Oracle Corporation et al v. SAP AG et al

Doc. 886 Att. 3

EXHIBIT C

Dockets.Justia.com

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

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.[2] 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

[2] *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.

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.[8]

Stratification by Programming Language	
Software Product Version	**Programming Language Groupings**
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

[8] 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.

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.[9]

[9] 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)		
	Totals:	**56,114**	**Programs/Files**	**56,114**	**Programs/Files**

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.[10] 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.[11]

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.

[10] 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]

[11] *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.[12] 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

[12] *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	
	Totals:	**15,425,284**	**15,425,284 SLOC**

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:[13]

[13] Function Point Counting Practices Manual, Release 4.2, ISBN 0-963-1742-9-0, The International Function Point

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,[34] 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.

[34] For example, Oracle's history and development as a provider of database software is discussed in Oracle Celebrates Thirty Years of Innovation, Oracle Magazine, July/August 2007, available at 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

[illegible]

Paul C. Pinto

APPENDIX B

Appendix B to Expert Report of Paul C. Pinto: Materials Considered

1. Third Amended Complaint for Damages and Injunctive Relief
2. Fourth Amended Complaint for Damages and Injunctive Relief
3. Oracle's 3rd Supplemental Response to TomorrowNow's Interrogatory No. 13
4. December 5, 2008 Deposition of Matthew Bowden
5. January 6, 2009 Deposition of Shai Agassi
6. May 21, 2009 Deposition of Seth Ravin
7. ORCL00485750
8. ORCL00485778
9. ORCL00264053
10. ORCL00264039
11. ORCL00264023
12. ORCL00264027
13. ORCL00264021
14. ORCL00264037
15. ORCL00264024
16. ORCL00264025
17. 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-000001]
18. Oracle Consulting, International Business Rates [ORCLX-PIN-000002]
19. COCOMO Model II, www.csse.usc.edu website, Center for Systems and Software Engineering, http://csse.usc.edu/csse/research/COCOMOII/cocomo_main.html [ORCLX-PIN-000003]
20. Notes from call with Dan Vardell re Siebel Table Numbers [ORCLX-PIN-000004]
21. Estimating Software Effort, SoftwareMetrics.com website, Function Point Analysis and Calculating User Documentation, http://www.softwaremetrics.com/Articles/estimating.htm [ORCLX-PIN-000005]
22. The Forrester Wave: Enterprise CRM Suites, Q3 2008, by William Band, August 28, 2008, Updated September 2, 2008 [ORCLX-PIN-000006]
23. Function Point Counting Practices Manual, Release 4.2, ISBN 0-963-1742-9-0, The International Function Point Users Group, January 2004 [ORCLX-PIN-000007]
24. International Function Point Users Group, website, About IFPUG, http://www.ifpug.org/about/ [ORCLX-PIN-000008]
25. International Standard ISO/IEC, 20926, Manual, October 2003 [ORCLX-PIN-000009]

26. Oracle Indefinitely Extends the Life of JDE World, IT Jungle News Letter, April 24, 2008, Timothy Prickett Morgan, http://www.itjungle.com/tfh/tfh042406-story02.html [ORCLX-PIN-000010]

27. Offshore Consulting Rates, www.cyberadsstudio.com Internet site, October 2009, for project team members provided through an offshore system integration firm, http://www.cyberadsstudio.com/globalstaffing/talentandrates.shtml [ORCLX-PIN-000011]

28. Oracle Celebrates Thirty Years of Innovation, Oracle Magazine, July/August 2007, http://www.oracle.com/oramag/profit/07-may/p27anniv_timeline.pdf [ORCLX-PIN-000012]

29. 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. [ORCLX-PIN-000013]

30. 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]

31. Email from Tanya Ishiguro re PeopleSoft CRM Table Numbers [ORCLX-PIN-000015]

32. Estimating Tech Writing Jobs, Tech-writer.net website article, Estimating Tech Writing Jobs. http://www.tech-writer.net/estimatingtechwritingjobs.html [ORCLX-PIN-000016]

33. 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. [ORCLX-PIN-000017]

34. Software Versioning Definition , Wikipedia, http://en.wikipedia.org/wiki/Software_versioning. [ORCLX-PIN-000018]

35. SPR Programming Languages Table Version PLT2007c, December 28th 2007, SPR Tables, Software Productivity Research, LLC. [ORCLX-PIN-000019]

36. 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]

37. GSA Schedule Rate Card, www.gsaadvantage.gov website, Oracle Corp. pricing catalog, 2009. https://www.gsaadvantage.gov/ref_text/GS35F0009T/0GKOT1.20JKGJ_GS35F0009T_GS-35F-0009T-6-30-2009-575485.PDF [ORCLX-PIN-000021]

38. GSA Schedule Rate Card, www.gsaadvantage.gov website, Alicon Group Inc. pricing catalog, 2009. https://www.gsaadvantage.gov/ref_text/GS35F0345U/0GGNHT.203VNA_GS-35F-0345U_ALICONFSSPRICELIST.PDF [ORCLX-PIN-000022]

39. 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]

40. PeopleCode text file [ORCLX-PIN-000024]
41. PeopleCode text file [ORCLX-PIN-000025]
42. PeopleCode text file [ORCLX-PIN-000026]
43. PeopleCode text file [ORCLX-PIN-000027]
44. PeopleCode text file [ORCLX-PIN-000028]
45. PeopleCode text file [ORCLX-PIN-000029]
46. PeopleCode text file [ORCLX-PIN-000030]
47. PeopleCode text file [ORCLX-PIN-000031]
48. PeopleCode text file [ORCLX-PIN-000032]
49. PeopleCode text file [ORCLX-PIN-000033]
50. PeopleCode text file [ORCLX-PIN-000034]
51. PeopleCode text file [ORCLX-PIN-000035]
52. PeopleCode text file [ORCLX-PIN-000036]
53. PeopleCode text file [ORCLX-PIN-000037]
54. PeopleCode text file [ORCLX-PIN-000038]
55. PeopleCode text file [ORCLX-PIN-000039]
56. PeopleCode text file [ORCLX-PIN-000040]
57. PeopleCode text file [ORCLX-PIN-000041]
58. PeopleCode text file [ORCLX-PIN-000042]
59. PeopleCode text file [ORCLX-PIN-000043]
60. PeopleCode text file [ORCLX-PIN-000044]
61. PeopleCode text file [ORCLX-PIN-000045]
62. PeopleCode text file [ORCLX-PIN-000046]
63. PeopleCode text file [ORCLX-PIN-000047]
64. PeopleCode text file [ORCLX-PIN-000048]
65. PeopleCode text file [ORCLX-PIN-000049]
66. PeopleCode text file [ORCLX-PIN-000050]
67. PeopleCode text file [ORCLX-PIN-000051]
68. PeopleCode text file [ORCLX-PIN-000052]
69. PeopleCode text file [ORCLX-PIN-000053]
70. PeopleCode text file [ORCLX-PIN-000054]

71. PeopleCode text file [ORCLX-PIN-000055]
72. PeopleCode text file [ORCLX-PIN-000056]
73. PeopleCode text file [ORCLX-PIN-000057]
74. PeopleCode text file [ORCLX-PIN-000058]
75. PeopleCode text file [ORCLX-PIN-000059]
76. PeopleCode text file [ORCLX-PIN-000060]
77. PeopleCode text file [ORCLX-PIN-000061]
78. PeopleCode text file [ORCLX-PIN-000062]