EXHIBIT E

Notes re Response to Rebuttal report of Donald J. Reifer
Paul Pinto - May 19, 2010 - *Oracle v. SAP*

1. Introduction and summary response

Throughout Mr. Reifer's 90-page rebuttal report, he accepts and adopts the ten-step approach that I have outlined. Then, instead of performing an unbiased assessment in response, he "manipulates" the COCOMO Model to drive toward the lowest estimate possible. Mr. Reifer has allowed his foregone conclusions to lead his analysis effort, as opposed to really conducting a fair assessment of the costs associated with developing the products in question. While Mr. Reifer appears to go through what is seemingly a series of logical gyrations, each step he takes is flawed.

2. My Estimating Team

See notes on response to Garmus' rebuttal.

3. Counting Source Lines of Code

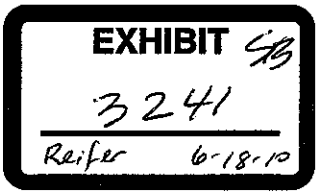
Mr. Reifer asserts that the code-counting utilities that I produced did not accurately count the number of logical source lines of code. In an attempt to prove his theory: Mr. Reifer custom-built a utility that purportedly replicates the code-counting utility that was used by my team (See number 5 below). I am perplexed as to why Mr. Reifer would construct a code-counting utility that replicates the functionality of my code-counting utility, when he was provided with all of the code-counting utilities (source code and executables) that were used by my team during my analysis.

Mr. Reifer then goes on to test his custom-built code-counting utility (purported replica of my code-counting utility) against a market-available code-counting utility which he obtained at no cost from the University of Southern California. He performs his test on a sampling of C code (58,739 physical lines) that is part of a flight simulator program, called FlightGear. As a result of this test, Mr. Reifer cites that his replica code-counting utility identified 30,215 logical source lines of code, while the free USC code-counting utility identified 27,585 logical source lines of code, a variance of 2,630. Mr. Reifer acknowledges this variance as representing a 9.5% difference (2,630/27,585).

Mr. Reifer then custom-built a second set of purported replica code-counting utilities in order to test the Java code, again electing not to use the actual utilities that were used by my team. He then tests both replica utilities (C and Java) on a hand-selected (see note #4 below) set of 5 programs (3-C programs, and 2-Java programs. Within his carefully hand-selected 935 lines of code (5-programs), the replica code-counting utilities identified 623 logical source lines of code, while the free USC code-counting utility identified 564 logical source lines of code. Mr. Reifer again acknowledges this variance as a 9.5% difference (59/623), and claims, on page 20 of his report, that the results "...verify that an error of nine and one half percent exists for all code inspected...".

Aside from the issues associated with purportedly replicating my code-counters as opposed to using the actual code-counting utilities used by my team, aside from introducing a non-germane application (FlightGear) as a supposedly valid benchmark, and aside from hand-selecting the C and Java programs to be tested: Mr. Reifer has manipulated mathematics in order to show the alignment of his results between the two tests. In his first test, against the FlightGear code base, Mr. Reifer acknowledged the USC source line count as the denominator (27,585) in his variance calculation, but when he performed the same variance calculation on the result of his second test, against the 5 hand-selected JD Edwards programs, he acknowledges the replica source line count as the denominator (623).

Mr. Reifer then goes on to conduct a third test, on a larger hand-selected set of JD Edwards EnterpriseOne programs (1,564 programs). As a result of his test, Mr. Reifer purports that the replica code-counting utilities identified 1,090,825 logical source lines of code, while the free USC code-counting utility identified 932,311 logical source lines of code. Mr. Reifer asserts this variance as a 14.5% difference (158,516/1,090,825), indicating a change back to the original variance formula.



When consistent mathematics is applied to Mr. Reifers results, the following data points are obtained:

Table #2 (PIN000108)

Test Bed	Variance
FlightGear	8.7% (2,630 / 30,215 SLOC from replica counter)
5 hand-selected C programs	9.5% (59 / 623 SLOC from replica counter)
1,564 hand-selected C and Java programs	14.5% (158,516 / 1,090,825 SLOC from replica counter)

Based on the development of his purported replica code-counters and hand-selecting the desired test beds, Mr. Reifer had the ability to manipulate the results of this analysis to justify his conclusions. As such, the there is no evidence that the code-counting utilities developed by my team were not accurate. As a matter of fact, my actual code-counting utilities were never tested.

4. Approach to sampling

Mr. Reifer elected to hand-select a series of source code programs to serve as the bed for testing the validity of his purported replica code-counting utilities. Based on the results of his test, Mr. Reifer then draws a number of conclusions about the accuracy of the replica code-counting utilities that he tested. In order for the results of his analysis to be valid, Mr. Reifer should have conducted a sampling of source code files that is in alignment with industry-accepted norms for random sampling, as opposed to the hand-selecting method that he chose.

5. Custom-built code-counting utilities

I elected to custom-build all of the code-counting utilities to meet the logical source line counting requirements associated with the entire code base that was being analyzed. In my early analysis, I determined that there was no single-source of code-counting utilities available in the market that could count **all** of the required code bases (e.g. C, Java, COBOL/400, SQC, SQR, DMS, SQL, RPT, MDL, and Peoplecode) in a consistent and repeatable manner, and that would provide me the ability to tune the counter for the specific code base. As opposed to using a mix of free code-counters and custom code-counters, I elected to custom-build a consistent set of code-counters that were specifically designed around the code bases in question.

As part of the decision to custom-build my code-counters, we reviewed a number of free code-counters including the ones offered by USC. While these free code-counters appeared to perform as advertised, they did not provide us with the ability to configure or tune them for use on a specific code base. As such, I deemed it more appropriate to custom-built code-counters that were specifically designed to analyze the nuances of each of the germane code sets. All of which were provided to Defendants, as both source code and executable files.

During the development of the custom code-counters, my team tested the automated results against a series of hand-counts, in order to verify the validity of our utilities. As a final test, my team ran all three code-counting utilities (e.g. my actual code-counting utility, Mr. Reifer's replica of my code-counting utility, and the free USC code-counting utility) side-by-side, against the FlightGear source code. The following table displays the results of my side-by-side test of the cited utilities:

Table #3: PIN000108

Code Counting Utilities	Source of Count	Type of Count	File Types		Total
			C	H	
Reifer's Custom-built Replicas	PintoCounter_Output.txt (from Reifer report)	File Count	29	139	168
	PintoCounter_Output.txt (from Reifer report)	Logical SLOC Count	2694	6917	9611
Free from USC	C_CPP_outfile.dat (from Reifer report)	File Count	29	139	168
	C_CPP_outfile.dat (from Reifer report)	Logical SLOC Count	2382	6914	9296
Pinto's Actual Counters	Counting performed by Pinto's team	File Count	29	139	168
	Counting performed by Pinto's team	Logical SLOC Count	2535	6720	9255

The following conclusions can be drawn from this analysis:

- Mr. Reifer's Custom-built purported Replica code-counters do not produce the same results as Pinto's Actual code-counters. Mr. Reifer's Replicas over counted the SLOC for C and H by 356 SLOC or 3.8% ((9,611-9,255)/9,255)
- Mr. Reifer's Custom-built Replica code-counters counted more logical source lines of code than did the Free USC code-counters. Mr. Reifer's Replicas over counted the SLOC for C and H by 315 SLOC or 3.4% ((9,611-9,296)/9,296)
- The Free USC code-counters counted slightly more logical source lines of code than did Pinto's Actual code-counters. The Free USC code-counter over counted the SLOC for C and H by 41 SLOC or less than one half of one percent ((9,296-9,255)/9,255)
- While there is a difference between the three counts performed by the three different code counting tools, my actual code counter produced the most conservative aggregate count of logical source lines of code. My counters have been in service since 2000 and throughout that time have been tuned to specifically address this code base. I'm confident that they yield a reasonable count of source lines of code, as demonstrated by the results of this analysis.

6. Localization and Translation

The products in question are supported in 21-languages, as identified in a number of configuration files that were reviewed as part of my analysis. During my team's review of the Oracle manuals as contained in the provided ISO files, we found numerous references to the 21 languages. These localizations and translation requirements are also referenced on the Oracle website.

Mr. Reifer attempts to portray the need to localize the application and translate the documentation into 21 languages as being within the scope of standard product development life-cycle, and simultaneously as being outside of the scope of what is reasonable for this analysis effort. He also misrepresents that I have double-counted the cost of documentation by not acknowledging it as being accounted for within the COCOMO II cost model.

With regard to localization: At no time throughout his analysis does Mr. Reifer acknowledge localization of the application modules as being anything more than documentation translation. In reality, localization of any commercial application is a significant undertaking which requires changes to screen layouts and additional functionality. While the additional functionality, associated with localization, can be accounted for by counting the lines of source code, there is a distinct effort that goes into rewriting a screen to accommodate double-byte sized Japanese characters. This is part of the localization effort that is not accounted for in the standard COCOMO II cost model.

With regard to Documentation translation: Mr. Reifer once again asserts that I did not address the documentation translation costs as being included in my COCOMO II cost model, when indeed I did include them, but excluded the costs associated with translating the documentation into 21 languages. On page 25 of his report, Mr. Reifer writes:

"Mr. Pinto's numbers are skewed on the high side, since it appears to me that the costs quoted by Mr. Pinto are primarily for manual translation…He does not consider using many powerful automated translation tools like SYSTRAN that can reduce the time to generate a page of user documentation from days to seconds and cost from $15 per page, as quoted in the Pinto Report, to about 30 cents a page."

Mr. Reifer incorrectly assumes that the costs I have cited do not include the use of translation tools, when in fact they do. He continues to assert that it is plausible to buy a tool and perform self-translation, which may appear attractive in theory, but in-practice is totally unviable. During my combined 7-year tenure with Epicor and NIIT, I was directly involved in engaging translation specialists to perform the task of translating User documentation into a multitude of languages. At no time, either then or today, would it be viable to do as Mr. Reifer has proposed for all the languages in question.

7. Effort Distribution

In regard to my distribution of effort across the product development life-cycle, Mr. Reifer writes "...Mr. Pinto misused the model when he allocated the estimated effort using percentages that were unsubstantiated…". In evidence of his assertion, Mr. Reifer cites, in table7 on page 37 of his report, a series of other effort distribution models.

In analyzing at the effort distributions offered by Mr. Reifer's other models, while not including my assessment of effort distribution, the following observation can be made:

Table #4 (PIN000108)

Lifecycle Phase	Range of Reifer 's Other Distribution Percentages (low to high)	Pinto Distribution Percentage	Comment
Plan and Specify	12% - 22% (10% range)	25%	My distribution is outside of the range, based on my appreciation of the level of effort that is required to conceive, vision, and define a commercial software product
Design and Build	45.8%-67% (21.2% range)	55%	My distribution is well within the range
Test	8.5% - 19% (10.5% range)	12%	My distribution is well within the range
Deploy	2% - 11% (9% range)	8%	My distribution is well within the range

The range of distributions, and the fact that the models do not even agree on a consistent set of development lifecycle phases (e.g. Plan, Specify, Plan/Specify, Design, Build, Design/Build, Test, Deploy…), demonstrates that there is no empirically right or wrong distribution of effort. Mr. Reifer's analysis proves that the Estimator is required to evaluate the specific circumstances associated with the estimating effort, and distribute the effort accordingly, which is exactly what I did in my analysis.

Mr. Reifer also assert that I have inappropriately allocated a higher percentage of work to more costly roles, where on page 37 and 38 of Mr. Reifer's report, he writes that "…his (Mr. Pinto's) distribution of effort to roles are not correct because they tend to foster excessive management and support overhead. Mr. Pinto's proportion of managers and support personnel to workers is sixty to forty percent. "

In support of this assertion Mr. Reifer provides a taxonomy, in table 8 on page 39 of his report, where he compares my distribution of effort to his supposed normal distribution of effort. In order to support his conclusion, Mr. Reifer attributes my roles to a generic set of roles that he manufactured (i.e. Managers, Leads, Support, Analysts, Technical Analysts, Developers, Testers…). He then performs an act of association through word-matching,

whereby if the role has the word "Lead" in its title, then it must be an overhead position. In going through this obtuse exercise, Mr. Reifer ignores the fact that I organized my roles in direct alignment with the phases of the lifecycle to which they contribute. By virtue of this ubiquitous hierarchy I have clearly allocated 15% of the effort within the development lifecycle to management activities (Program Manager:3%, Project Manager:9%, and Quality Manager:3%) and another 11% to analysis activities (Product Manager:3%, and Business Analyst:8%). In total, I have allocated 26% of the development effort to non-code developing activities, with the remaining 74% of effort dedicated to developing, testing, and deploying the code.

Once again, Mr. Reifer manipulates the data in way that is not consistent with my 25-years of industry experience.

8. Rates

Mr. Reifer writes, on page 41 of his report, that "Mr. Pinto's hourly rates for personnel working in India seem high compared to current salary benchmarks for that country…". In evidence of this, Mr. Reifer offers an alternate set of rates, in table 10 on page 42 of his report, where he has added a burden factor to the base salary costs in order to derive his newly proposed hourly rates. In doing so, Mr. Reifer has made the assumption that SAP TN would have built-out an offshore facility located in India, as opposed to hiring an already established India-based Consulting company, as I have asserted.

Mr. Reifer's assumption is incorrect because it ignores the need for a significant upfront investment in recruiting, hiring, and retaining employees; designing, developing, and implementing the application and technical infrastructure that is required to support a professional software development shop; and establishing and institutionalizing the required processes and level of process maturity. Fielding a captive offshore center of this nature, assuming SAP TN had the required knowledge and capability to do so, would cost millions of dollars to establish, and more importantly would require at least 2-years before the new organization could reach a level of maturity (CMM level 3) to where it could even consider having the necessary experience to undertake a development effort of this magnitude. These points are discussed in detail by Vijay Machigad, in his whitepaper titled: Offshore Models for Engineering Product Development, and by NeoIT in its whitepaper titled: Captive Offshore Development Centers (available online).

Mr. Reifer also assumes the low hourly rates associated with running a captive, offshore development center, but he also repeatedly assumes that the People/Team-centric characteristics of the COCOMO model should be calibrated to reflect a high-performing, mature, and cohesive team. Throughout his analysis, he applies his lower labor rates associated with a start-up facility, while citing the following scaling factors:

Table #5: PIN000108

COCOMO Category	I.D.	Reifer's Selection	Description
Team Cohesion	TEAM	"Very High" for 3 of the 4 products	Strong synchronization between all stakeholders
Process Maturity	PMAT	"High" for all 4 of the products	CMM Level 3
Analyst Capabilities	ACAP	"Very High" for 3 of the 4 products	90th percentile based on efficiency
Programmer Capabilities	PCAP	"Very High" for 3 of the 4 products	90th percentile based on efficiency
Personnel Continuity	PCON	"High" for 3 of the 4 products	6% turn-over per year
Application Experience	APEX	"Very High" for 3 of the 4 products	6-years of experience
Platform Experience	PLEX	"Very High" for 3 of the 4 products	6-years of experience
Language and Toolset Experience	LTEX	"Very High" for 3 of the 4 products	6-years of experience
Multisite Development	SITE	"High" for 3 of the 4 products	Entire team located in the same city

In order to be consistent within his own estimating model, Mr. Reifer would either need to accept the use of an experienced offshore external service provider, and the commensurate hourly rates, or he would need to

recalibrate the aforementioned COCOMO scaling factors to reflect the use of a less-stable, less-experienced, multi-national team.

The reasonable approach is to engage a proven, CMM level 5 assessed, offshore consulting company that specializes in developing commercial software applications. The rates that I cited in my report are in direct alignment with engaging an external service provider of this caliber, as are the scaling factors cited in my COCOMO II cost model.

In establishing the rates that I used in my analysis, I relied on 2009 market data, which represents what was publically available. However, based on my industry experience, the 2004-2005 rates were actually higher than the rates I have cited here.

- ✓ Vijay Machigad, in his whitepaper titled: Offshore Models for Engineering Product Development,
- ✓ NeoIT in its whitepaper titled: Captive Offshore Development Centers.

9. Commercial Software

On page 17 of his report, Mr. Reifer writes "…most organizations do not develop source code from scratch. They try to reuse legacy and Commercial Off-The-Shelf (COTS) software to reduce the volume of work involved. They use application generators to develop the code wherever possible because they accomplish this task automatically without human intervention."

While his statement is correct for a subset of application development shops, it is incorrect for the vast majority of commercial software development shops. In my 25-years of experience, with two roles as a senior executive within large-scale commercial software develop shops, I have never used a code generator to develop commercial software, simply because the resulting code will always perform sub-optimally, and will then be required to be rewritten by hand. This is common knowledge to any expert who has experience in commercial software development.

In these few statements Mr. Reifer again displays a lack of appreciation for the difference between software application development (for internal use) and software product development (for commercial use).

10. Code Reuse

On page 18 of his report, Mr. Reifer writes, "…the size for a software package is most often never all new source lines of code. It is smaller because of the reuse consideration. Mr. Pinto assumes 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…". Once again, Mr. Reifer displays his lack of understanding of the commercial software products in question, and the concepts of Releases.

If I were to have acknowledged all prior releases of a product as being within the scope of my estimating effort, then I would have been obligated to identify New Code, Reused Code, and Modified Code as separate components. By assuming this approach, I have limited the potential for double-counting, while maintaining an extremely conservative posture, by assuming that all work efforts that were performed on prior releases would be accounted for by analyzing the code base of the most current release.

11. COCOMO II.1997 vs. COCOMO II.2000

Mr. Reifer criticizes my use of the COCOMO II.1997 model as opposed to using the COCOMO II.2000 model. I selected the COCOMO II.1997 model because I have the highest degree of confidence in applying it when product source code is available to be used as the primary input. Its accuracy has been proven to me in real-world

scenarios where the conditions were very similar to the estimating scenario in question: a commercial software product, access to the source code, and using a multi-shore software development team.

The difference between the 1997 model (parent) and the 2000 model (derivative) is that the 1997 model was calibrated based on the use of 83 data points, while the 2000 model was calibrated based the use of 161 data points. While I have successfully applied both the 1997 and the 2000 models, I elected to use the 1997 model because it is proven to be highly tuned for this task. As such, there was no valid reason for me to abandon its use for a newer version of the same model that was extended to accommodate the "early-design" capabilities, especially when the newer version offers no additional benefit except for the fact that it is "new."

12. Mathematical Errors

Mr. Reifer asserts, no less than 30-times throughout the 90-page body of his report, that I have performed simple mathematical errors. See e.g. Mr. Reifer Report, page 30: "When I reviewed the Pinto Report, the first thing that struck me was the large number of mathematical errors that he made. This immediately led me to question the correctness and validity of Mr. Pinto's results. While some errors may occur in a multi-page report, those made consistently in the area of simple addition and multiplication are not acceptable. For example, Mr. Pinto multiplies incorrectly again and again when multiplying the effort in staff-months generated by the COCOMO II.1997 cost model and his hourly expansion factor of 144 hours/staff-month of effort. As another example, Mr. Pinto consistently gets the wrong answer when he multiplies the number of labor hours that result using the aforementioned expansion factor by his blended labor rates."

Upon further review of the mathematical calculation that I conducted, I found no errors. All mathematical calculations were performed within a spreadsheet, with the germane formulas embedded in the appropriate cells. Mr. Reifer's observations can be directly attributed to his performing the math on the rounded (truncated) numbers as portrayed in the body of my report, as opposed to reviewing the detailed spreadsheet analysis which was conducted at a greater level of mathematical significance (see ORCLX-PIN-000065, table 25, table 26, table 27, and table 28). If Mr. Reifer reviewed the provided spreadsheet, all of the detailed mathematical calculations would have been clear to him, as well the rounding that occurred.

✓ Pinto Software Estimating Worksheet, Tabs 25, 26, 27, and 28 (PIN000065)

13. Conservative Estimates

When referring to a number of components of my estimates, Mr. Reifer writes: "seems high" or "appears high to me," while never acknowledging that I have intentionally limited the overall scope of my analysis in order to maintain an unquestionably conservative posture, for example:

- I elected to only estimate one version of each of the products in question, as opposed to estimating each prior version as a separate instance. As such, I have assumed that the effort that was invested in creating all prior releases, is subsumed by estimating the development costs associated with the most current release. This represents an extremely conservative posture, and ensures that no double-counting between versions will occur.

- I elected to not to quantify the avoided costs associated with Defendants unlicensed use of the Oracle Database products.

- I elected to only estimate the product suites as they were delivered from Oracle. As such, I have intentionally ignored all derivative works that were produced and reused by SAP TN, while using the product code as its base.

- I elected to only estimate the direct cost associated with the labor force that would be required to perform the development work. I did not include any estimates associated with capital outlays (the

purchase of application development hardware, multiple operating systems, testing tools), or staff travel and expenses.

- I elected not to acknowledge any "throw-away" code that would most certainly have been developed throughout the years of the product lifecycle. As per the estimating guidelines, I could have acknowledged a percentage of the code as a BRAK value, and adjusted the effective size of the code-base upward, but I did not.

- I elected not to assess any costs associated with the additional value added to SAP TN that could be attributed to having instant access to the software, or through avoiding litigation.

- I elected not to assess any additional risk factors associated with the development effort. As per estimating guidelines, I should have developed a risk score which would have increased the estimated cost of development somewhere between 10% and 30%.

I am confident that Experts who are experienced with the development of commercial software would agree that the points I have cited here would most certainly support the conservative nature of my estimates.

14. Productive Hours

On pages 23 and 24 of his report, Mr. Reifer writes " He (Pinto) used the value 144 productive staff hours per staff month in his calculations, but this value comes from a reference that was developed in Europe [ORCLX-PIN 000013]. Instead, he should have increased the number of hours to 152 staff hours per staff-month because this is the expansion ratio used by the COCOMO II and other cost models for labor in the United States."

If I were to have used the COCOMO II recommended hours, my estimated cost of development would have been 5.56% higher ((152-144)/144). While I acknowledge 152 hours per month as a recommended guideline; in order to maintain a conservative posture, I elected to use 144 hours per month, which was derived as follows:

Table #6: PIN000108

Category	Value
Number of weeks in calendar year	52 weeks
Number of available work hours in a week	40 hours
Number of available work hours in a calendar year	2,080 hours
Average number of vacation days per Full-time staff member	120 hours (15 days)
Number of legal holidays within a calendar year	80 hours (10 days)
Percentage of time spent doing unproductive work (internal meetings, administrative functions, training…)	150.4 hours (8% of 1,880 hours) per year
Total number of productive hours within a calendar year	1,729.6 hours
Total number of productive hours within a calendar month	144.13 hours

As I have demonstrated, the 152 hours per month that is recommended in the COCOMO II Guide, is simply unrealistic when applied to real world costing constraints. This is an example of an estimating guideline that must be evaluated against reality before it is applied.

On page 94, of Manual-Software Estimating Methods, the author writes"…the observed number of hours actually worked in a month can run from less than 120 to more than 170. Because the actual number of hours varies from project to project, company to company, and country to country, it is best to replace the generic rate…" [PIN000098]

On page 15, of the Cocomo II Model Definition Manual, the author writes: "COCOMO II treats the number of person-hours per person-month, PH/PM, as an adjustable factor with a nominal value of 152 hours per Person-

Month. This number excludes time typically devoted to holidays, vacations, and weekend time off… If you use a different value of PH/PM–say, 160 instead of 152–COCOMO II adjusts the PM estimate accordingly (in this case, reducing by about 5%). This reduced PM will result in a smaller estimate of development schedule." Also, on page 67, the author writes: "Some implementations, such as USC COCOMO II, provide an adjustable parameter for the number of person-hours per person-month. Thus, if you enter 137 (10% less) for this parameter, USC COCOMO II will increase your person-month estimate by 10%, and calculate an appropriately longer development schedule."

Within the offshore I.T. Consulting arena, Infosys Technologies Limited (INFY) is acknowledged as the industry's top performer in the area of consultant utilization. Between the years of 2001 to 2009, the highest level of utilization reported by INFY, can be found on page 35 of Infosys' 2004 annual report, where the utilization rate is cited as being 82.2%. This benchmark indicates that the average Consultant billed 1,710 hours per year, out of the available 2,080 hours, which correlates to 142.5 hours per month.

- ✓ On page 94, of Manual-Software Estimating Methods (PIN000098)
- ✓ On page 15, of the Cocomo II Model Definition Manual
- ✓ On page 35 of Infosys' 2004 annual report

15. Project Duration

Mr. Reifer contends that I have unnaturally constrained the project duration to 2-years, as seen on page 51 of his report where he writes "Mr. Pinto states that he was trying to achieve a two year schedule when he developed his estimates. But, he rated the SCED driver 'High' for three of the four suites of products under consideration." He goes on to cite "A 'High' rating for SCED means that Mr. Pinto believed that the schedule needed to be extended, not shortened."

While I use the 2-year duration as a reference point, I intentionally did not constrain the project schedule (SCED) in my COCOMO analysis, as Mr. Reifer acknowledges. In my report, I actually point out that it would be difficult to achieve this development effort within a time 2-year frame, which Mr. Reifer confirms on page 51 of his report, where he cites "…a properly run COCOMO analysis shows that the development could not be completed in two-years." I acknowledge this 2-year frame, because it represents a meaningful period in which SAP TN had to capitalize on the business opportunity created by Oracles acquisition of PeopleSoft and the resulting customer opportunity.

The fact that development effort could not be achieved within a meaningful timeframe (2-years), further supports the need for SAP TN to gain immediate access to the software.

16. Estimation by Inference

On page 54 of his report, Mr. Reifer writes: "Mr. Pinto used the process that he developed for estimating costs for only two of four suites of products in question, JD Edwards EnterpriseOne and PeopleSoft. To develop estimates for the remaining two suites of products, JD Edwards World and Siebel, he makes a leap of faith assuming that these two suites of products are similar to the former two or subsets of them."

Based on a number of factors, I elected to employ "analogy-based estimating" as an alternate technique for sizing the JD Edwards World and Seibel products. In Murali Chemuturi's whitepaper, Analogy based Software Estimation, he writes: "Analogy Based Software Estimation is based on the principle that actual values achieved within the organization in an earlier and similar project are better indicators and predict the future project performance much better than an estimate developed afresh from scratch." [PIN000110] He also goes on to describe the 13 project characteristics that should be used when comparing one project to another, for the purpose of identifying a suitable project to serve as the basis for an analogy-based sizing effort.

Due the enormity of the effort required to perform a detailed estimating analysis of JD Edwards EnterpriseOne and PeopleSoft, and their ability to likeness to JD Edwards World and Siebel, respectively, permitted me to acknowledge my estimates as valid analogs.

- ✓ Murali Chemuturi's whitepaper, Analogy based Software Estimation (PIN000110)

17. Reasonableness

Throughout Mr. Reifer's report, he asserts that the estimates I have developed are unreasonably high. For a host of reasons already discussed, I consider Mr. Reifer's estimates as being unreasonably low. After receiving his report I reviewed the historic R&D spends for each of the products in question, contained in Mr. Meyer's report in paragraphs 144 – 148, and 282 – 287, which confirmed the reasonableness of my estimates. In the world of estimating, it is customary to consider all historic data when developing an estimate. In my case I was careful not to look to these numbers so as not bias my assessment, but I would have expected Mr. Reifer and Mr. Garmus to have researched these publically available numbers.

18. Mr. Reifer's use of COCOMO as a shield

The COCOMO estimating methodologies provided through USC, as with all reputed estimating methodologies, are intended to be used as a guideline, as opposed to being applied in a rote manner, as Mr. Reifer asserts. All estimating methodologies are acknowledged as providing a baseline of meaningful thoughts that must then be evaluated for their specific applicability to the situation at hand. No estimating methodology can be viewed, or appropriately implemented as a rigid recipe, without applying the Estimator's knowledge of the environment that is to be estimated, as discussed on page 68 through 70 of the COCOMO II Model Definition Manual User Manual.

There are literally hundreds of valid estimating techniques and approaches that an informed Estimator would evaluate when determining the most appropriate techniques and approach to use on a specific estimating job. e.g. Magne Jørgensen, of the Simula Research Laboratory, in his white paper entitled: Forecasting of Software Development Work Effort: Evidence on Expert Judgment and Formal Models [PIN000113] concludes that it is well-advised to use Estimating models in combination with Experts that understand the domain that is being estimated:

"The models' ability to weight variables more correctly, to reduce biases, and to produce consistent estimates may consequently have been insufficient to compensate for the low quality of the models and their inability to use all of the relevant contextual information. The software development community is, consequently, still in a position where the evidence supports neither a replacement of models with expert judgment, nor a replacement of expert judgment with models. If, as suggested in MacDonell and Shepperd (2003), there is a high degree of independence between estimates based on common effort estimation models and expert judgment, and it is difficult to devise rules for selecting the most accurate estimation method, the solution seems to be to use a combination of models and experts."

Another criticism leveled by Mr. Reifer is that my complexity rating for PeopleSoft product is inappropriate (see pages 64-65) under COCOMO protocols. However, in the business world, customers view ERP software as "mission critical." This software has real-time and major implications on key aspects of company operations. My rating is therefore appropriate.

- ✓ page 68 through 70 of the COCOMO II Model Definition Manual User Manual
- ✓ Magne Jørgensen, of the Simula Research Laboratory, in his white paper entitled: Forecasting of Software Development Work Effort: Evidence on Expert Judgment and Formal Models (PIN000113)

19. The ten-step estimating process

At no time throughout the body of Mr. Reifer's 90-page report does he contradict the ten-step estimating process as being invalid. While he takes opposition to the numbers and setting I use within every step, at no time does he

ever assert that I skipped a step, or took a leap of faith in getting from one step to another. This is relevant because it is in direct opposition to the conclusions drawn by Mr. Garmus', in his parallel rebuttal report, where he writes: "That Approach and that Analysis are not recognized steps in any FPA. Consequently, any function point value that he (Pinto) achieved through his ten-step analysis is totally contrived and without any cognizable basis. The Function Point values he (Pinto) reported are completely fabricated, and his (Pinto's) analysis and process would not be recognized by any expert in the field as being a legitimate…"

Mr. Reifer actually dedicates 11 pages of his 90-page report, page 17 through 27, to reviewing each of the ten-steps. Then he goes on to repeatedly use the ten-step approach in developing his alternate effort estimates.

20. Reifer's Assumption about the number of Test Cases

On page 52 of his report , Mr. Reifer writes "For database systems like those involved in the suites of products in question, a "Nominal (N)" seems the most appropriate rating. A Nominal setting is more appropriate because Mr. Pinto pointed to no evidence that large amounts of test data were produced by Oracle to support product testing activities."

Based on my industry experience with commercial software applications of this size and complexity would require a significant amount of testing and associate test data, in order to certify that the application is ready for commercial distribution and usage. Based on the below cited metric, and the estimated number of test cases, it is fair to say the a "Nominal" rating is understated.

On page 11 of the whitepaper titled: How Software Estimation Tools Work, the author cites that the number of test cases, associated with the development of commercial software, can be estimated by multiplying the number of function points by a factor of 1.65. [PIN000117] For the JDE and PeopleSoft products, which were estimated in terms of function points, the following table provides an estimated number of test cases:

Table #7: PIN000108

Product	Estimated Number of Function Points	Multiplier	Estimated Number of Test Cases
JDE EnterpriseOne Version 8.12	133,139	1.65	219,679
PeopleSoft Version 8.X	399,838	1.65	659,732

21. Reifer's Approach to Re-Estimating

While Mr. Reifer goes through what appears to be a logical progression from one model to the next, on page 88 of his report, he makes a number of invalid assumptions along the way. I will address each of these iterations individually, but to provide an appropriate level of context, Mr. Reifer's models can be summarized as follow:

1. Correct Mathematical Errors – which did not exist. Accounts for no reduction in costs.
2. Lift and shift of 1997 calibration to 2000 – which was inappropriate and inaccurate. Accounts for 18.5% of the reduction.
3. Application of COCOMO II.2000 – Inconsistent calibration of the model actually contradict himself. Accounts for 32.9% of the reduction.
4. Application of new Labor Rates – Used improper mix of resources, and used internal staff costs as opposed to Consulting costs. Accounts for 18.0% of the reduction.
5. Changes the size of code estimates – Unjustified changes based on improper analysis. Accounts for 4.9% of the reduction.
6. Optimal scheduling – totally unrealistic to meet the needs of the business. Accounts for 7.7% of the reduction.

22. Reifer's First Estimate

The first estimate undertaken by Mr. Reifer was to correct his perception of mathematical errors, as he asserts on page 88 of his report: "The first estimate was that developed by Mr. Pinto using the COCOMO II.1997 model with only changes made needed to correct mathematical errors." In reality, there were no mathematical errors, just a lack of attention to detail on the part of Mr. Reifer.

23. Reifer's Second Estimate

The second estimate undertaken by Mr. Reifer was to convert my COCOMO II.1997 model to his preferred COCOMO II.2000 model, as he writes on page 88 of his report: "The second estimate was developed using the COCOMO II.2000 with the number of hours assumed per staff-month of effort increased from 144 hours assumed by Mr. Pinto to the 152 hours used by the COCOMO team…"

In this second estimate, Mr. Reifer migrates the scale factors from my COCOMO II.1997 model to his COCOMO II.2000 model. While at first this may appear to be a legitimate migration technique, but in reality it is not a valid approach. While the COCOMO II.2000 model is a derivative of the COCOMO 11.1997 model, both of which make use of essentially the same descriptions for each of the scaling factors, the models differ in the effort multipliers that are associated with each of the scaling factors.

The follow table shows the COCOMO II.1997 calibrated values for the scale factors and effort multipliers, as cited on page 76 of The COCOMO II Model Definition Manual, which was provided as one of Mr. Reifer's references:

Table #8: PIN000108

Baseline Effort Constants: **A** = 2.45; **B** = 1.01 Baseline Schedule Constants: **C** = 2.66; **D** = 0.33							
Driver	**Symbol**	**VL**	**L**	**N**	**H**	**VH**	**XH**
PREC	SF1	4.05	3.24	2.43	1.62	0.81	0.00
FLEX	SF2	6.07	4.86	3.64	2.43	1.21	0.00
RESL	SF3	4.22	3.38	2.53	1.69	0.84	0.00
TEAM	SF4	4.94	3.95	2.97	1.98	0.99	0.00
PMAT	SF5	4.54	3.64	2.73	1.82	0.91	0.00
RELY	EM1	0.75	0.88	1.00	1.15	1.39	
DATA	EM2		0.93	1.00	1.09	1.19	
RUSE	EM3		0.91	1.00	1.14	1.29	1.49
DOCU	EM4	0.89	0.95	1.00	1.06	1.13	
CPLX	EM5	0.75	0.88	1.00	1.15	1.30	1.66
TIME	EM6			1.00	1.11	1.31	1.67
STOR	EM7			1.00	1.06	1.21	1.57
PVOL	EM8		0.87	1.00	1.15	1.30	
ACAP	EM9	1.50	1.22	1.00	0.83	0.67	
PCAP	EM10	1.37	1.16	1.00	0.87	0.74	
PCON	EM11	1.24	1.10	1.00	0.92	0.84	
APEX	EM12	1.22	1.10	1.00	0.89	0.81	
PLEX	EM13	1.25	1.12	1.00	0.88	0.81	
LTEX	EM14	1.22	1.10	1.00	0.91	0.84	
TOOL	EM15	1.24	1.12	1.00	0.86	0.72	
SITE	EM16	1.25	1.10	1.00	0.92	0.84	0.78
SCED	EM17	1.29	1.10	1.00	1.00	1.00	

The follow table shows the COCOMO II.2000 calibrated values for the scale factors and effort multipliers, as cited on page 75 of The COCOMO II Model Definition Manual, which was provided as one of Mr. Reifer's references:

Table #9: PIN000108

Baseline Effort Constants: **A** = 2.94; **B** = 0.91 Baseline Schedule Constants: **C** = 3.67; **D** = 0.28							
Driver	**Symbol**	**VL**	**L**	**N**	**H**	**VH**	**XH**
PREC	SF1	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	SF2	5.07	4.05	3.04	2.03	1.01	0.00
RESL	SF3	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	SF4	5.48	4.38	3.29	2.19	1.10	0.00
PMAT	SF5	7.80	6.24	4.68	3.12	1.56	0.00
RELY	EM1	0.82	0.92	1.00	1.10	1.26	
DATA	EM2		0.90	1.00	1.14	1.28	
CPLX	EM3	0.73	0.87	1.00	1.17	1.34	1.74
RUSE	EM4		0.95	1.00	1.07	1.15	1.24
DOCU	EM5	0.81	0.91	1.00	1.11	1.23	
TIME	EM6			1.00	1.11	1.29	1.63
STOR	EM7			1.00	1.05	1.17	1.46
PVOL	EM8		0.87	1.00	1.15	1.30	
ACAP	EM9	1.42	1.19	1.00	0.85	0.71	
PCAP	EM10	1.34	1.15	1.00	0.88	0.76	
PCON	EM11	1.29	1.12	1.00	0.90	0.81	
APEX	EM12	1.22	1.10	1.00	0.88	0.81	
PLEX	EM13	1.19	1.09	1.00	0.91	0.85	
LTEX	EM14	1.20	1.09	1.00	0.91	0.84	
TOOL	EM15	1.17	1.09	1.00	0.90	0.78	
SITE	EM16	1.22	1.09	1.00	0.93	0.86	0.80
SCED	EM17	1.43	1.14	1.00	1.00	1.00	

As is apparent, by even a cursory review of the effort multipliers in the two above tables, that it is not appropriate to "lift" a scale rating from the 1997 model and simply "shift" it to the same rating within the 2000 model. An intelligent Estimator understands that you must review the effort multipliers for reasonability, as part of the evaluation process.

COCOMO II provides the Estimator with a set of reasonability guidelines, that are embedded within the scaling factors and effort multiplier, as described on page 16 of the COCOMO II Model Definition Manual, where the author writes: "…The exponent E in Equation 11 is an aggregation of five *scale factors* (SF) that account for the relative economies or diseconomies of scale encountered for software projects of different sizes [Banker et al. 1994]. If E < 1.0, the project exhibits economies of scale. If the product's size is doubled, the project effort is less than doubled. The project's productivity increases as the product size is increased. Some project economies of scale can be achieved via project-specific tools (e.g., simulations, testbeds), but in general these are difficult to achieve. For small projects, fixed start-up costs such as tool tailoring and setup of standards and administrative reports are often a source of economies of scale. If E = 1.0, the economies and diseconomies of scale are in balance. This linear model is often used for cost estimation of small projects.

If E > 1.0, the project exhibits diseconomies of scale. This is generally because of two main factors: growth of interpersonal communications overhead and growth of large-system integration overhead. Larger projects will have more personnel, and thus more interpersonal communications paths consuming overhead. Integrating a small product as part of a larger product requires not only the effort to develop the small product, but also the additional overhead effort to design, maintain, integrate, and test its interfaces with the remainder of the product. See [Banker et al. 1994] for a further discussion of software economies and diseconomies of scale."

In applying this ubiquitous guideline as a reasonability check, all of the COCOMO models that I constructed resulted in E being greater 1.0, which indicates that the projects exhibit diseconomies of scale associated with the amount of interpersonal communication and overhead that would be required. In direct opposition to my models, the "lift-and-shift" models constructed by Mr. Reifer resulted in E being less than 1.0., which indicates that the projects exhibit economies of scale, which is an unreasonable assertion for these projects.

There is no valid reason for Mr. Reifer to insist on using the COCOMO II. 2000 model as opposed to using the COCOMO.II 1997 analysis that I provided. If Mr. Reifer felt compelled to only work within the confines of the COCOMO II.2000 model, as opposed to working with the provided COCOMO II.1997 cost analysis that I provided, then he should have performed an "intelligent" migration of my scaling factors, as opposed to the simple "lift-and-shift" approach which he employed.

page 76 of The COCOMO II Model Definition Manual
page 75 of The COCOMO II Model Definition Manual
page 16, of the COCOMO II Model Definition Manual

24. Reifer's Third Estimate

The third estimate undertaken by Mr. Reifer was to recalibrate the scale and cost drivers for the COCOMO II.2000 model, as he writes on page 88 of his report: "…I updated the ratings for the scale and cost drivers used by the model employing my knowledge of the estimation package to calibrate it more closely with the realities of the situation."

Mr. Reifer change of the scaling and cost drivers in the COCOMO II.2000 model is suspect at best. The following table provides a summary of Mr. Reifer's ratings for each of the four software products in question:

Table #10: PIN000108

	PeopleSoft	JD Edwards EnterpriseOne	JD Edwards World	Seibel
PREC	VH	VH	VH	H
FLEX	VH	H	H	H
RESL	VH	VH	VH	H
TEAM	VH	VH	VH	H
PMAT	H	H	H	H
RELY	H	H	H	H
DATA	H	N	N	N
CPLX	H	N	N	N
RUSE	N	N	N	N
DOCU	N	N	N	N
TIME	N	N	N	N
STOR	N	N	N	N
PVOL	N	N	L	N
ACAP	VH	VH	VH	H
PCAP	VH	VH	VH	H
PCON	H	H	H	N
APEX	VH	VH	VH	N
PLEX	VH	VH	VH	N
LTEX	VH	VH	VH	N
TOOL	H	H	H	H
SITE	H	H	H	N
SCED	VL then N	VL then N	VL then N	VL then N

Given Mr. Reifer's extensive knowledge of the inner-workings of the COCOMO model, he has manipulated the specific rating factors that have a significant impact on the final estimate, specifically the five Scaling Factors PREC, FLEX, RESL, TEAM, and PMAT, while only conceding on one effort weighting factors, specifically PCON.

In a further analysis of Mr. Reifer's COCOMO II.2000 cost models, where he developed 4 cost estimates, each with 22 factors, Mr. Reifer made the following changes to my assessed factors.

Table #11: PIN000108

	PeopleSoft	JD Edwards EnterpriseOne	JD Edwards World	Seibel
PREC	-	-	-	-
FLEX	-	0	0	-
RESL	-	-	-	-
TEAM	-	-	-	-
PMAT	0	0	0	-
RELY	-	0	-	0
DATA	-	-	-	-
CPLX	-	-	-	-
RUSE	-	-	0	-
DOCU	-	-	-	-
TIME	0	0	0	0
STOR	0	0	0	0
PVOL	0	0	0	0
ACAP	0	0	0	0
PCAP	0	0	0	0
PCON	+	+	+	0
APEX	0	0	0	0
PLEX	0	0	0	0
LTEX	0	0	0	0
TOOL	0	0	0	-
SITE	0	0	0	0
SCED	0	0	0	0

"+" : increased the factor "0": left the factor as-is "-" : decreased the factor

It is important to note that a move from a "High" rating to a "Very High" rating does not necessarily correlate to an increase in the underlying factor, in some case it correlates to a decrease in the underlying factor.

Based on this analysis, Mr. Reifer chose to decrease the "lifted-and-shifted" factors 37.5% of the time (33/88), accept them as-is 58% of the time (51/88), and increase them 3.5% of the time (3/88). While it may appear to an untrained eye that Mr. Reifer has moved some factors "up" while also moving other factors "down", in reality Mr. Reifer has skewed the model to outwardly appear to be performing an unbiased assessment.

Mr. Reifer did not perform an unbiased cost analysis; he simply set all the scaling factors to the lowest conceivable setting, except in 3.5% (3/88) of the instances. Whenever I used a low setting, he defaulted to it. When I chose a higher setting, Mr. Reifer elected to migrate it downward to the lowest possible factor that he felt he could defend. In only 3.5% of the instances (3 out of 88 opportunities) did Mr. Reifer increase my settings, which was done as a token gesture to being "unbiased."

Throughout his report, Mr. Reifer points out that he adopted the setting from my COCOMO models: in tables 24, 33, 40, 49 on pages 60, 69, 75, 83 of his report, where he writes: "I assumed Mr. Pinto checked on the CMM rating", "Best people must be used to achieve Mr. Pinto's schedule goals", "Best tools needed to achieve Mr. Pinto's schedule goals." By doing so, Mr., Reifer has contradicted himself. He defaults to my original model setting

23-times, even though these low setting are in direct conflict with his underlying premise that SAP TN would have built-out a captive development center, in India.

In a later section of this document, I will analyze the scale factors that were selected by Mr. Reifer, within the 4 COCOMO II.2000 models that he built.

25. Reifer's Fourth Estimate

The fourth estimate delivered by Mr. Reifer was to include new cost rates and new code sizes, as he writes on page 88 of his report, he: "…used new labor rates I derived based primarily on variation to the salaries paid in India…".

In a previous section of this document, I addressed the issue associated with Mr. Reifer's assumptions about starting-up a captive, off-shore, software develop shop in an attempt to reduce the offshore costs.

26. Reifer's Fifth Estimate

The Fifth estimate delivered by Mr. Reifer, was to include a reduction in the count of source lines of code, as he writes on page 88 of his report: "I also developed the changes needed to take into account my new size estimates for the code written in C Language for the JD Edwards EnterpriseOne application software."

In a previous section of this document, I addressed the issue associated with how Mr. Reifer determined that the free USC code-counter was supposedly more accurate than the replica of my code-counter.

27. Reifer's Sixth Estimate

The sixth estimate delivered by Mr. Reifer, was to include the application of an optimal development schedule, as he writes on page 85 of his report: "If we then remove the artificial constraint imposed by Mr. Pinto and we assume that the development project should proceed in accordance with the "optimal" schedule…"

In this final model Mr. Reifer changes the one scaling factor that he consistently increased from my model. He changes the SCED to "nominal", which correlates to the same setting that I have in my model. While I applaud the fact that he has come full-circle on this point, it is unrealistic for the development effort to have been completed within 36-months, and still have value to the business model that was being pursued by SAP TN.

28. Reifer's JDE EnterpriseOne Estimates

While I firmly stand by the accuracy of my COCOMO II.1997 cost analysis, as an intelligent Estimator, I accept that the COCOMO II.2000 model can also be used to derive an accurate and fair estimate when it is appropriately calibrated. As such, I went through the effort of evaluating Mr. Reifer's COCOMO II.2000 model for JD Edwards EnterpriseOne, with the intent of only changing the scaling factors that were inconsistent with Mr. Reifer's base assumption about starting-up a captive, off-shore, software develop center.

In performing this analysis, I only focused on the 9 scaling factors that are directly related to the team makeup, level of process maturity, and personnel: TEAM, PMAT, ACAP, PCAP, PCON, AEXP, PLEX, LTEX, and SITE. The results of this analysis are attached in Table #12 in my surrebuttal calculations (PIN000108), and supported by a printout of the results of the USC COCOMO model: USC COCOMO II 2000 JDE EI Reifer(corrected). (PIN000104)

Based on these corrected COCOMO II.2000 models, Mr. Reifer's estimate for the avoided R&D costs associated with JD Edwards EnterpriseOne would be 6.87% higher than my COCOMO II.1997 cost estimate, and 8.66% higher than my Function Point estimate. *See* Table #12, PIN000108.

- ✓ Table #12, PIN000108, COCOMO II 2000 JDE E1 Reifer (corrected),
- ✓ USC COCOMO model: USC COCOMO II 2000 (PIN000104)