis of the code.

TEXT REMOVED - NOT RELEVANT TO MOTION

TEXT REMOVED - NOT RELEVANT TO MOTION

- **Mr. Pinto's Ten-Step Estimating Approach**

Because of its potential impact on the factors used in the COCOMO II model, I reviewed Mr. Pinto's ten-step estimating approach. My comments are as follows:

TEXT REMOVED - NOT RELEVANT TO MOTION

TEXT REMOVED - NOT RELEVANT TO MOTION

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

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].

TEXT REMOVED - NOT RELEVANT TO MOTION

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

Notes

[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.

TEXT REMOVED - NOT RELEVANT TO MOTION

COCOMO II.2000 is a source lines of code model because this was the size metric used for its calibration using its 161 data points. The reason for this is that the COCOMO II.2000 model takes inputs in either function points or source lines of code because model users wanted this feature. The reason it uses the SPQR backfiring standards referenced in the Pinto Report was that this reference was the most readily available report on the topic that was accessible for that purpose when we were developing the model. That said, the backfiring and conversion proposed by Mr. Pinto was not, in my opinion, the correct step, and should not have been done.

TEXT REMOVED - NOT RELEVANT TO MOTION

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]

TEXT REMOVED - NOT RELEVANT TO MOTION

X. OPTION TO REVISE

I reserve the right to modify and/or supplement this report and/or opinions set forth herein if additional damages or other rulings are made by the Court and/or additional evidence becomes available, including through my continued analysis of information provided by the parties to this litigation.

I, Donald J. Reifer, having conducted the aforementioned analysis and having authored this report, confirm that the opinions contained herein represent a fair and impartial analysis of the facts presented to me in the materials considered and provided.

Date: _26 March 2010_

[illegible]

Donald J. Reifer

APPENDIX A –VITAE OF DONALD J. REIFER

CONTACT INFORMATION:

Donald J. Reifer

Reifer Consultants, Inc.

14820 N. Dragons Breath Lane

Prescott, AZ 86305

Office: 928-237-9060 Cell: 310-922-7043

Email: don@reifer.com Web site: www.reifer.com

CURRENT POSITION:

President and Chief Technical Officer - Reifer Consultants, Inc.

CAREER SUMMARY:

- Innovator, entrepreneur, businessman and internationally recognized leader in the fields of software engineering and management.
- Consultant who helped clients to implement value-based software project management concepts.
- Expert in the area of software estimation with years of experience in the field.
- Entrepreneur who built a software company from scratch into a respected force in the industry.
- Senior Executive Official with the Department of Defense in charge of managing major Service and Agency-wide information management and software technology initiatives.
- Recognized expert in the area of large-scale software project management with forty plus years of experience in managing the development of large, real-time, software-intensive systems.
- Consultant who helped clients to introduce new technology, process improvement per the Software Engineering Institute’s (SEI) Capability Maturity Model (CMM) and other innovations into large organizations.
- Senior Manager responsible for multi-million dollar satellite contracts while with TRW and for pulling together the Department of Defense's efforts on the Space Shuttle project.
- Recognized expert in the field of software reverse engineering with a decade of experience in inventing network security/software protection technologies for the Department of Defense.