

# EXHIBIT 2

2009

# Expert Report of Paul C. Pinto

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

Designated Highly Confidential

Pursuant to Protective Order

Paul C. Pinto  
Managing Partner, Sylvan VI. Inc.  
November 16, 2009



## TABLE OF CONTENTS

	<u>Page</u>
I. INTRODUCTION AND SUMMARY OF OPINIONS .....	1
II. QUALIFICATIONS OF EXPERT WITNESS.....	3
A. Background.....	3
B. Publications.....	4
C. Compensation .....	4
D. Prior Testimony .....	4
E. Material considered.....	4
III. BACKGROUND .....	4
A. General Approach.....	5
B. Additional Value to Infringers.....	6
C. Selection of Function Point Analysis.....	7
IV. SCOPE OF ANALYSIS .....	10
V. FUNCTION POINT METHODOLOGY .....	11
A. Stratification of Products by Source Code Language.....	12
B. Multi-Step Function Point Analysis.....	12
VI. TEN-STEP ANALYSIS TO DETERMINE THE COST OF DEVELOPMENT USING FUNCTION POINT .....	14
A. Step One: Identify and Group Source Code Components .....	14
B. Step Two: Count the Number of Source Lines of Code .....	15
C. Step Three: Determine the Amount of Functionality .....	17
D. Step Four: Determine the Number of Pages of Documentation .....	20
E. Step Five: Derive the Productive Hours of Effort .....	21
F. Step Six: Distribute the Effort across the Product Development Life-Cycle .....	24
G. Step Seven: Allocate Productive Hours of Effort to Team Roles.....	26
H. Step Eight: Derive the Cost of Localization and Documentation Translation....	28
I. Step Nine: Apply Hourly Rates to Determine the Development Costs.....	30
1. JD Edwards EnterpriseOne Development Costs.....	32
2. PeopleSoft Development Costs.....	32
J. Step Ten: Analyze the Estimated Development Costs .....	33
VII. ALTERNATE ESTIMATES .....	34
A. Constructive Cost Model (COCOMO).....	34

TABLE OF CONTENTS  
(continued)

	<u>Page</u>
B. COCOMO II Estimate for JD Edwards EnterpriseOne .....	36
C. COCOMO II Estimate for PeopleSoft .....	38
D. COCOMO II Estimate for JD Edwards World .....	39
E. COCOMO II Estimate for Siebel.....	41
VIII. RESULTS .....	43
A. Summary of Analysis.....	43
IX. REFERENCE MATERIALS.....	44
X. OPTION TO REVISE.....	45

## I. INTRODUCTION AND SUMMARY OF OPINIONS

I, Paul Pinto, submit the following expert 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 Plaintiffs Oracle USA, Inc., Oracle EMEA, Oracle International Corporation, and Siebel Systems, Inc. (collectively, “Oracle” or “Plaintiffs”). I am the co-founder and managing partner of Sylvan VI, Inc., an advisory services firm that provides management consulting services to clients contemplating the selection of a packaged software product or engaging an external service provider to custom-develop software. Prior to founding Sylvan VI, I served as a Senior Executive with Infor Global Software and Epicor Software, both of which publish software products that directly compete with Oracle and SAP. In these roles, I was responsible for running the Software Product Implementation and Managed Services business lines, which focused on implementing, upgrading, customizing, and supporting a variety of ERP and Financial Management software products.

The law firm Bingham McCutchen, on behalf of Oracle, engaged my expert services to estimate the costs associated with development of certain products Oracle alleges were accessed, copied, and used by Defendant TomorrowNow, Inc. (“SAP TN”) through the actions specified in the Fourth Amended Complaint (“Complaint”). I have reached the opinions expressed in this report based on my experience and review and analysis of certain materials produced in this matter.

As explained in more detail below, to reach a conclusion regarding the cost of development, I conducted a series of formal analysis techniques on certain software applications identified in the Complaint to estimate what it would have cost SAP AG, SAP America, Inc., and SAP TN (together, “Defendants”) to independently develop certain software applications accessed, copied, and used by SAP TN as alleged in Complaint. As part of that assessment, I analyzed the available materials and employed industry-accepted methods for estimating the costs associated with conducting all phases of the Product Development Life-Cycle (PDLC), including Plan, Specify, Design, Build, Test, Document, and Deploy, for what I understand to be

the most current copyrighted versions of certain JD Edwards and PeopleSoft software products for which SAP TN provided support services to customers:

- JD Edwards EnterpriseOne, Version 8.12,
- PeopleSoft 8.8 Customer Resource Management (“CRM”),
- PeopleSoft 8.8 Human Resources Management System (“HRMS”),
- PeopleSoft 8.4 Financial Supply Chain Management - rev 1 (“FSCM”),
- PeopleSoft 8.0 Student Administration (“Student Admin”), and
- PeopleSoft 8.8 Enterprise Performance Management - rev 1 (“EPM”)

Using two industry-accepted and reliable methodologies known as Function Point Analysis and COCOMO, I estimated Defendants would have incurred costs in the range of \$764M to \$2,323M (depending on the selected staffing model) to independently develop JD Edwards EnterpriseOne and PeopleSoft CRM, HRMS, FSCM, Student Admin, and EPM modules. Further, the Complaint also alleges Defendants violated Oracle’s intellectual property rights with respect to JD Edwards World and Siebel products. While I did not conduct a low-level Function Point Analysis for these two products, I did conduct a high-level COCOMO analysis. Based on reasonable assumptions regarding likely costs of development of these additional products, in light of the range of costs of development estimated for JD Edwards EnterpriseOne and PeopleSoft products, I estimated that Defendants would have incurred costs in the range of \$1,134M to \$3,477M (depending on the selected staffing model) to independently develop the most current version of JD Edwards EnterpriseOne, JD Edwards World, PeopleSoft, and Siebel applications.

Based on my analysis, it is estimated that 9,772,236 person-hours of productive effort would be required to perform full life-cycle application development for the cited software products (JD Edwards EnterpriseOne, JD Edwards World, PeopleSoft, and Siebel). Assuming there are 144 productive hours in a month, this translates into 67,863 person-months of effort. If the development effort were to be completed within a two-year time frame, the organization would require access to, and the ongoing retention of, more than 2,828 well-trained resources,

throughout the 24-month duration of the project.

## **II. QUALIFICATIONS OF EXPERT WITNESS**

### **A. Background**

A copy of my curriculum vitae is attached as Appendix A. I have worked in the field of Software Development and Enterprise Resource Planning (“ERP”) system related services for 24 years. I spent the first half of my career as a delivery agent assuming progressively more challenging roles in providing Product Development, System Integration, and Managed Services associated with SAP, Oracle, PeopleSoft, and JD Edwards products. The second half of my career has been focused on leveraging my product knowledge to serve as a backdrop for evolving my skills as a management consultant that is focused on providing product development and outsourcing advisory services to global clients.

I am the co-founder of Sylvan VI, Inc., an advisory services firm, which provides management consulting services to clients contemplating the selection of a packaged software product or considering engaging an external service provider to custom develop software. In this role, I leverage my deep knowledge of the software industry and system development life-cycle to provide independent and unbiased advice associated with a client’s “buy” vs. “build” decision.

Prior to founding Sylvan VI, I served as a Senior Executive with Infor Global Software (a \$2.3B Software company in 2008) and Epicor Software (a \$480M Software company in 2007), both of which publish software products that directly compete with Oracle and SAP. In these roles, I was responsible for running the Software Product Implementation and Managed Services business lines, which focused on implementing, upgrading, customizing, and supporting a variety of ERP and Financial Management software products.

Prior to my employment with Epicor, I served as a Senior Vice President for NIIT Technologies (one of the largest India-based systems integration firms). In this role, I was responsible for the day-to-day operations of the U.S. business entity, along with overseeing the sales, estimating, and product development functions for a number of India-based software development centers.

In the 1990s, I was employed by Computer Task Group (a \$500M system integration firm). Throughout my 7-year tenure, I held multiple roles as a delivery agent, where I led a number of high-profile product development projects, provided guidance to troubled projects, and served as a Management Consultant focused on providing ERP package implementation and customization services.

**B. Publications**

I have no publications from the last ten years.

**C. Compensation**

My agreed-upon compensation in this litigation is \$381/hour. My compensation is in no way contingent on the results of my analysis.

**D. Prior Testimony**

I have provided expert witness services in one other matter, *Dibon Solutions Inc. v. Chugach Alaska Corporation* (Case No. 3 AN-08-10957 CI, Case Filing Date: October 3, 2008), where I performed an assessment of a failed software development effort and submitted an expert report. As part of my services, I conducted an analysis of multiple versions of delivered source code, as well as a comparison of two specific versions of source code to identify any copyright violations. As of the date of this report, the *Dibon v. CAC* case is still active, with trial scheduled for January, 2010. Given that this is an active case, under which I am currently governed by a confidentiality agreement, I am disallowed to provide details about my work product.

**E. Material considered**

A list of materials I have considered in preparing this report is attached as Appendix B.

**III. BACKGROUND**

My understanding of the scope of SAP TN's activities is based on the Complaint and my discussion with Kevin Mandia of Mandiant Consulting, who is also retained by Bingham McCutchen on behalf of Oracle in this litigation. I understand that SAP TN provided third party software support services for Oracle's JD Edwards, PeopleSoft, and Siebel applications. Further,



I understand that SAP TN maintained entire copies of Oracle's PeopleSoft, JD Edwards, and Siebel enterprise software applications, as well as fixes, patches, and updates to those enterprise software applications, on SAP TN's computer systems and that SAP TN used these sources in providing support services to its customers. I also understand that SAP TN used copies of Oracle's database software<sup>1</sup> in the provision of support services to its customers.

In light of SAP TN's use of the underlying JD Edwards, PeopleSoft and Siebel enterprise software applications (in addition to using the fixes, patches, and updates for these applications) in providing support for its customers, I have quantified what it would have cost Defendants to independently create the underlying applications - and not just particular fixes, patches, and updates - for the Oracle products identified herein.

The cost of development of the underlying body of applications including the time and technical and litigation risks associated with such development would, in my opinion, and based on my experience, significantly factor into a decision by a potential licensee whether to license a product from the original developer, as well as factoring into the reasonable amount to be paid for that license. In addition, while I do not quantify the cost of development of the database software involved in Defendants' allegedly illegal activities through this report, the cost of development of the database software would also factor into this analysis.

#### **A. General Approach**

In light of the above circumstances, I have focused my analysis on what it would have cost Defendants to independently develop the underlying software applications used in administration of maintenance services provided by SAP TN. I understand that Paul Meyer of Navigant Consulting, who is also retained by Bingham McCutchen on behalf of Oracle in this litigation, will be quantifying actual copyright damages based on the fair market value of Defendants' use. My analysis is related to this fair market value of use analysis because it

---

<sup>1</sup> The term "database software" as used herein refers to any version and edition of Oracle's Relational Database Management System software.

demonstrates a portion of Defendants' avoided costs and avoided risks and avoided delays from infringing, rather than independently developing, the cited products.

Further, over my career as an outsourcing advisor and software company executive, I have been involved in hundreds of license negotiations, from the perspective of both the buyer and the seller of products. In negotiating the price of licenses, I would regularly consider the avoided costs, including saved time and avoided risks (such as avoided Research and Development ("R&D") missteps and avoided litigation from the IP owner) associated with licensing productized software, as opposed to independently developing software. Time and cost, are indeed, the most important considerations to potential licensees in my experience. My estimation of the cost of development is evidence of the investments avoided by not independently developing the products at issue in this litigation.

#### **B. Additional Value to Infringers**

Through my years of industry experience and active consulting work, I am very familiar with the challenges and efforts associated with the development of enterprise application software and the provision of support services for that software. By infringing Oracle's intellectual property rights rather than independently creating the products specified in the Complaint, Defendants would have avoided the costs associated with independent development.

Defendants also received a number of other benefits related to avoided cost, in the form of quicker time to market and avoided risks, including the avoidance of: the significant upfront monetary outlay necessary to create the intellectual property; the risk of taking wrong turns or making errors in the development process; the risk that the personnel necessary to complete the project were unavailable; and the risk that the creation of the product would take longer than anticipated and therefore the desired customer base would remain with the original support provider.

As discussed above in my summary of opinions, the ramp-up needed for a software development effort of this size would require access to, and the ongoing retention of, more than 2,828 well-trained personnel, for a period of no fewer than two years, to develop all of the cited

software products. A development effort of this scope and complexity would be an extremely large project, very aggressive, and of high-risk to be pursued within this timeframe. It would be exceedingly difficult for a project of this magnitude to be successfully completed within a 24 month period, but equally difficult for business reasons (e.g., pursuing a time sensitive market opportunity) for the development effort to exceed 24 months.

Based on the required level of business and technical knowledge and expected attrition, however, it would be tenuous to retain a team of this size and caliber for the required duration within a single U.S. city. While there are a limited number of U.S. cities that possess a large enough, technically qualified talent pool, these same cities house a number of established software development shops which actively compete for the best technical resources. As a whole, these circumstances highlight why my cost estimate is particularly conservative in light of the constraints at issue. Infringement, rather than independent development, would save not only the costs of development identified through my Function Point and COCOMO analyses, but the significant time and risk associated with independent development.

Further, testimony from this litigation reflects the additional value that would come from hiring personnel with experience through former employment by JD Edwards, PeopleSoft, Siebel, and/or Oracle.<sup>2</sup> The desire to hire an even smaller subset of available personnel, namely, personnel with experience in similar roles at JD Edwards, PeopleSoft, Siebel, or Oracle, could potentially drive up the labor costs even further.

### **C. Selection of Function Point Analysis**

While the benefits to Defendants from infringement rather than development are extensive, this report specifically quantifies a sub-set of those benefits associated with the dollar value of avoided R&D expenses. As described in Section V, I created an estimated cost of development for JD Edwards EnterpriseOne and PeopleSoft applications, using Function Point

---

<sup>2</sup> See, e.g., December 5, 2008 Deposition of Matthew Bowden at 46:13-47:25; January 6, 2009 Deposition of Shai Agassi at 119:17-120:2; May 21, 2009 Deposition of Seth Ravin at 11:15-12:20, 19:11-21:12.

Analysis. This method of analysis is focused on assessing the size of a software product, in normalized terms that are directly related to the amount of business functionality provided to the end-user of the application. As such, this approach can be applied across a wide range of application development environments and throughout the full life-cycle of the software development effort. When coupled with a series of business metrics, such as productivity and the hourly rates for assigned personnel, the total cost of application development can be readily derived.

The method of Function Point Analysis was introduced in 1979 (by IBM), and is actively maintained by the International Function Point Users Group (“IFPUG”) as part of its Functional Size Measurement Method. Function Point Analysis provides an objective, comparative measure that assists in the evaluation, planning, management, and control of software production. Among other things, it is used, as applied here, to develop an estimated cost of development of a software product.<sup>3</sup>

I chose to use Function Point Analysis for this assessment because it is recognized by the International Standards Organization (“ISO”) as a valid method for assessing the size of a software product and for deriving the associated cost of product development.<sup>4</sup> It is also recognized by a number of the world’s largest I.T. consulting companies and has been used by IBM, TCS, and Infosys since its inception. Also, I have considerable experience applying the required techniques in real business scenarios, where it is regularly used to estimate software development efforts and associated costs that are based on a set of defined requirements, which is known as “forward-engineering.” I have also applied this method in situations where legacy software products needed to be redeveloped onto a modern computing platform, while maintaining the existing functionality.

---

<sup>3</sup> International Function Point Users Group, About IFPUG, <http://www.ifpug.org/about>. [ORCLX-PIN-000008]

<sup>4</sup> International Standard ISO/IEC, 20926, Manual, October 2003, Software engineering - IFPUG 4.1 Unadjusted functional size measurement method - Counting practices manual, [http://webstore.iec.ch/preview/info\\_isoiec20926%7Bed1.0%7Den.pdf](http://webstore.iec.ch/preview/info_isoiec20926%7Bed1.0%7Den.pdf). [ORCLX-PIN-000009]

#### **D. Selection of COCOMO Analysis**

To confirm the estimates reached through Function Point Analysis for the JD Edwards EnterpriseOne and PeopleSoft products, and to assess the cost of development for the JD Edwards World and Siebel products, I applied an alternate estimating method known as Constructive Cost Model (COCOMO) analysis. COCOMO is an industry-accepted method that provides a reliable approach to performing high-level “top-down” estimating, as a valid alternate method to performing a low-level “bottom-up” analysis as is required for Function Point Analysis.

COCOMO is an algorithm-based software cost estimation model that employs the use of regression formulas, coupled with parameters that were derived from historical project characteristics. The model was originally published in 1981 as a method for estimating the level of effort, project duration, and costs associated with developing software. This original model was referred to as COCOMO 81.<sup>5</sup>

In 2001, the second version of the model, COCOMO II, was published. This recent iteration is better suited for estimating modern software development projects, by providing an updated set of project characteristics that are more aligned with today’s software development tools, iterative approaches, and relational databases. The need for this new model was prompted by the evolution of software development technologies, which moved away from mainframe and overnight batch processing, and moved toward desktop development and code reusability.<sup>6</sup>

COCOMO II estimates the software development effort as a function of a limited set of “scaling drivers” that describe the development process, and a set of “cost drivers” that include subjective assessments about the product, platform, personnel, and project attributes. The end result of a COCOMO II analysis is the estimated total cost of development.

---

<sup>5</sup> COCOMO Model II, Center for Systems and Software Engineering, [http://csse.usc.edu/csse/research/COCOMOII/cocomo\\_main.html](http://csse.usc.edu/csse/research/COCOMOII/cocomo_main.html). [ORCLX-PIN-000003]

<sup>6</sup> *Id.*

I chose to apply COCOMO II analysis here (which I also refer to generally as “COCOMO”), because it provides a reliable method for confirming the development costs for JD Edwards EnterpriseOne and PeopleSoft that were estimated through Function Point Analysis. COCOMO analysis also allows the JD Edwards EnterpriseOne and PeopleSoft estimates to be reasonably extrapolated to the JD Edwards World and Siebel products, respectively.

#### **IV. SCOPE OF ANALYSIS**

As described in Section III above, I understand that SAP TN used copies of Oracle’s PeopleSoft, JD Edwards, and Siebel enterprise software applications, as well as fixes, patches, and updates to those software applications, to provide support services to SAP TN customers. In light of the overall volume of material put at issue by SAP TN’s actions, I focused the majority of my effort on a targeted subset of this material. Specifically, I analyzed the cost of development for the following Oracle products using a Function Point Analysis:

- JD Edwards EnterpriseOne, Version 8.12,
- PeopleSoft 8.8 Customer Resource Management (“CRM”),
- PeopleSoft 8.8 Human Resources Management System (“HRMS”),
- PeopleSoft 8.4 Financial Supply Chain Management - rev 1 (“FSCM”),
- PeopleSoft 8.0 Student Administration (“Student Admin”), and
- PeopleSoft 8.8 Enterprise Performance Management - rev 1 (“EPM”)

These products offered the advantage of providing relatively easy access to the components of Source Code, which was required for my analysis. I also understand that these products represent the latest copyrighted versions of these products that were also supported by SAP TN.

After concluding my Function Point Analysis, I performed a COCOMO analysis on the products listed above, as well as on the JD Edwards World and Siebel products that I understand are also at issue in this litigation.

If I were to assume a less conservative posture, I could have reasonably analyzed the cumulative development costs associated with each of the prior product versions, along with the development costs associated with producing the ongoing fixes, patches, and updates for each

version. I did not do so in order to ensure that there was no double counting of any development efforts between versions. I also could have analyzed the value of time associated with acquiring instant access to the software applications, as opposed to enduring the time required for developing the cited products. Instead, however, I focused my analysis on the pure cost of development of the underlying products themselves. These examples demonstrate ways in which my report represents a conservative position.

## V. FUNCTION POINT METHODOLOGY

The approach of Function Point Analysis was carefully selected, based on the existence of well-documented and widely-accepted estimating practices that provided the ability to reverse-engineer the costs associated with full life-cycle product development, using the size and complexity of the underlying Source Code as a proxy for the total cost of development.

IFPUG, [www.ifpug.org](http://www.ifpug.org), which is a non-profit, member-governed organization, provides a measurement technique called Function Point Analysis (“FPA”) for the functional sizing of software. IFPUG endorses FPA as its standard methodology for software sizing. Furthermore, IFPUG participates as a Lead Member in the International Software Benchmarking Standards Group Ltd. (“ISBSG”).<sup>7</sup>

In adopting a conservative posture in the scope of my analysis, I determined that only the most recent copyrighted versions of JD Edwards EnterpriseOne and PeopleSoft that were supported by SAP TN would be analyzed to derive the cost of development for purposes of my analysis. By focusing on the most recent versions, I thereby assumed that all of the work effort associated with creating prior versions would be accounted for when estimating the costs associated with developing the most recent version.

JD Edwards World and Siebel products are also at issue in this litigation. Although these two products were not analyzed using Function Point Analysis, the alternate method of

---

<sup>7</sup> International Standard ISO/IEC, 20926, Manual, October 2003, Software engineering - IFPUG 4.1 Unadjusted functional size measurement method - Counting practices manual, [http://webstore.iec.ch/preview/info\\_isoiec20926%7Bed1.0%7Den.pdf](http://webstore.iec.ch/preview/info_isoiec20926%7Bed1.0%7Den.pdf). [ORCLX-PIN-000009]

COCOMO analysis was applied. JD Edwards World and Siebel products were estimated based on the results of the PeopleSoft and JD Edwards EnterpriseOne analyses, my industry experience with these products, and input from Oracle, which combined, reasonably and reliably inform the expected total cost of development of the entire body of stolen products/modules, with the exception of the Oracle database software. The following description of my analysis applies to the explicit steps taken only to analyze JD Edwards EnterpriseOne and PeopleSoft. Section VIII provides my opinions regarding what can be reasonably opined regarding all of the infringed products.

#### **A. Stratification of Products by Source Code Language**

For the purpose of conducting my analysis, I extracted the underlying Source Code from the analyzed products as described in Section IV. This underlying Source Code next had to be organized into groups, based on the affinity of their underlying programming languages. Below, Table 1 (ORCLX-PIN-000065 Table 1) describes these groupings.<sup>8</sup>

<b>Stratification by Programming Language</b>	
<b>Software Product Version</b>	<b>Programming Language Groupings</b>
JDE EnterpriseOne Version 8.12	C
	Java J2EE
PeopleSoft Version 8.X	COBOL/400
	SQC, SQR, DMS and SQL
	RPT and MDL
	PeopleCode

*Table 1 - Language Groupings*

#### **B. Multi-Step Function Point Analysis**

To develop an accurate and demonstrable cost estimate associated with developing the intellectual property contained within the products, I adopted a “bottom-up” approach to performing my assessment. This micro-approach required a detailed analysis of the underlying Source Code components for each product. The approach to estimating encompassed a ten-step

<sup>8</sup> Throughout my report, references to “PeopleSoft Version 8.X” refers to the PeopleSoft modules I described in Section IV (“Scope of Analysis”). The “8.X” reflects that for different modules, there are different numbering conventions within Version 8. For example, my model includes version 8.8 of module HRMS, but version 8.4 of module FSCM.



process, with each step building on the results of previous steps. As part of this process, I applied a variety of conversion and translation techniques that are based on well-documented, industry-recognized metrics and standards. To maintain a conservative posture, I elected to use the most conservative weighting/conversion factors from the provided metrics, whenever it made sense to do so. In certain instances, and based on my personal field experience, I elected to assume a more conservative metric than was stipulated by the cited source. Below, Table 2 (ORCLX-PIN-000065 Table 2) provides a high-level description of this process, along with a brief description of the required input information, activities performed, techniques applied, and resulting output information. Section VI explains each of these steps in greater detail.

<b>Estimating Approach (high-level view)</b>				
<b>Step Number</b>	<b>Input Information</b>	<b>Activity Performed</b>	<b>Technique Applied</b>	<b>Output Information</b>
1	Code Components	Identify, group, and count the source code components for the most recent versions of the Analyzed Products	Manual identification	Stratified Source Code Components
2	Stratified Source Code Components	Count the number of Source Lines of Code in each grouping, within each of the Analyzed Products	Automated counting of source lines of code	Number of Source Lines of Code
3	Number of Source Lines of Code	Determine the amount of functionality contained in each grouping, within each of the Analyzed Products	Applying conversion tables that converts source lines to function points	Number of Function Points
4	Number of Function Points	Determine the number of pages of documentation associated with each of the Analyzed Products	Applying conversion tables that converts function points to pages of documentation	Number of Pages of Documentation
5	Number of Function Points	Derive the effort associated with performing full life-cycle product development for each grouping, within each of the Analyzed Products	Apply development phase metrics and documentation metrics	Effort Estimate
6	Effort Estimate	Distribute effort across the product development life-cycle for each of the Analyzed Products	Apply development role-based metrics	Effort Distribution by Phase
7	Effort Distribution by Phase	Allocate phase effort to the product development team roles for each of the Analyzed Products	Apply productivity tables that associate roles with effort	Effort Allocation by Role
8	Number of Pages of Documentation	Derive the level of effort associated with localizing and translating the required documentation into a number of foreign languages	Apply localization/translation metrics	Estimated Cost of Document Translation

Estimating Approach (high-level view)				
Step Number	Input Information	Activity Performed	Technique Applied	Output Information
9	Effort Allocation by Role	Apply hourly rates and determine the cost of development for each of the Analyzed Products, across a number of staffing scenarios	Apply resource costs	Estimated Cost of Development
10	Estimated Cost of Development	Analyze the estimated development costs for each of the Analyzed Products	Automated calculations	Per Unit Cost Calculations

Table 2 - Estimating Approach

## VI. TEN-STEP ANALYSIS TO DETERMINE THE COST OF DEVELOPMENT USING FUNCTION POINT

### A. Step One: Identify and Group Source Code Components

The purpose of identifying and grouping the Source Code components, from the total population of application components, was to identify and isolate those components from which meaningful estimates could be derived. While other components, such as application code, utilities, database files, screens, and documentation are all relevant and interesting, Function Point Analysis is designed to derive the effort required to create all of these other components as a function of understanding the size and characteristics of the associated Source Code. By applying Function Point Analysis, the development costs for all components can be extrapolated from understanding the underlying Source Code.

The entire set of software components was reviewed, with a focus on identifying the components that represented Source Code. This was done by reviewing the file extensions to identify the file types that could contain Source Code, and then opening each suspected file to confirm that it did indeed contain valid Source Code. As the components of Source Code were identified, they were then grouped by product, module, version, and programming language.<sup>9</sup>

---

<sup>9</sup> As the components of Source Code were identified, they were then grouped by product, module, version, and programming language. For the JD Edwards EnterpriseOne and PeopleSoft products, I was provided with the complete applications in the form of ISO files (images of CDs or hard drives), which were physically delivered in a series of external hard drives. With regard to PeopleSoft, a significant component of the Source Code was written in PeopleCode, which resided as objects within the database. For the purpose of my analysis, Oracle extracted these objects en masse from the PeopleSoft modules listed in Section IV and provided me with the set of PeopleCode as

The original input for Step One was the software products/modules for the Analyzed Products. Below, Table 3 (ORCLX-PIN-000065 Table 3) displays the number of Source Code programs, for the identified groupings.

Number of Source Code Programs			
Software Product Version	Programming Language Groupings	Number of Source Code Programs	Totals
JDE EnterpriseOne Version 8.12	C	28,471 Programs	38,634 Programs
	Java J2EE	10,163 Programs	
PeopleSoft Version 8.X	COBOL/400	3,657 Programs	17,480 Programs/Files
	SQC, SQR, DMS and SQL	12,146 Programs	
	RPT and MDL	1,663 Programs	
	PeopleCode	14 Files (w/multiple programs)	
<b>Totals:</b>		<b>56,114 Programs/Files</b>	<b>56,114 Programs/Files</b>

Table 3 - Source Code Programs

Detailed inventories of Source Code files, grouped by stratum, have been produced as bates number ORCLX-PIN-000063 for JD Edwards EnterpriseOne, and bates number ORCLX-PIN-000064 for PeopleSoft.

### **B. Step Two: Count the Number of Source Lines of Code**

The next step involved counting Source Lines of Code (“SLOC”) using specially-designed counting utilities. Counting SLOC is a simple procedure that provides an accurate predictor of development effort.<sup>10</sup> When development effort is appropriately attributed to the roles that participate in the Product Development Life-Cycle, and then combined with hourly rates, enough information is available to develop a reliable estimate of the cost of product development.<sup>11</sup>

Counting SLOCs still requires a certain amount of nuance, however. Imbedded within Source Code are various statements such as: physical lines of code, logical source lines of code, blank lines, and commented (unused or educational) lines of code. Each software development

---

text files produced at ORCLX-PIN-000024 to ORCLX-PIN-000062.

<sup>10</sup> Software Size Measurement: A Framework for Counting Source Statements, Technical Report CMU/SEI-92-TR-020, ESC-TR-92-020, September 1992, Robert E. Parker, Software Engineering Institute at Carnegie Mellon University, pgs. 13-15. [ORCLX-PIN-000017]

<sup>11</sup> *Id.* at 1-15.

language has rules for constructing its Source Code, in the same way that the English language has rules for constructing statements and sentences. These software coding rules, or standards, enable software utilities to be built that can distinguish the different rules and, therefore, count the different types of statements. The end product is the total number of logical Source Lines of Code.

Since 1984, the Software Engineering Institute (SEI), at Carnegie Mellon University, has established standards for defining a Logical Source Code Statement. SEI is a federally-funded research and development center that conducts software engineering research in acquisition, architecture and product lines, process improvement and performance measurement, security, and system interoperability and dependability.<sup>12</sup> I relied on these standards for this portion of my analysis.

In order to use the logical Source Lines of Code count as the foundation for estimating software size and ultimately deriving the total cost of development, I constructed a number of software utilities that counted the logical Source Lines of Code, which are produced as ORCLX-PIN-000066 to ORCLX-PIN-000085. Each line counting utility was specifically designed and tailored to address the specific needs of each type of source code that was analyzed (e.g., COBOL, C, SQL, SQR, etc). Below, Table 4 (ORCLX-PIN-000065 Table 4) is a sample of the output from the automated code counting utility for a series of “C” program files.

Sample SLOC Counting Utility Output (for JDE EnterpriseOne example)		
File Name	Total Lines of Source Code	Logical Source Lines of Code
n4002340.c	701	379 SLOC
n4002350.c	984	519 SLOC
n4002380.c	882	315 SLOC
n4002400.c	192	81 SLOC
n4002440.c	801	410 SLOC

*Table 4 - Sample SLOC Counting*

In sum, Step Two involved counting the number of logical SLOC within each grouping, which then served as the basis for establishing the size of the code base in subsequent steps. The

<sup>12</sup> *Id.* at 13-21.

Source Code components, as identified in Step One, were used as the input for determining the number of logical SLOC. Below, Table 5 (ORCLX-PIN-000065 Table 5) displays the size of code base, for the identified groupings, expressed as the number of logical SLOC.

Number of Source Lines of Code			
Software Product Version	Programming Language (stratum)	Number of logical Source Lines of Code	Totals
JDE EnterpriseOne Version 8.12	C	6,906,168	7,774,791 SLOC
	Java J2EE	868,623	
PeopleSoft Version 8.X	COBOL/400	2,057,468	7,650,493 SLOC
	SQC, SQR, DMS and SQL	2,282,005	
	RPT and MDL	244,760	
	PeopleCode	3,066,260	
<b>Totals:</b>		<b>15,425,284</b>	<b>15,425,284 SLOC</b>

Table 5 - Source Lines of Code

### C. Step Three: Determine the Amount of Functionality

Step Three involves a process known as Backfiring to determine the amount of functionality. As explained above in Section V, Function Point Analysis is a method for determining the size of a software product, by describing it in terms of the amount of work being performed within the programming code. The major objective of Function Point Analysis is to describe the quantity of functionality that is contained in a component of Source Code, and to establish an objective statement of software size, which is independent of the technology in which it is written. Function Point Analysis, when paired with Backfiring, is a valuable technique for deriving the size of software in normalized terms. Backfiring refers to the process of using the end-product, in this case the Source Code, to determine the size of the application development effort that was used to produce it.

Considerable research has been performed regarding the expressive power of computer languages. In particular, this research indicates how many logical SLOCs are required to implement a Function Point of work, with a single Function Point of work consisting of an elementary process that performs one of the following types of system-related activities:<sup>13</sup>

<sup>13</sup> Function Point Counting Practices Manual, Release 4.2, ISBN 0-963-1742-9-0, The International Function Point

- Internal logical File (ILF) – holds and maintains information that is stored within the boundaries of a specific program/module. For every piece of data that is stored, updated, maintained, and retrieved from within the database, the system is acknowledged as performing a corresponding Function Point of work.
- External Interface File (EIF) – controls information that is passed to other related application programs/modules that are outside the boundaries of the specific program/module. For every piece of data that is passed from the database to another program, the system is acknowledged as performing a corresponding Function Point of work.
- External Input (EI) – controls information that is entered into the system from a User of the application. For every piece of data that is keyed into an application through a User's screen, the system is acknowledged as performing a corresponding Function Point of work.
- External Output (EO) – sends processed information outside of the program/module, so it can be viewed by a User. For every calculation that occurs and returns a value to the User's screen, the system is acknowledged as performing a corresponding Function Point of work.
- External Inquiry (EQ) – presents requested information to the User that is retrieved from the database, without any further processing. For every piece of information that is displayed on the User's screen, as a result of a lookup in the database, the system is acknowledged as performing a corresponding Function Point of work.

In particular, this research has produced a series of tables that indicate how many logical Source Lines of Code (SLOC) are required to implement a Function Point of work. These figures vary by computing language, in direct relationship with the language's level of

sophistication. As a language becomes more robust and powerful, fewer logical SLOCs are required to perform one Function Point of work.

To complete Step Three, I estimated how many Function Points were present based on the number of logical SLOCs contained in each grouping by applying a set of conversion rules. For example, there was a component of JD Edwards EnterpriseOne Source Code that was counted to have 91,182 logical SLOCs of C Code. By applying the conversion rate cited in the Backfiring table, I determined that there were 914.56 Function Points of work being performed by this specific component of code. The complete set of conversion tables (SPR PLT) can be found in a study that is provided by Software Productivity Research LLC (SPR).<sup>14</sup>

Below, Table 6 (ORCLX-PIN-000065 Table 6) is an excerpt from the SPR-provided conversion tables (SPR PLT). Based on the languages that were relevant to performing my analysis, I applied the low values for determining the number of logical source statements per Function Point (FP). My selection of the low values was based on my assessment of the source code as being produced in an efficient manner, by a professional product development shop that clearly adhered to a set of application coding standards.

Logical Source Lines of Code (SLOC) Mapping to Function Points (FP)			
Programming Language	SLOC/FP		
	Low	Median	High
C	99.7	139.2	178.8
COBOL/400	85.3	91.9	98.6
Java J2EE	13.6	19.4	25.1
SQC, SQR, DMS and SQL (comparable to SQL)	10.5	13.3	16.2
RPT and MDL (comparable to Crystal Reports)	14.9	22.3	29.8
PeopleCode (comparable to Visual Basic 6)	21.6	28.2	34.9

*Table 6 - SLOC Mapping to FP*

I converted the number of logical SLOC, for each grouping, into a number of Function Points (FPs). The number of logical SLOC, as identified in Step Two, along with an industry-

<sup>14</sup> SPR Programming Languages Table Version PLT2007c, December 28, 2007, SPR Tables, Software Productivity Research, LLC, pgs. 16-21. [ORCLX-PIN-000019]

accepted SLOC-to-FP conversion table (ORCLX-PIN-000065 Table 6), was used as the input for determining the number of Function Points. Below, Table 7 (ORCLX-PIN-000065 Table 7) displays the amount of functionality for the identified grouping, expressed as Function Points of work.

Number of Function Points		
Software Product Version	Programming Language Groupings	Number of Function Points
JDE EnterpriseOne Version 8.12	C	69,269 FPs
	Java J2EE	63,869 FPs
PeopleSoft Version 8.X	COBOL/400	24,120 FPs
	SQC, SQR, DMS and SQL	217,334 FPs
	RPT and MDL	16,427 FPs
	PeopleCode	141,956 FPs
<b>Totals:</b>		<b>532,976 FPs</b>

*Table 7 - Function Points*

#### **D. Step Four: Determine the Number of Pages of Documentation**

Step Four estimates the number of pages of documentation that will accompany a software product. As the functionality of a software product increases, so does the required amount of documentation. With regard to producing a commercially-available product, the documentation must include a set of User Documentation as well as Support Documentation. User Documentation includes all documentation that is expressly developed to educate the End User about how to use the software to perform their daily work activities, while Support Documentation is targeted at aiding Technicians in installing and maintaining the software.

Based on industry metrics, the number of pages of documentation required to provide an appropriate level of support for a commercial software product is equal to the number of Function Points contained within the product, multiplied by a factor that ranges between 2.50 and 3.33 pages.<sup>15</sup> For the purposes of my analysis, I assumed a conservative posture, and elected to use a multiplier of 2.50 pages (the low-end of the range) for deriving the number of pages of

---

<sup>15</sup> Estimating Software Effort, SoftwareMetrics.com website, <http://www.softwaremetrics.com/Articles/estimating.htm>. [ORCLX-PIN-000005]



documentation. Below, Table 8 (ORCLX-PIN-000065 Table 8) displays the type of documentation, the contents, and the number of pages that are produced for each Function Point.

Pages of Documentation per Function Points (FPs)			
Type of Documentation	Contents	Number of Pages per FP (low-end)	Number of Pages per FP (high-end)
User Documentation	Quick start users guide	2.14	2.86
	Full users guides		
	Feature users guides		
	HELP text		
	README files		
	Training course materials		
	Instructors guides		
	Reference manuals		
	Glossaries		
CD-ROM information			
Support Documentation	Operators guides	0.36	0.47
	Maintenance Guides		
<b>Range:</b>		<b>2.50</b>	<b>3.33</b>

Table 8 - Pages per FP

In Step Four, therefore, I determined the number of pages of documentation for the analyzed products from the number of Function Points. The count of Function Points, as identified in Step Three, along with a Function Point-to-Page conversion table (ORCLX-PIN-000065 Table 8), were used to determine the number of pages of documentation. Below, Table 9 (ORCLX-PIN-000065 Table 9) displays the number of pages of documentation (in English) that are required for the analyzed products.

Number of Pages of Documentation (in English)	
Software Product Version	Number of Pages of English Documents
JDE EnterpriseOne Version 8.12	332,847 Pages
PeopleSoft Version 8.X	999,594 Pages
<b>Totals:</b>	<b>1,332,441 Pages</b>

Table 9 - Pages of Documentation

### E. Step Five: Derive the Productive Hours of Effort

The next step in the Function Point Analysis is to determine the number of hours of work required to develop a single Function Point, also known as the Productive Hours of Effort (PHE). The PHE refers to the number of person-hours that are required to produce one Function Point of work, within a specific programming language. As the sophistication of the underlying

computing language increases, so does its ability to increase productivity by providing more functionality with fewer logical SLOC. It is commonly accepted, in the product development arena, that one staff month is equal to 144 productive hours.<sup>16</sup> To facilitate the extension of development efforts into costs, I have used productive hours as the base unit of development time.

Below, Table 10 (ORCLX-PIN-000065 Table 10) identifies the productivity levels for the coding languages that are required for developing the analyzed products (ORCLX-PIN-000065).<sup>17</sup> The “Median” language level was assigned because it is best suited for the typical product development scenarios, and represents the most fair and objective metric.

Language Level Relationship to Productivity			
Language	Language Level (Median)	Productivity (FPs per month)	Productivity (Hour per FP)
C	2.5	8.75	16.46
Java J2EE	18.1	19.50	7.38
COBOL/400	3.5	10.36	13.90
SQC, SQR, DMS and SQL (comparable to SQL)	25.1	30.71	4.69
RPT and MDL (comparable to Crystal Reports)	16.1	15.21	9.46
PeopleCode (comparable to Visual basic 6)	12.0	19.50	7.38

*Table 10 - Language Levels and Productivity*

I also needed to calculate the PHE associated with producing the required amount of commercial documentation. To accomplish this, I applied an alternate set of metrics that is based on the number of pages of documentation. Each page of documentation is developed through an iterative writing process that includes a series of interviews, drafts, and reviews. With regard to producing documentation for a commercially-available software product, the process becomes even more rigorous. Based on industry metrics, the number of hours required to produce commercial documentation is equal to the number of pages of documentation multiplied by a factor that ranges between 1.75 and 3.25 hours of effort.<sup>18</sup> For the purposes of

<sup>16</sup> The ESA Initiative for Software Productivity Benchmarking and Effort Estimation, D. Greves & B. Schreiber, ESA Cost Analysis Division, ESTEC. <http://www.esa.int/esapub/bulletin/bullet87/greves87.htm>, at pg. 4. [ORCLX-PIN-000013]

<sup>17</sup> SPR Programming Languages Table Version PLT2007c, December 28, 2007, SPR Tables, Software Productivity Research, LLC, pg 8, 15-21. [ORCLX-PIN-000019]

<sup>18</sup> Estimating Tech Writing Jobs, Tech-writer.net website article, Estimating Tech Writing Jobs, <http://www.tech->

my analysis, I assumed a conservative posture, and elected to use a multiplier of 1.75 hours (the low-end of the range) for deriving the level of effort required to produce the base product documentation in English. Below, Table 11 (ORCLX-PIN-000065 Table 11) displays the writing activities and their associated amount of effort (in hours).

Effort (hours) per Page of Documentation (in English)		
Writing Activity	Time Required per Page (high-end)	Time Required per Page (low-end)
Interview/Discussions	0.50	0.25 Hours
Draft	1.00	0.50 Hours
Graphics (1 every 2-3 pages min.)	1.00	0.50 Hours
Edit/Review	0.50	0.25 Hours
Index	0.25	0.25 Hours
<b>Total:</b>	<b>3.25</b>	<b>1.75 Hours</b>

*Table 11 - Effort per Page of Documentation*

Thus, for Step Five, I converted the number of FPs within each grouping by product language into Productive Hours of Effort (PHE), and converted the number of pages of documentation into PHE. The count of FPs, as identified in Step Three, along with a number of FP-to-effort conversion tables (ORCLX-PIN-000065 Table 11), were used as the input for converting the number of FPs into the PHE. Below, Table 12 (ORCLX-PIN-000065 Table 12) displays the amount of PHE required to perform full life-cycle product development (including documentation), for the identified groupings, expressed as person-hours of effort.

Number of Productive Hours of Effort			
Software Product Version	Programming Language (stratum)	Total number of Productive Hours of Effort	Total Hours of Effort
JDE EnterpriseOne Version 8.12	C	1,139,978	2,194,111 Hours
	Java J2EE	471,650	
	Documentation (in English)	582,482	
PeopleSoft Version 8.X	COBOL/400	335,356	4,307,511 Hours
	SQC, SQR, DMS and SQL	1,019,095	
	RPT and MDL	155,477	
	PeopleCode	1,048,294	
	Documentation (in English)	1,749,289	
<b>Totals:</b>		<b>6,501,622</b>	<b>6,501,622 Hours</b>

*Table 12 - Productive Hours of Effort*

## F. Step Six: Distribute the Effort across the Product Development Life-Cycle

After determining the amount of PHE required to perform full life-cycle product development, it is necessary to distribute that effort across the Product Development Life-Cycle (PDLC). This is an interim step to ultimately assigning particular hours to specific roles that perform the activities within the PDLC. The PDLC refers to the activities associated with constructing a software application from inception to deployment, and underpins many types of software development methodologies, which form the framework for estimating the software development effort.

The International Software Benchmarking Standards Group (ISBSG) defines the standard phases for the PDLC as Plan, Specify, Design, Build, Test, and Implement.<sup>19</sup> Below, Table 13 (ORCLX-PIN-000065 Table 13) provides a listing of the ISBSG phases, their associated components, and their associated Productivity Benchmarks.

Effort Distribution by Phases of the PDLC		
Phase of the PDLC	Activities	Percentage of Effort per Phase
1. Plan	Product Management	14%
	Preliminary Investigations	
	Overall Project Planning	
	Feasibility Study	
	Cost Benefit Study	
	Project Initiation Report	
	Terms of Reference	
2. Specify	System Analysis	11%
	Requirement Specification	
	Review & Rework Requirements Specification	
	Architecture Design / Specification	
	Review & Rework Architecture Specification	
3. Design	Functional / External Design	21%
	Create Physical / Internal Design(s)	
	Review and Rework Design(s)	
4. Build	Construct Code and Program Software	34%
	Review or Inspect and Rework Code	
	Package Customization / Interfaces	
	Unit Test	
	Integrate Software	
	Configuration Management	

<sup>19</sup> Industry Software Cost, Quality and Productivity Benchmarks, whitepaper, April 2004, by Donald J Reifer, Reifer Consultants, Inc., <http://www.compaid.com/caiinternet/ezine/Reifer-Benchmarks.pdf>. [ORCLX-PIN-000014]

Effort Distribution by Phases of the PDLC		
Phase of the PDLC	Activities	Percentage of Effort per Phase
	Multiple Platforms	
	Localization	
5. Test	Plan System or Performance Testing	12%
	System Testing	
	Performance Testing	
	Create & Run Automated Tests	
	Acceptance Testing	
6. Localize / Document	Prepare User Documentation	Defined as part of an alternate estimating technique
	Prepare Technical Documentation	
	Prepare User Training Curriculum	
	Prepare Computer-based Training	
	Prepare Web-based Training	
7. Deploy	Alpha-Release	8%
	Beta-Release	
	Prepare Releases for Delivery	
	Install Software Releases for Users	
	Deliver User Training	
	Provide User Support	
<b>Total:</b>		<b>100%</b>

*Table 13 - Effort Distribution Across PDLC*

While the PDLC is acknowledged to accurately represent the full life-cycle of the software development process, it is also acknowledged that it does not include the development of documentation, and its localization/translation into foreign languages. For this reason, the cost associated with developing documentation was estimated through an alternate method as described in Step Five and Step Eight.

Step Six therefore involved distributing the PHE, for the analyzed products, across the entire PDLC. The PHE, as identified in Step Five, along with the standard PDLC effort allocation distribution percentages (ORCLX-PIN-000065 Table 13), were used as the input for distributing the PHE across the PDLC. Below, Table 14 (ORCLX-PIN-000065 Table 14) displays the PHE distributed across the phases of the PDLC, including the localization and documentation, for each software product version.

Productive Hours of Effort (PHE) Distributed by Phase of Product Development Life-cycle (PDLC)			
Phase of the PDLC	JDE EnterpriseOne version 8.12	PeopleSoft Version 8.X	
1. Plan	225,628	358,151	Hours
2. Specify	177,279	281,404	Hours
3. Design	338,442	537,227	Hours
4. Build	547,954	869,795	Hours
5. Test	193,395	306,987	Hours
6. Localize / Document	582,482	1,749,289	Hours
7. Deploy	128,930	204,658	Hours
<b>Totals:</b>	<b>2,194,111</b>	<b>4,307,511</b>	<b>Hours</b>

Table 14 - PHE by PDLC

### G. Step Seven: Allocate Productive Hours of Effort to Team Roles

Within the phases of the PDLC, there are a number of Team Roles that are involved in performing specific work activities. The next step in the Function Point Analysis was to appropriately distribute the total work effort across all of the involved Roles, by using the following distribution method.<sup>20</sup> Below, Table 15 (ORCLX-PIN-000065 Table 15) displays this distribution method.

Effort Distribution by Project Roles		
Phase of the PDLC	Team Role	Percentage of Effort by Role
1. Plan	Program Manager	3%
	Project Manager	9%
	Quality Manager	2%
2. Specify	Product Manager	3%
	Business Analyst	8%
3. Design	Technical Leader	6%
	Technical Architect	6%
	Technical Analyst	9%
4. Build	Developer	21%
	Database Administrator	9%
	Configuration Manager	4%
5. Test	Test Lead	3%
	Tester	7%
	Quality Auditor	2%
6. Localize / Document	Documentation Lead	Defined as part of an alternate estimating technique
	Technical Writer	
	Curriculum Developer	
7. Deploy	Implementation Lead	2%
	Functional Lead	3%
	Implementation Analyst	3%
<b>Total:</b>		<b>100%</b>

<sup>20</sup> *Id.*

Effort Distribution by Project Roles		
Phase of the PDLC	Team Role	Percentage of Effort by Role

Table 15 - Effort Distribution by Project Roles

Thus, for Step Seven, I allocated the PHE, for the analyzed products, to the appropriate team member Roles within each phase of the PDLC. The PHE distributed across the PDLC, as identified in Step Six, along with the standard team member effort allocation distribution percentages (ORCLX-PIN-000065 Table 15), were used as the input for distributing the PHE across the Roles. Below, Tables 16 (ORCLX-PIN-000065 Table 16) display the PHE distributed across the development team Roles, for each software product version.

Productive Hours of Effort Distributed by Project Roles			
Phase of the PDLC	Team Role	JDE EnterpriseOne Version 8.12	PeopleSoft Version 8.X
1. Plan	Program Manager	48,349	76,747 Hours
	Project Manager	145,047	230,240 Hours
	Quality Manager	32,233	51,164 Hours
2. Specify	Product Manager	48,349	76,747 Hours
	Business Analyst	128,930	204,658 Hours
3. Design	Technical Leader	96,698	153,493 Hours
	Technical Architect	96,698	153,493 Hours
	Technical Analyst	145,047	230,240 Hours
4. Build	Developer	338,442	537,227 Hours
	Database Administrator	145,047	230,240 Hours
	Configuration Manager	64,465	102,329 Hours
5. Test	Test Lead	48,349	76,747 Hours
	Tester	112,814	179,076 Hours
	Quality Auditor	32,233	51,164 Hours
6. Localize / Document	Documentation Lead	87,372	262,393 Hours
	Technical Writer	407,738	1,224,502 Hours
	Curriculum Developer	87,372	262,393 Hours
7. Deploy	Implementation Lead	32,233	51,164 Hours
	Functional Lead	48,349	76,747 Hours
	Implementation Analyst	48,349	76,747 Hours
<b>Totals:</b>		<b>2,194,111</b>	<b>4,307,511 Hours</b>

Table 16 - Effort Distribution by Project Roles

Based on these effort estimates, it would require a team of more than 1,881 people, to be dedicated on a full-time basis (1,728 productive hours per year), for a period of not less than two years, to complete the development effort for JD Edwards EnterpriseOne and PeopleSoft. Further, while a two year period would likely be the goal of SAP TN if undertaking this effort (because a longer development time would impact the business feasibility of the project overall),

the complexity of identifying and organizing a development team of this size makes achieving these results in this timeframe extremely challenging, which in turn highlights the risk that would be involved in undertaking this process.

#### **H. Step Eight: Derive the Cost of Localization and Documentation Translation**

Step Eight involves deriving the cost of localization and documentation of the User Documentation. For the analyzed products to be viable for use by global clients there must be an acceptable level of User Documentation that has been localized and translated into 21 languages, as specified in the installation manuals for each product. To determine the number of pages of User Documentation that must be translated into non-English languages, I applied the metric of 2.14 pages (the low end of the scale) of User Documentation.<sup>21</sup> This assumes that Support Documentation can remain in English, and does not need to be translated into any other languages.

To determine the cost for translating the User Documentation (from English) into a number of different languages, I applied the most conservative translation cost of \$60 per every 1,000 words (\$15.00 per page).<sup>22</sup> Although a number of published sources cite the minimum translation costs to be \$169 per 1,000 words, I have experienced situations in which high volume translation costs can be negotiated down to a lower cost per word, so I have elected to use this most conservative measure. Below, Table 17 (ORCLX-PIN-000065 Table 17) shows the published costs for translating documentation from English to a variety of target languages.<sup>23</sup>

---

<sup>21</sup> Estimating Software Effort, SoftwareMetrics.com website, <http://www.softwaremetrics.com/Articles/estimating.htm>. [ORCLX-PIN-000005]

<sup>22</sup> Generally Accepted Number of Words per Page, Google.com website, October 2009, generally accepted number of words per page of documentation, posed to Google Search engine, <http://answers.google.com/answers/threadview?id=608972>. [ORCLX-PIN-000023]

<sup>23</sup> Competitive Translation Price List, www.hll.co.uk Internet site, January 2009; for document translation rates from English, <http://www.hll.co.uk/rates-list.aspx>. [ORCLX-PIN-000020]



Cost for Localizing and Translating English Documentation into Foreign Languages			
Target Language	Cost of Localization/Translation from English (based on 250 words per page)	High Volume Discount	Discounted Cost per Page
French, Italian, Spanish, and German	\$169 / 1,000 words (4-pages)	\$60 / 1,000 words (4-pages)	\$15.00
Portuguese, Polish, Russian, Ukrainian, Czech, Slovak	\$176 / 1,000 words (4-pages)		
Danish, Dutch, Norwegian, and Swedish	\$216 / 1,000 words (4-pages)		
Albanian, Bulgarian, Estonian, Hungarian, Latvian, Lithuanian, Romanian, Serbo-Croat, Slovene	\$198 / 1,000 words (4-pages)		
Chinese, Japanese, Arabic, Greek, Finnish, Hebrew, Maltese, Turkish	\$242 / 1,000 words (4-pages)		
Bengali, Farsi, Gujarati, Hindi, Punjabi, Urdu	\$263 / 1,000 words (4-pages)		
Indonesian, Kurdish, Malay, Tagalog	\$285 / 1,000 words (4-pages)		
Korean, Thai	\$364 / 1,000 words (4-pages)		

Table 17 - Cost for Localization and Translation

Thus, in Step Eight, I applied a set of localization and translation metrics to the number of pages of documentation, to develop the estimated cost for localizing the required User Documentation into 21 non-English languages.

The number of pages of documentation (in English), as identified in Step Four, along with the selected metric for translating English documentation into non-English languages (ORCLX-PIN-000065 Table 17), were used as the inputs for determining the documentation localization and translation costs. Below, Table 18 (ORCLX-PIN-000065 Table 18) displays the costs associated with localizing and translating the required documentation into 21 languages for each of software product version.

Cost of Document Localization and Translation				
Software Product Version	Number of Target Languages	Number of Pages to Localize/Translate from English	Cost per page	Cost of Localization / Translation
JDE EnterpriseOne Version 8.12	21	7,996,318	\$15.00	\$119,944,769
PeopleSoft Version 8.X	21	24,014,241	\$15.00	\$360,213,614

Table 18 - Cost of Localization and Translation

## **I. Step Nine: Apply Hourly Rates to Determine the Development Costs**

To place a monetary value on the level of effort required to develop the analyzed products, a series of rate cards were applied. These rate cards display the Roles that are involved throughout the PDLC, as described in Step Seven, and their associated fair market hourly rates. Four staffing scenarios were analyzed to adequately appreciate the potential range of development costs by considering a variety of options:

- Offshore: The product development effort would be entirely outsourced to an offshore system development company, typically located in India.<sup>24</sup>
- On-staff: The product development effort would be entirely staffed with full-time employees that reside in Bryan, Texas, the location of SAP TN.<sup>25</sup>
- Outsourced to U.S.-based System Integrator: The product development effort would be entirely outsourced to a U.S.-based system development company that has the required experience and knowledge with JD Edwards EnterpriseOne and/or PeopleSoft.<sup>26</sup>
- Outsourced to Oracle Consulting: The product development effort would be entirely outsourced to Oracle Consultants or to Consultants with expertise allowing them to charge fees consistent with Oracle's Consulting Rates.<sup>27</sup>

In Step Nine, I applied a set of standard hourly rates to each team member's role, to develop the estimated cost of development, by Role. The PHE by Role, as identified in Step Seven, and the documentation translation costs in Step Eight, along with a series of rate cards by role (ORCLX-PIN-000065 Table 19), were used as the input for determining the development

---

<sup>24</sup> Offshore Consulting Rates, www.cyberadsstudio.com Internet site, October 2009, <http://www.cyberadsstudio.com/globalstaffing/talentandrates.shtml>. [ORCLX-PIN-000011]

<sup>25</sup> Salary Wizard, www.salary.com internet site, October 2009, for project team members located in Bryan, Texas (zip code 77801), <http://swz.salary.com>. [ORCLX-PIN-000054]

<sup>26</sup> GSA Schedule Rate Card, www.gsaadvantage.gov website, Alicon Group Inc. pricing catalog, page 18, 2009, [https://www.gsaadvantage.gov/ref\\_text/GS35F0345U/0GGNHT.203VNA\\_GS-35F-0345U\\_ALICONFSSPRICELIST.PDF](https://www.gsaadvantage.gov/ref_text/GS35F0345U/0GGNHT.203VNA_GS-35F-0345U_ALICONFSSPRICELIST.PDF). [ORCLX-PIN-000022]

<sup>27</sup> Oracle Consulting International Business Rates [ORCLX-PIN-000002].

costs. Below, Table 19 (ORCLX-PIN-000065 Table 19) displays the Roles and hourly rates for each of the four scenarios.

Hourly Rates by Role					
Phase of the PDLC	Team Role	Offshore	On-staff in Bryan Texas	U.S.-based Oracle SI	Oracle Consulting Rates
1. Plan	Program Manager	\$95	\$88	\$188	\$484
	Project Manager	\$95	\$85	\$140	\$440
	Quality Manager	\$95	\$67	\$134	\$440
2. Specify	Product Manager	\$95	\$86	\$188	\$380
	Business Analyst	\$85	\$78	\$116	\$308
3. Design	Technical Leader	\$85	\$74	\$140	\$352
	Technical Architect	\$75	\$70	\$111	\$352
	Technical Analyst	\$26	\$37	\$96	\$253
4. Build	Developer	\$26	\$52	\$101	\$220
	Database Administrator	\$26	\$66	\$101	\$253
	Configuration Manager	\$26	\$37	\$159	\$352
5. Test	Test Lead	\$30	\$68	\$96	\$308
	Tester	\$26	\$47	\$96	\$165
	Quality Auditor	\$22	\$58	\$96	\$253
6. Localize / Document	Documentation Lead	\$30	\$46	\$135	\$352
	Technical Writer	\$26	\$32	\$96	\$220
	Curriculum Developer	\$22	\$41	\$96	\$352
7. Deploy	Implementation Lead	\$95	\$74	\$135	\$352
	Functional Lead	\$85	\$59	\$116	\$308
	Implementation Analyst	\$75	\$58	\$101	\$253

Table 19 - Rate Cards

Based on my industry experience, given the number of personnel that would be required, it would not be feasible to pursue only one of the defined staffing scenarios, because no single scenario could provide an effective balance between quality, productivity, and cost, while producing the analyzed products within a reasonable timeframe. To complete the development effort within a two year period, a hybrid staffing model would need to be used, whereby resources would be deployed from all four scenarios, thereby resulting in a fifth scenario. In developing the Hybrid scenario, I applied the Oracle Consulting resources to the Planning Phase, U.S.-based SI resources to the Specify Phase and the Deploy Phase, On-staff in Bryan, Texas resources to the Design Phase and Documentation Phase, and Offshore SI resource to the Build Phase and Test Phase. The Documentation translation costs remained consistent as that these

would be outsourced to a professional organization that specializes in localization and translation.

## 1. JD Edwards EnterpriseOne Development Costs

Below, Table 20 (ORCLX-PIN-000065 Table 20) displays the JD Edwards EnterpriseOne Version 8.12 development costs, associated with the five staffing scenarios.

Total Cost of JDE EnterpriseOne Version 8.12 Development by Team Role						
Phase of the PDLC	Team Role	Offshore	On-staff in Bryan Texas	U.S.-based JDE SI	Oracle/JDE	Hybrid
1. Plan	Program Manager	\$4,593,141	\$4,269,472	\$9,089,584	\$23,400,844	\$23,400,844
	Project Manager	\$13,779,423	\$12,329,796	\$20,306,517	\$63,820,483	\$63,820,483
	Quality Manager	\$3,062,094	\$2,173,777	\$4,319,164	\$14,182,330	\$14,182,330
2. Specify	Product Manager	\$4,593,141	\$4,151,818	\$9,089,584	\$18,348,389	\$9,089,584
	Business Analyst	\$10,959,073	\$10,120,653	\$14,955,911	\$39,710,523	\$14,955,911
3. Design	Technical Leader	\$8,219,305	\$7,203,699	\$13,537,678	\$34,037,591	\$7,203,699
	Technical Architect	\$7,252,328	\$6,759,942	\$10,733,445	\$34,037,591	\$6,759,942
	Technical Analyst	\$3,771,210	\$5,394,506	\$13,924,469	\$36,696,778	\$5,394,506
4. Build	Developer	\$8,799,491	\$17,607,012	\$34,182,638	\$74,457,230	\$8,799,491
	Database Administrator	\$3,771,210	\$9,620,498	\$14,649,702	\$36,696,778	\$3,771,210
	Configuration Manager	\$1,676,093	\$2,397,558	\$10,249,956	\$22,691,727	\$1,676,093
5. Test	Test Lead	\$1,450,466	\$3,301,124	\$4,641,490	\$14,891,446	\$1,450,466
	Tester	\$2,933,164	\$5,260,931	\$10,830,143	\$18,614,308	\$2,933,164
	Quality Auditor	\$709,116	\$1,876,745	\$3,094,326	\$8,154,840	\$709,116
6. Localize / Document	Documentation Lead	\$2,621,171	\$4,061,904	\$11,795,268	\$30,755,069	\$4,061,904
	Technical Writer	\$10,601,179	\$13,125,708	\$39,142,815	\$89,702,284	\$13,125,708
	Curriculum Developer	\$1,922,192	\$3,600,520	\$8,387,746	\$30,755,069	\$3,600,520
	Document Translation	\$119,944,769	\$119,944,769	\$119,944,769	\$119,944,769	\$119,944,769
7. Deploy	Implementation Lead	\$3,062,094	\$2,401,233	\$4,351,397	\$11,345,864	\$4,351,397
	Functional Lead	\$4,109,652	\$2,833,920	\$5,608,467	\$14,891,446	\$5,608,467
	Implementation Analyst	\$3,626,164	\$2,815,117	\$4,883,234	\$12,232,259	\$4,883,234
<b>Totals:</b>		<b>\$221,456,475</b>	<b>\$241,250,703</b>	<b>\$367,718,303</b>	<b>\$749,367,618</b>	<b>\$319,722,837</b>

Table 20 - Cost of Development for JDE EnterpriseOne

## 2. PeopleSoft Development Costs

Below, Table 21 (ORCLX-PIN-000065 Table 21) displays the PeopleSoft Version 8.X development costs, associated with the five staffing scenarios.

Total Cost of PeopleSoft Version 8.X Development by Team Role						
Phase of the PDLC	Team Role	Offshore	On-staff in Bryan Texas	U.S.-based Peoplesoft SI	Oracle/PeopleSoft	Hybrid
1. Plan	Program Manager	\$7,290,932	\$6,777,156	\$14,428,370	\$37,145,379	\$37,145,379
	Project Manager	\$21,872,796	\$19,571,728	\$32,233,593	\$101,305,579	\$101,305,579
	Quality Manager	\$4,860,621	\$3,450,550	\$6,856,034	\$22,512,351	\$22,512,351
2. Specify	Product Manager	\$7,290,932	\$6,590,397	\$14,428,370	\$29,125,354	\$14,428,370
	Business Analyst	\$17,395,908	\$16,065,040	\$23,740,297	\$63,034,583	\$23,740,297
3. Design	Technical Leader	\$13,046,931	\$11,434,807	\$21,489,062	\$54,029,642	\$11,434,807
	Technical Architect	\$11,511,998	\$10,730,408	\$17,037,757	\$54,029,642	\$10,730,408
	Technical Analyst	\$5,986,239	\$8,562,981	\$22,103,035	\$58,250,708	\$8,562,981
4. Build	Developer	\$13,967,890	\$27,948,528	\$54,259,882	\$118,189,842	\$13,967,890
	Database Administrator	\$5,986,239	\$15,271,118	\$23,254,235	\$58,250,708	\$5,986,239
	Configuration Manager	\$2,660,551	\$3,805,769	\$16,270,290	\$36,019,762	\$2,660,551
5. Test	Test Lead	\$2,302,400	\$5,240,046	\$7,367,678	\$23,637,968	\$2,302,400
	Tester	\$4,655,963	\$8,350,951	\$17,191,250	\$29,547,461	\$4,655,963
	Quality Auditor	\$1,125,618	\$2,979,055	\$4,911,786	\$12,944,602	\$1,125,618
6. Localize / Document	Documentation Lead	\$7,871,801	\$12,198,558	\$35,423,105	\$92,362,465	\$12,198,558
	Technical Writer	\$31,837,062	\$39,418,631	\$117,552,228	\$269,390,524	\$39,418,631
	Curriculum Developer	\$5,772,654	\$10,812,945	\$25,189,763	\$92,362,465	\$10,812,945
	Document Translation	\$360,213,614	\$360,213,614	\$360,213,614	\$360,213,614	\$360,213,614
7. Deploy	Implementation Lead	\$4,860,621	\$3,811,602	\$6,907,199	\$18,009,881	\$6,907,199
	Functional Lead	\$6,523,465	\$4,498,429	\$8,902,612	\$23,637,968	\$8,902,612
	Implementation Analyst	\$5,755,999	\$4,468,583	\$7,751,412	\$19,416,903	\$7,751,412
<b>Totals:</b>		<b>\$542,790,232</b>	<b>\$582,200,895</b>	<b>\$837,511,574</b>	<b>\$1,573,417,402</b>	<b>\$706,763,803</b>

Table 21 - Cost of Development for PeopleSoft

## J. Step Ten: Analyze the Estimated Development Costs

For the final step, I evaluated the estimated development costs for the analyzed products, to provide per-unit costing metrics for each of the five staffing scenarios. The development costs for each software product version, as identified in Step Nine, were used as the inputs for analyzing the ultimate development costs. To determine a number of per-unit development costs, I divided the development cost for each software product by its associated contributory components (e.g., number of programs, lines of source code, number of function points). The resulting data provided an analysis of the estimated development costs for each software product version.

Below, Table 22 (ORCLX-PIN-000065 Table 22) displays an analysis of the total development costs for JD Edwards EnterpriseOne Version 8.12, across the five staffing scenarios.

<b>Analysis of Full Life-cycle Product Development Costs for JDE EnterpriseOne Version 8.12</b>					
<b>Cost Category</b>	<b>Offshore</b>	<b>On-staff in Bryan Texas</b>	<b>U.S.-based JDE SI</b>	<b>Oracle/JDE</b>	<b>Hybrid</b>
Cost per program	\$5,732	\$6,245	\$9,518	\$19,397	\$8,276
Per Source Line of Code	\$28	\$31	\$47	\$96	\$41
Per Function Point	\$1,663	\$1,812	\$2,762	\$5,628	\$2,401
<b>Total Cost of Development</b>	<b>\$221,456,475</b>	<b>\$241,250,703</b>	<b>\$367,718,303</b>	<b>\$749,367,618</b>	<b>\$319,722,837</b>

Table 22 - Analysis of JDE EnterpriseOne Development Costs

Below, Table 23 (ORCLX-PIN-000065 Table 23) displays an analysis of the total development costs for PeopleSoft Version 8.X, across the five staffing scenarios.

<b>Analysis of Full Life-cycle Product Development Costs for PeopleSoft Version 8.X</b>					
<b>Cost Category</b>	<b>Offshore</b>	<b>On-staff in Bryan Texas</b>	<b>U.S.-based PeopleSoft SI</b>	<b>Oracle/PeopleSoft</b>	<b>Hybrid</b>
Cost per program	\$31,052	\$33,307	\$47,913	\$90,012	\$40,433
Per Source Line of Code	\$71	\$76	\$109	\$206	\$92
Per Function Point	\$2,105	\$2,258	\$3,248	\$6,101	\$2,927
<b>Total Cost of Development</b>	<b>\$542,790,232</b>	<b>\$582,200,895</b>	<b>\$837,511,574</b>	<b>\$1,573,417,402</b>	<b>\$706,763,803</b>

Table 23 - Analysis of PeopleSoft Development Costs

## VII. ALTERNATE ESTIMATES

While Function Point Analysis represents the most rigorous approach to performing a detailed assessment of the level of effort required to independently develop the analyzed products, I performed an alternate method of estimation, also reliable, as a mechanism for confirming or denying the results of this “bottom-up” analysis, and for estimating the development costs associated with producing JD Edwards World and Siebel.

In support of developing these estimates, I chose to use the Constructive Cost Model (COCOMO), which is also accepted as a valid approach to estimating, but from a “top-down” perspective, as opposed to performing a detailed-level Function Point Analysis.

### A. Constructive Cost Model (COCOMO)

COCOMO is an algorithm-based software cost estimation model that employs the use of regression formulas, coupled with parameters that were derived from historical project characteristics. The model was originally published in 1981, by Barry Boehm, as a method for

estimating the level of effort, project duration, and costs associated with developing software. This original model was referred to as COCOMO 81.<sup>28</sup>

In 2001, the second version of the model, COCOMO II, was published. This recent iteration is better suited for estimating modern software development projects, by providing an updated set of project characteristic that are more aligned with today's software development tools, iterative approaches, and relational databases. The need for this new model was prompted by the evolution of software development technologies, which moved away from mainframe and overnight batch processing, and moved toward desktop development and code reusability.<sup>29</sup>

COCOMO II estimates the software development effort as a function of a limited set of "scaling drivers" that describe the development process, and a set of "cost drivers" that include subjective assessments about the product, platform, personnel, and project attributes. Below, Tables 24 (ORCLX-PIN-000065 Table 24) cites the scaling drivers and their supporting attributes.<sup>30</sup>

COCOMO II Scale Factor - Parameters							
Scaling Drivers	Short Name	Very Low	Low	Nominal	High	Very High	Extra High
Precedentedness	PREC	4.05	3.24	2.43	1.62	0.81	
Development Flexibility	FLEX	6.07	4.86	3.64	2.43	1.21	
Architecture / Risk Resolution	RESL	4.22	3.38	2.53	1.69	0.84	
Team Cohesion	TEAM	4.94	3.95	2.97	1.98	0.99	
Process Maturity	PMAT	4.54	3.64	2.73	1.82	0.91	

Table 24a - COCOMO Parameters

Effort Multipliers - Parameters								
Category	Cost Drivers	Short Name	Very Low	Low	Nominal	High	Very High	Extra High
Product	Required Software Reliability	RELY	0.75	0.88	1	1.15	1.39	
	Database Size	DATA		0.93	1	1.09	1.19	1.29
	Product Complexity	CPLX	0.75	0.88	1	1.15	1.30	1.66
	Required Reusability	RUSE		0.91	1	1.14	1.29	1.49
	Documentation to match lifecycle needs	DOCU	0.89	0.95	1	1.06	1.13	

<sup>28</sup> COCOMO Model II, Center for Systems and Software Engineering, [http://csse.usc.edu/csse/research/COCOMOII/cocomo\\_main.html](http://csse.usc.edu/csse/research/COCOMOII/cocomo_main.html). [ORCLX-PIN-000003]

<sup>29</sup> *Id.*

<sup>30</sup> *Id.*

COCOMO II Scale Factor - Parameters								
Scaling Drivers		Short Name	Very Low	Low	Nominal	High	Very High	Extra High
Platform	Execution Time Constraint	TIME			1	1.11	1.31	1.67
	Main Storage Constraint	STOR			1	1.06	1.21	1.57
	Platform Volatility	PVOL		0.87	1	1.15	1.3	
Personnel	Analyst Capability	ACAP	1.5	1.22	1	0.83	0.67	
	Programmer Capability	PCAP	1.37	1.16	1	0.87	0.74	
	Personnel continuity	PCON	1.24	1.1	1	0.92	0.84	
	Applications Experience	AEXP	1.22	1.1	1	0.89	0.81	
	Platform Experience	PEXP	1.25	1.12	1	0.88	0.81	
	Language and Tool Experience	LTEX	1.22	1.1	1	0.91	0.84	
Project	Use of Software Tools	TOOL	1.24	1.12	1	0.86	0.72	
	Multi-site operation	SITE	1.25	1.1	1	0.92	0.84	0.78
	Required Development Schedule	SCED	1.29	1.1	1	1	1	

Table 24b - COCOMO Parameters

## B. COCOMO II Estimate for JD Edwards EnterpriseOne

In performing this top-down analysis for JD Edwards EnterpriseOne, the 7,774,791 Source Lines of Code (SLOC) (identified pursuant to the procedure described in Section VI.B., above) was used as input, along with my assessment of the characteristics of the JD Edwards EnterpriseOne application, which are annotated in the following tables (ORCLX-PIN-000065 Table 25). I assigned the “Ratings” for the various Effort Drivers in Table 25 below, based on my familiarity with the products and my industry experience.

JD Edwards EnterpriseOne Source Lines of Code	
Number of Source Lines of Code	7,774,791
Number of Source Lines of Code (in 1,000s)	7,775

Table 25a - COCOMO Analysis for JDE EnterpriseOne: SLOC

Scaling Characteristic		
Categories	Assessment	Weighting
Precedentedness	High	1.62
Development Flexibility	High	2.43
Architecture / Risk Resolution	High	1.69
Team Cohesion	High	1.98
Process Maturity	High	1.82
<b>Total:</b>		<b>9.54</b>
<b>Process Scale Factor:</b>		<b>1.1054</b>

Table 25b - COCOMO Analysis for JDE EnterpriseOne: Scaling



Effort Characteristics			
Category	Effort Drivers	Rating	Weighting
Product	Required Software Reliability	High	1.15
	Database Size	High	1.09
	Product Complexity	High	1.15
	Required Reusability	High	1.14
	Documentation to match lifecycle needs	High	1.06
Platform	Execution Time Constraint	Nominal	1
	Main Storage Constraint	Nominal	1
	Platform Volatility	Nominal	1
Personnel	Analyst Capability	Very High	0.67
	Programmer Capability	Very High	0.74
	Personnel continuity	Very High	0.84
	Applications Experience	Very High	0.81
	Platform Experience	Very High	0.81
	Language and Tool Experience	Very High	0.84
Project	Use of Software Tools	High	0.86
	Multi-site operation	High	0.92
	Required Development Schedule	High	1
<b>Overall Weighting Factor:</b>			<b>0.316340143</b>

Table 25c - COCOMO Analysis for JDE EnterpriseOne: Effort

JDE EnterpriseOne Estimated Effort	
Person Months	15,491
Person Hours	2,230,746
Average Blended Rate	\$145.72
Total Cost	\$325,061,334

Table 25d - COCOMO Analysis for JDE EnterpriseOne: Cost

As a result of the performing COCOMO II analysis, the model indicated that the development effort would require 15,491 person-month of effort, or 2,230,746 person-hours of effort. When the number of person hours is multiplied by the average blended rate of \$145.72/hour (identified pursuant to the “Hybrid” staffing scenario for JD Edwards EnterpriseOne discussed in Section VI(I), above), the estimated development cost is calculated to be \$325,061,334. This cost estimate is within 10% of the estimate that was derived for JD Edwards EnterpriseOne through Function Point Analysis using the “Hybrid” staffing scenario (specifically 1.7% higher), and confirms the reasonableness of the estimated development costs associated with JD Edwards EnterpriseOne.

### C. COCOMO II Estimate for PeopleSoft

In performing this top-down analysis for PeopleSoft, the 7,650,493 Source Lines of Code (SLOC) was used as input, along with my assessment of the characteristics of the PeopleSoft application, which are annotated in the following tables (ORCLX-PIN-000065 Table 26). I assigned the “Ratings” for the various Effort Drivers in Table 26 below, based on my familiarity with the products and my industry experience.

PeopleSoft Source Lines of Code	
Number of Source Lines of Code	7,650,493
Number of Source Lines of Code (in 1,000s)	7,650

*Table 26a - COCOMO Analysis for PeopleSoft: SLOC*

Scaling Characteristic		
Categories	Assessment	Weighting
Precedentedness	High	1.62
Development Flexibility	High	2.43
Architecture / Risk Resolution	High	1.69
Team Cohesion	High	1.98
Process Maturity	High	1.82
<b>Total:</b>		<b>9.54</b>
<b>Process Scale Factor:</b>		<b>1.1054</b>

*Table 26b - COCOMO Analysis for PeopleSoft: Scaling*

Effort Characteristics			
Category	Effort Drivers	Rating	Weighting
Product	Required Software Reliability	Very High	1.39
	Database Size	Very High	1.19
	Product Complexity	Very High	1.30
	Required Reusability	Very High	1.29
	Documentation to match lifecycle needs	Very High	1.13
Platform	Execution Time Constraint	Nominal	1
	Main Storage Constraint	Nominal	1
	Platform Volatility	Nominal	1
Personnel	Analyst Capability	Very High	0.67
	Programmer Capability	Very High	0.74
	Personnel continuity	Very High	0.84
	Applications Experience	Very High	0.81
	Platform Experience	Very High	0.81
	Language and Tool Experience	Very High	0.84
Project	Use of Software Tools	High	0.86
	Multi-site operation	High	0.92
	Required Development Schedule	High	1
<b>Overall Weighting Factor:</b>			<b>0.569239041</b>

*Table 26c - COCOMO Analysis for PeopleSoft: Effort*

<b>JDE PeopleSoft Estimated Effort</b>	
Person Months	27,384
Person Hours	3,943,243
Average Blended Rate	\$164.08
Total Cost	\$646,995,805

*Table 26d - COCOMO Analysis for PeopleSoft: Cost*

As a result of the performing COCOMO II analysis, the model indicated that the development effort would require 27,384 person-month of effort, or 3,943,243 person-hours of effort. When the number of person hours is multiplied by the average blended rate of \$164.08/hour (identified pursuant to the “Hybrid” staffing scenario for PeopleSoft discussed in Section VI(I), above), the estimated development cost is calculated to be \$646,995,805. This cost estimate is within 10% of the estimate that was derived for PeopleSoft through Function Point Analysis using the “Hybrid” staffing scenario (specifically, 8.5% lower), and confirms the reasonableness of the estimated development costs associated with PeopleSoft.

#### **D. COCOMO II Estimate for JD Edwards World**

In performing this top-down analysis for JD Edwards World, I assumed that the product had similar functionality to that of JD Edwards EnterpriseOne. This assumption is based on the fact that JD Edwards World was the predecessor to JD Edwards EnterpriseOne, and that it was predominantly developed in the RPG programming language as opposed to COBOL.<sup>31</sup> As a result of this base assumption, I assumed that JD Edwards World contains the same number of SLOC as JD Edwards EnterpriseOne (specifically, 7,774,791 SLOC), as well as similar application characteristics to those found in the JD Edwards EnterpriseOne application, with two modifications. The modifications are associated with Reusability and Platform Volatility stemming from its underlying technology for the product (namely, that JD Edwards World was written in RPG programming language and is run on the IBM I-Series platform), with my assessments annotated in Table 27 (ORCLX-PIN-000065 Table 27), below.

<sup>31</sup> Oracle Indefinitely Extends the life of JDE World, IT Jungle Newsletter, April 24, 2008, by Timothy Prickett Morgan, <http://www.itjungle.com/tfh/tfh042406-story02.html>. [ORCLX-PIN-000010]

JD Edwards World Lines of Code	
Number of Source Lines of Code	7,774,791
Number of Source Lines of Code (in 1,000s)	7,775

Table 27a - COCOMO Analysis for JDE World: SLOC

Scaling Characteristic		
Categories	Assessment	Weighting
Precedentedness	High	1.62
Development Flexibility	High	2.43
Architecture / Risk Resolution	High	1.69
Team Cohesion	High	1.98
Process Maturity	High	1.82
<b>Total:</b>		<b>9.54</b>
<b>Process Scale Factor:</b>		<b>1.1054</b>

Table 27b - COCOMO Analysis for JDE World: Scaling

Effort Characteristics			
Category	Effort Drivers	Rating	Weighting
Product	Required Software Reliability	High	1.15
	Database Size	High	1.09
	Product Complexity	High	1.15
	Required Reusability	Nominal	1
	Documentation to match lifecycle needs	High	1.06
Platform	Execution Time Constraint	Nominal	1
	Main Storage Constraint	Nominal	1
	Platform Volatility	Low	0.87
Personnel	Analyst Capability	Very High	0.67
	Programmer Capability	Very High	0.74
	Personnel continuity	Very High	0.84
	Applications Experience	Very High	0.81
	Platform Experience	Very High	0.81
	Language and Tool Experience	Very High	0.84
Project	Use of Software Tools	High	0.86
	Multi-site operation	High	0.92
	Required Development Schedule	High	1
<b>Overall Weighting Factor:</b>			<b>0.241417477</b>

Table 27c - COCOMO Analysis for JDE World: Effort

JDE World Estimated Effort	
Person Months	11,822
Person Hours	1,702,412
Average Blended Rate	\$145.72
Total Cost	\$248,073,123

Table 27d - COCOMO Analysis for JDE World: Cost

As a result of the performing COCOMO II analysis, the model indicated that the development effort would require 11,822 person-month of effort, or 1,702,412 person-hours of effort. When the number of person hours is multiplied by the average blended rate of

\$145.72/hour, for the “Hybrid” staffing scenario (identified in the Function Point Analysis discussions, above), the estimated cost of development is calculated to be \$248,073,123. In adopting similar proportions to the cost ranges estimated for JD Edwards EnterpriseOne, the JD Edwards World development costs would have ranged between \$172M and \$581M, depending on the selected staffing model.

### **E. COCOMO II Estimate for Siebel**

In performing this top-down analysis for Siebel, I based my analysis on the assumption that the Siebel product contained 79.4% more functionality than the PeopleSoft CRM module, including its use of PeopleTools. This analysis was based on the fact that Siebel contained 7,593 tables (4,435 for SIA and 3,158 for HOR<sup>32</sup>), while PeopleSoft CRM contained 4,233 tables. This method of sizing provides a reasonable, while simplistic, approach to estimating the relative amount of functionality between software products that are built in similar technologies. The reasonableness of this approach is supported by the fact that PeopleSoft CRM was acknowledged as a competitor to Siebel, and that Siebel was acknowledged as the industry leader in the CRM space and offered significantly greater functionality than PeopleSoft CRM.<sup>33</sup> As a result of this analysis, it is estimated that Siebel contains 1,195,091 Source Lines of Code (SLOC), and similar application characteristics to those found in the PeopleSoft, with modifications associated with the Personnel characteristics stemming from the use of a non-integrated development environment (not PeopleCode with PeopleTools), which are annotated in Table 28 (ORCLX-PIN-000065 Table 28), below.

<b>Siebel Source Lines of Code</b>	
Number of Source Lines of Code	1,195,091
Number of Source Lines of Code (in 1,000s)	1,195

*Table 28a - COCOMO Analysis for Siebel: SLOC*

<sup>32</sup> Siebel SIA refers to Siebel Industry Application, while Siebel HOR refers to Siebel’s Horizontal Application. Both are components of Siebel available to customers as part of Siebel’s CRM product. Table numbers are identified in ORCLX-PIN-000004 and ORCLX-PIN-000015.

<sup>33</sup> The Forrester Wave: Enterprise CRM Suites, Q3 2008, by William Band, August 28, 2008, updated September 2, 2008. [ORCLX-PIN-000006]

Scaling Characteristic		
Categories	Assessment	Weighting
Precedentedness	Nominal	2.43
Development Flexibility	Nominal	3.64
Architecture / Risk Resolution	Nominal	2.53
Team Cohesion	Nominal	2.97
Process Maturity	Nominal	2.73
<b>Total:</b>		<b>14.3</b>
<b>Process Scale Factor:</b>		<b>1.153</b>

Table 28b - COCOMO Analysis for Siebel: Scaling

Effort Characteristics			
Category	Effort Drivers	Rating	Weighting
Product	Required Software Reliability	High	1.15
	Database Size	High	1.09
	Product Complexity	High	1.15
	Required Reusability	High	1.14
	Documentation to match lifecycle needs	High	1.06
Platform	Execution Time Constraint	Nominal	1
	Main Storage Constraint	Nominal	1
	Platform Volatility	Nominal	1
Personnel	Analyst Capability	High	0.83
	Programmer Capability	High	0.87
	Personnel continuity	Nominal	1
	Applications Experience	Nominal	1
	Platform Experience	Nominal	1
	Language and Tool Experience	Nominal	1
Project	Use of Software Tools	Nominal	1
	Multi-site operation	Nominal	1
	Required Development Schedule	Nominal	1
<b>Overall Weighting Factor:</b>			<b>1.257854015</b>

Table 28c - COCOMO Analysis for Siebel: Effort

JDE Siebel Estimated Effort	
Person Months	10,890
Person Hours	1,568,203
Average Blended Rate	\$164.08
Total Cost	\$257,306,140

Table 28d - COCOMO Analysis for Siebel: Cost

As a result of the performing COCOMO II analysis, the model indicated that the development effort would require 10,890 person-month of effort, or 1,568,203 person-hours of effort. When the number of person hours is multiplied by the average blended rate of \$164.08/hour, for the Hybrid scenario, the estimated cost of development is calculated to be \$257,306,140. In adopting similar proportions to the cost ranges estimated for PeopleSoft, the

Siebel development costs would have ranged between \$198M and \$573M, depending on the selected staffing model.

## **VIII. RESULTS**

For the foregoing reasons, based on my experience as an I.T. professional specializing in commercialized product development and managed services, and after reviewing the materials in the case, examining the associated intellectual property, and conducting a methodical approach to estimating the associated development costs, my conclusions are as follows:

### **A. Summary of Analysis**

I am highly confident that the cost associated with performing full life-cycle product development for JD Edwards EnterpriseOne Version 8.12, as described in this report, and based on the Hybrid staffing scenario, will be in the area of \$320M, with a range between \$221M and \$749M depending largely on the labor source and associated costs.

I am highly confident that the cost associated with performing full life-cycle product development for PeopleSoft Version 8.X, as described in this report, and based on the Hybrid staffing scenario, will be in the area of \$707M, with a range between \$543M and \$1,573M depending largely on the labor source and associated costs.

I am highly confident that the cost associated with performing full life-cycle product development for JD Edwards World, as described in this report, and based on a similar Hybrid staffing scenario (the same used for JD Edwards EnterpriseOne), will be in the area of \$248M with a range between \$172M and \$581M depending largely on the labor source and associated costs.

I am highly confident that the cost associated with performing full life-cycle product development for Siebel, as described in this report, and based on a Hybrid staffing scenario (the same used for PeopleSoft), will be in the area of \$257M, with a range between \$198M and \$573M depending largely on the labor source and associated costs.

In total, I am highly confident that the cost associated with performing full life-cycle product development for all of the cited products, as described in this report, and based on a

Hybrid staffing scenario, will be in the area of \$1,532M, with a range between \$1,134M and \$3,477M depending largely on the labor source and associated costs.

Based on my complete analysis, it is estimated that 9,772,236 person-hours of productive effort would be required to perform full life-cycle application development for the cited software products (JD Edwards EnterpriseOne, JD Edwards World, PeopleSoft, and Siebel). Assuming there are 144 productive hours in a month, this translates into 67,863 person-months of effort. If the development effort were to be completed within a two-year time frame, the organization would require access to, and the ongoing retention of, more than 2,828 well-trained resources, throughout the 24-month duration of the project

#### **B. Valuation of Independent Development of Infringed Software**

Oracle's Complaint informs me that SAP TN also misused the Oracle database software by using it to support SAP TN customers. I did not qualify the costs of developing the Oracle database software because of time constraints. However, given the costs associated with the development of other software, and given Oracle's more than thirty-year history of development and innovation of database software,<sup>34</sup> I expect the costs Defendants avoided by not developing the Oracle database software themselves are significant. My estimate of Defendants' avoided R&D costs is thus conservative in that it does not include the database-related avoided costs.

#### **IX. REFERENCE MATERIALS**

A list of material considered in generating this report is listed in Appendix B. All Tables are also produced in native form as ORCLX-PIN-000065. Other back-up material is produced as ORCLX-PIN-000063 through ORCLX-PIN-000085. A Glossary of Terms is provided in Appendix C.

---

<sup>34</sup> For example, Oracle's history and development as a provider of database software is discussed in [Oracle Celebrates Thirty Years of Innovation](http://www.oracle.com/oramag/profit/07-may/p27anniv_timeline.pdf), Oracle Magazine, July/August 2007, available at [http://www.oracle.com/oramag/profit/07-may/p27anniv\\_timeline.pdf](http://www.oracle.com/oramag/profit/07-may/p27anniv_timeline.pdf). [ORCLX-PIN-000012]



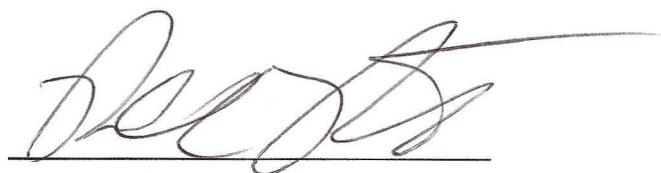
**X. OPTION TO REVISE**

I reserve the right to modify and/or supplement this report and/or the opinions set forth herein if additional damages rulings are made by the Court and/or additional evidence becomes available.

I, Paul C. Pinto, having conducted the aforementioned analysis and having authored this report, confirm that the opinions contained herein represent a fair and unbiased analysis of the facts presented to me.

Date:

11/16/09

A handwritten signature in black ink, appearing to read 'Paul C. Pinto', written over a horizontal line.

Paul C. Pinto