EXHIBIT 3

Expert Rebuttal Report Of David P. Garmus

I. Introduction And Summary Of Opinions

As fully documented in this report, I am a universally recognized expert in Function Point Analysis (FPA) and software measurement. Having reviewed the document entitled "Expert Report of Paul C. Pinto" (Pinto Report), I refute any contention that Mr. Pinto applied any recognized form or version of FPA in his report. There is nothing in the Pinto Report that indicated to me that Mr. Pinto understood how Function Points are to be determined, or how they are to be utilized. Mr. Pinto's "Estimating Approach" and his "Ten-Step Analysis To Determine The Cost Of Development Using Function Point Analysis" did not analyze anything using FPA. That "Approach" and that "Analysis" are not recognized steps in any FPA. Consequently, any Function Point value that he achieved through his "Ten-Step Analysis" is totally contrived and without any cognizable basis. The Function Point values he reported are completely fabricated, and his analysis and process would not be recognized by any expert in the field as being a legitimate means of valuing or assessing anything using FPA.

In the Pinto Report, Mr. Pinto purported to estimate "what it would have cost [the Defendants] to independently develop certain software applications."[1] If – for the sake of argument – the Defendants actually were to independently develop all four of the Oracle suites of products recited in the Pinto Report to support TomorrowNow's (TN) customers (as proposed by Mr. Pinto), the newly developed software essentially would have to be an exact replica of the four, recited Oracle suites of products (especially for the purpose of providing most tax updates, bug fixes, etc.). The probability that a software-development project as proposed by Mr. Pinto would result in the creation of four exact replicas of the four Oracle suites of products is essentially zero (i.e., it is essentially impossible). In my opinion, determining the cost for independently developing the four underlying application suites is not appropriate for the case in question.

Further, in my expert opinion, Mr. Pinto should not have attempted to place a value on entire suites of products as TN utilized only a limited percentage of the applications contained in those suites of products.

Mr. Pinto's derivation of productive hours of effort and his estimated development cost are erroneous and baseless, as documented in this report.

[1] See Pinto Report, page 1.

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 in excess of 30 years. I have worked in excess of 20 years specifically in the field of sizing, measurement, and estimation of software application development and maintenance with significant expertise in Function Point Analysis.

I have been a member of the International Function Point Users Group (IFPUG) and a member of the Counting Practices Committee since 1990 (with the exception of three years during which I served as President and Past President of IFPUG). I have during that time period been one of the principal editors of each version of the IFPUG Counting Practices Manual (CPM). I was among the very first individuals recognized as a Certified Function Point Specialist (CFPS). I have been re-certified with every subsequent release of their Counting Practices Manual, and my certification as a CFPS is current at this time. I was also certified by IFPUG as a Certified Software Measurement Specialist (CSMS) at the Third (and highest) Level.

In 1994, I co-founded the David Consulting Group (DCG) together with David Herron. DCG is dedicated to supporting software development organizations in their quest for achieving software excellence. DCG was founded on the principles of strong customer support, quality deliverables, and a professional work ethic. Through the use of software process metrics and methods, we help our clients to use technology to advance their competitive position, enhance the time to market delivery of software, and use industry data to realize best in class practices.

DCG is a specialty software process management consulting firm that helps clients to achieve software excellence with a metric-centered approach. The difference between our approach and others is quite measurable. DCG offers an integrated selection of service areas to its clients including:

- Software Process Measurement – DCG provides clients with the methods, standards, metrics, tools, cost models, and other techniques that are necessary to evaluate, quantify, model, and improve the creation and overall management of software deliverables in both development and maintenance environments. As the industry leader in the Function Point Analysis methodology, DCG helps their clients to manage what they measure.

- Software Process Improvement – DCG provides software process advancement services to clients seeking higher levels of software process maturity. DCG uses the Software Engineering Institute's (SEI) Capability Maturity Model / Capability Maturity Model Integrated (SEI CMM®/CMMI®) as a guide to identifying and implementing best practices that are benchmarked against verified industry data contained in the DCG Industry

Performance and Benchmark Database. With DCG, software delivery performance is improved in quantitative and qualitative terms, leading organizations to improved software processes up to industry best practices levels. DCG is a SEI[sm] approved CMMI® Transition Partner able to perform SCAMPI[sm] Appraisals and CMMI® training.

- Outsourcing Metrics and Governance – DCG provides a comprehensive service to clients engaged in application development and maintenance outsourcing. This service includes all aspects of outsourcing such as opportunity identification, vendor selection, benchmarking, performance monitoring, and contract governance.

Most of my professional engagements have centered on the use of Function Point Analysis as a sizing metric. I co-developed with David Herron, my partner and co-founder of DCG, a Functional Metrics Training Course that was the first such course certified by IFPUG under both Release 3.- and 4.- of the CPM. As indicated in Appendix B, I have written many articles on the use of Function Points and have published two books, together with David Herron, on the process of counting and utilizing Function Points:

- *Measuring The Software Process: A Practical Guide To Functional Measurements*, Prentice Hall, 1996 with David Herron

- *Function Point Analysis; Measurement Practices for Successful Software Projects*, Addison-Wesley, 2001 with David Herron

I have authored another book on Function Points, together with Janet Russac and Royce Edwards, which is due to be released this summer:

- *The Certified Function Point Specialist Examination Guide*, Auerbach Publications/CRC Press with Janet Russac and Royce Edwards, both experts in FPA and members together with David Garmus on the IFPUG Counting Practices Committee

I have been a frequent speaker at IFPUG, SEI, Quality Assurance Institute (QAI), Project Management Institute (PMI), Institute of Electrical and Electronics Engineers (IEEE), and IT Metrics and Productivity Institute (ITMPI) Conferences. I am recognized as one of five IT experts with a personal archive at the ITMPI Website.[2] The site includes various articles, presentations, and webcasts written and presented by me and a schedule of my 2010 speaking engagements for ITMPI.

B. Publications

A list of my publications related to this report is attached as Appendix B.

[2] See http://www.itmpi.org/default.aspx?pageid=510.

C. Compensation

My agreed-upon compensation in this litigation is $300.00/hour. My compensation is not in any way contingent upon the results of my analysis.

D. Prior Testimony

A list of the cases in which I previously have provided testimony as an expert witness is attached as Appendix C.

E. Material Considered

- Pinto Report and Appendices
- PeopleBooks located on TomorrowNow's BU01 Servers: TN-OR02989997, TN (Hard Drive).33 at BU01_G\JDE 36-44\Generic PeopleBooks – JDE; TN-OR02989993, TN (Hard Drive).29 at U01_G\BU01_LOGICAL_IMAGE_43-76\PeopleSoft Enterprise Documentation
- Plaintiff's Fifth Amended and Seventh Supplemental Responses and Objections to Defendant Tomorrow Now, Inc.'s Interrogatory No. 13 (Responses to Interrogatory 13)
- TN-OR06515453
- TN-OR06515454
- TN-OR06515455
- TN-OR07717977
- Appendix L to the Expert Report of Stephen K. Clarke (Clarke Appendix L)
- IFPUG CPM, Releases 4.2, 4.2.1, and 4.3, for each of which I was a principal editor
- IFPUG Website (http://www.ifpug.org/) and miscellaneous materials available on the Website (e.g., http://www.ifpug.org/discus/messages/1751/3800.html; and www.ifpug.org/discus/messages/1751/4699.html)
- David Consulting Group White Papers, Studies, etc. (attached as Appendices E and F)
- Materials prepared or provided by others, as specified in this report
- Sylvan VI Corporate Fact Sheet

III. My Rebuttal Report

In the following report, I am rebutting the arguments and claims made by Mr. Pinto and his approach taken in deriving a value of the four suites of products referenced in the Pinto Report. In particular, I disagree with what Mr. Pinto called his Function Point Analysis for the reasons identified in this report.

Mr. Pinto Did Not Correctly Apply The FPA Methodology

I concur with Mr. Pinto's recognition that "Function Point Analysis provides an objective, comparative measure that assists in the evaluation, planning, management, and control of software production."[3] However, it is my opinion as a recognized industry expert in the field of FPA that Mr. Pinto did not correctly apply the FPA methodology as evidenced by his faulty definitions of FPA terminology and his inaccurate usage of its concepts and measurement techniques.

IV. The FPA Methodology

A. Introduction

Functional size metrics provides a software project manager, an IT organization as well as a business user with a key piece of information (the size) that can be used to improve the effectiveness of how they design, develop, deploy, and maintain software. In addition, FPA enables the sizing of portions of applications (also known as products), which arguably could have been used in this case (rather than sizing the entirety of the four suites of applications, as Mr. Pinto purported to do).

The IFPUG method for FPA is an approved ISO standard and conforms to ISO/IEC 14143-1:2007. The IFPUG methodology measures Functional Size only. Functional Size is a size of the software derived by quantifying the Functional User Requirements.[4]

Functional User Requirements (FURs) are a subset of the User Requirements, requirements that describe what the software shall do, in terms of tasks and services. FURs include but are not limited to:

- Data transfer (for example: sending an invoice for items ordered)
- Data transformation (for example: calculating the cost of the items included in an order when sending an invoice)
- Data storage (for example: storing the invoice)
- Data retrieval (for example: searching for and displaying order information in the invoice)

The result of the above is that one transaction occurs, that of creating and sending an invoice. It does not make any difference how many different steps are involved or how many source lines of code are required; that invoice is counted as one External Output when applying FPA. Actual points (or values) are assigned based upon the number of non-repetitive attributes (fields) on the invoice and the number of logical data files (and not physical files) referenced in the process.

[3] Pinto Report, page 8.

[4] See IFPUG CPM, Release 4.3, Part 2, page 1-3.

B. Intent Of Report

I intend to review the approach that Mr. Pinto used to establish an estimated cost to independently develop the software suites of products allegedly used in the administration of maintenance services provided by TN. My analysis is based upon my industry-wide experience in sizing and estimating software development projects. During the last 20 years, I have worked with numerous clients almost exclusively sizing, measuring, and estimating the development and maintenance of software applications. Most of that effort included the use of FPA. Consequently, I will evaluate whether Mr. Pinto properly used FPA and whether Mr. Pinto's estimated cost values are reasonable.

I disagree with Mr. Pinto's claims concerning "Additional Value," since I do not believe that is relevant to this case.[5] In fact, TN was not attempting to develop a competing suite of products. TN was providing maintenance support for individual applications, for clients that had previously purchased (licensed) the applications from Oracle and their predecessors. It is alleged that TN, on behalf of their clients, downloaded and stored software and support materials in order to provide customer support for clients. Based upon my own experience at DCG client sites, most users of the PeopleSoft, JD Edwards, and Siebel software applications do not utilize much of the functionality included in the applications, and they certainly did not use a substantial portion of the many applications comprising the suites of products. I believe that TN would have needed only some portions of the four suites of products – such as those applications that required revision for adaptive maintenance or corrective maintenance based upon a prior developer defect by Oracle and their predecessors.

I understand that TN was not attempting to sell competing systems. In fact, it is my understanding that TN was supporting many applications (or releases) that were no longer supported by Oracle and their predecessors. This service is much the same as when the reader of this report calls in a local technical support person to fix his/her Dell™ computer with an older operating system such as Microsoft Windows 95™ operating system and an older application such as Microsoft Word 93™ application.

C. Mr. Pinto's Professed Selection Of Function Point Analysis (FPA) And His Misappropriate Expression Of Expertise In The Use Of FPA

It is important to recognize that Mr. Pinto did not utilize or apply any recognized form or version of FPA, and there is nothing in his report that would lead me to conclude that he understands how Function Points are to be determined. He claimed in Section III C of his report that he created an estimated cost of development for JD Edwards EnterpriseOne and PeopleSoft applications, using FPA. He further stated, "I chose to use FPA 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. 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

[5] Pinto Report, pages 6-7.

inception."[6] (I note that IBM, TCS, and Infosys have all been Function Point clients of DCG.). Mr. Pinto went on to state, "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 applications needed to be redeveloped onto a modern computing platform, while maintaining the existing functionality."[7]

Although he states that he intended to apply FPA, he did not do so. In my opinion, no one with Function Point experience would consider that Mr. Pinto applied FPA. Although Mr. Pinto claimed that he had considerable experience applying the required techniques; however, the IFPUG Office stated that they had no record of Mr. Pinto or his company, Sylvan VI, in their database, that Mr. Pinto had no authority to copy or use their copyrighted manual (which he included as ORCLX-PIN-000007), that Mr. Pinto had never been certified and that there was no record of any communication at any time between IFPUG and Mr. Pinto or Sylvan VI.[8] Furthermore, neither Mr. Pinto's curriculum vitae nor the public Sylvan VI Corporate Fact Sheet found on the Internet refers to any experience with FPA.

D. A Brief Description On How To Correctly Apply FPA

FPA has been used successfully to measure software size and, as a result, to determine delivery rates and quality metrics. It is a synthetic method, much the same as BTUs or temperature, which provides a methodology to calculate the relative size of individual components, applications, or subsystems.

A size measure must be able to accurately quantify the functionality (value) being delivered to the business user. When a user specifies a desired functionality, the size must measure that functionality in a direct way; e.g., the user asks for one widget, and that is exactly what gets measured.

FPA has proven to be an effective way to establish a meaningful size measure and can be used to establish baseline costs and performance level monitors. FPA centers on its ability to measure the size of any software deliverable in logical, user-oriented terms. *Rather than counting lines of code, FPA measures the functionality being delivered to the end user.* Appendix D contains a copy of the IFPUG CPM, Version 4.2 Function Point Counting Guidelines published by DCG and approved by IFPUG.

FPA evaluates the software deliverable and measures its size based on well-defined functional characteristics. It essentially accounts for:

- Data that is entering a system: external inputs (EIs) such as logical transaction inputs or system feeds.

[6] Pinto Report, page 8.
[7] Pinto Report, page 8.
[8] Phone conversation between Mr. David Garmus and Mr. Christopher Decker, Assistant Association Manager of the IFPUG Office on February 1, 2010

- Data that is leaving the system: external outputs (EOs) or external inquiries (EQs) such as online displays, reports or feeds to other systems.
- Data that is manufactured and stored within the system: internal logical files (ILFs) such as logical groups of user defined data.
- Data that is maintained within a different system but is necessary to satisfy a particular process requirement: external interfaces (EIFs) such as interfaces to other systems.

These five functional elements are assessed based on their complexity and used to determine a Function Point (FP) count. Low, average, and high ILFs, EIFs, EIs, EOs, and EQs are totaled using IFPUG matrices. Values for each category are calculated using the standard weights shown in Table 1 below to determine the total number of Function Points.

Table 1. Function Point Counting

Function Point Counting Weights

Type	Low	Avg	High	Total
EI	__ x 3 +	__ x 4 +	__ x 6 =	___
EO	__ x 4 +	__ x 5 +	__ x 7 =	___
EQ	__ x 3 +	__ x 4 +	__ x 6 =	___
ILF	__ x 7 +	__ x10 +	__ x15 =	___
EIF	__ x 5 +	__ x 7 +	__ x10 =	___

ILF And EIF Complexity Matrix

RETs	1-19 DETs	20-50 DETs	51+ DETs
1	Low	Low	Avg
2-5	Low	Avg	High
6+	Avg	High	High

EI Complexity Matrix

FTRs	1-4 DETs	5-15 DETs	16+ DETs
0-1	Low	Low	Avg
2	Low	Avg	High
3+	Avg	High	High

EO And EQ* Complexity Matrix

FTRs	1-5 DETs	6-19 DETs	20+ DETs
0-1	Low	Low	Avg

2-3	Low	Avg	High
4+	Avg	High	High

* an EQ must have at least 1 FTR

Note: DETs are equivalent to non-repeated fields or attributes.
RETs are equivalent to mandatory or optional sub-groups.
FTRs are equivalent to ILFs or EIFs referenced by that transaction

The final Function Point calculation yields a single number that represents the total amount of functionality being delivered. Once completed, the Function Point size of an application or a new development project can be communicated in a variety of ways. As a stand-alone value, the Function Point size of a system tells us the size of the overall software deliverable. This could be reflected for a previously developed application or for a development or enhancement project. During the years 2003 to 2007, costs were often estimated for development projects at a value of $400 to $1,200 per Function Point; costs varied significantly based upon the development location, which is why outsourcing and off-shore development were so popular then and remain so today.

E. Mr. Pinto's Analysis Of JD Edwards World And Siebel

Mr. Pinto claims that he was able to extrapolate estimates for JD Edwards World and Siebel suites of products from the JD Edwards EnterpriseOne and PeopleSoft estimates, but he does not provide evidence of his methodology for doing so accurately. As I have already discussed, he did not apply FPA to size any application, so he should not have extrapolated from his faulty FPA-derived estimates. Mr. Pinto then purports to provide estimates for all four suites of products using what he calls a COCOMO analysis. Donald J. Reifer, another software-estimation expert engaged by the Defendants, is a recognized COCOMO expert, and, in his expert report, Mr. Reifer comments on Mr. Pinto's COCOMO analysis. I do note that neither Mr. Pinto's curriculum vitae nor the public Sylvan VI Corporate Fact Sheet found on the Internet refers to any experience with COCOMO.

F. Mr. Pinto Inappropriately Sized Applications Not Maintained By TN

Although Mr. Pinto clearly states that cumulative development costs would amount to double counting, his estimations derived from both his so-called FPA and COCOMO analysis significantly over-valued Source Lines of Code carried over from previous versions of each application as well as reused code.[9] He also sized applications and components within applications that were not utilized by TN in the course of TN's business. This was true for substantial numbers of applications included in Mr. Pinto's analysis; apparently, he did not attempt to eliminate the significant over-counting of Source Lines of Code. TN provided maintenance services for only a limited number of the many applications comprising the four suites of products sized / valued by Mr. Pinto.

[9] With respect to COCOMO issues, see Expert Report of Donald J. Reifer.

Moreover, in my experience, most users of the PeopleSoft, JD Edwards, and Siebel software do not utilize much of the functionality included in the applications.

I determined by reviewing the spreadsheets of TN's maintenance activities[10] that many of the applications (sometimes referred to as modules) identified in Plaintiff's Fifth Amended and Seventh Supplemental Responses and Objections to Defendant Tomorrow Now, Inc.'s Interrogatory No. 13 as modules contained within a sample of the suites of products allegedly infringed by TN, were in fact not being used by TN, including:

PeopleSoft HRMS:

Global Payroll for Brazil
Global Payroll for France
Global Payroll for Germany
Global Payroll for India
Global Payroll for Italy
Global Payroll for Spain
Global Payroll for Switzerland
Global Payroll for The Netherlands

PeopleSoft CRM:

Analytics - Customer Value
Analytics - Smart Views
Call Center
CRM for Communications
CRM for Energy
CRM for Financial Services
CRM for Government
CRM for High Technology
CRM for Insurance
Data Transformer
HelpDesk for Employee Self Service
Configurator
FieldService
Multichannel Interactions
Telemarketing

PeopleSoft Financials, Distribution & Manufacturing and/or Financials & Supply Chain Management:

Application Fundamentals
Collaborative Supply Management
Customer Portal
Deduction Management
Engineering
Enterprise Service Automation
eRFQ

[10] See TN-OR06515453, TN-OR06515454, Clarke Appendix L, and Responses to Interrogatory 13.

eStore
Inventory Policy Planning
MarketPay
Mobile Order Management
Remote Order Entry
Order Promising
Pay/Bill Management
Promotions Management
Remote Order Entry
Supplier Portal
Bank Setup and Processing
Databridge
eBid Payment
Global Options and Reports
Grants
Mobile Time and Expense for Palm

PeopleSoft Enterprise Performance Management:

Activity-Based Management
CFO and Governmental Portal Solution Key Performance Indicators
Customer Behavior Modeling
Customer Scorecard
ESA Warehouse
Funds Transfer Pricing
Inventory Policy Planning
Investor Portal
Investor Portal Pack
Profitability Management for the Financial Services Industry
Project Portfolio Management
Risk-Weighted Capital
Sales Incentive Management
Sales Incentive Management for High-Tech & Industrial
Supplier Rating System
Supply Chain Warehouse

PeopleSoft Student Administration Solutions:

Community Access
Community Directory
Contributor Relations
Financial Aid
Gradebook
Involvement
Learner Services
Personal Portfolio
Student Administration Portal Pack

EnterpriseOne 8.12 ALM:

Adv Real Estate Forecasting
Condition-base Maintenance
Commercial Property Management
Equipment Cost Analysis
Resource Assignments

EnterpriseOne 8.12 CRM:

Call Management
CRM Call Scripting
CRM Case Management
CRM Integrated Solution
CRM Sales Force Automation
CRM Solution Advisor
Extensity Integration
JDE E-Procure Intgr for Ariba
JDE Integrators for Siebel
M&D Complementary Product
Mobile Sales
Promotions Management
Siebel - Consumer Goods

EnterpriseOne 8.12 Financial Management:

Argentina
Asia Pacific Localization
Austria
Belgium
Brazil
Chile
Colombia
Conversion Programs
Czech Republic
Denmark
DREAMwriter Conversion Tool
Ecuador
Electronic Mall
EMEA Localization
Enhanced Accounts Receivable
Expense Reimbursement
FASTR Conversion Tool-Columnar
Finance Report Writer - FASTR
Finland
France
General Back Office
Germany
Greece
Hungary
India

Ireland
Italy
Japan
Latin America Localization
M&D Complementary Product
Mexico
Multi-National Products
Netherlands
Norway
Peru
Plant Manager Dashboard
Poland
Portugal
Rapid Start
Russian Federation
Singapore
South Korea
Spain
Sweden
Switzerland
Taiwan
Turkey
United Kingdom
Venezuela
World Writer Conversion Tool

EnterpriseOne 8.12 HCM:

ADP Integrator
HCM Learning Management Integr
New Zealand
Old Payroll
OneWorld HR Canadian
Payroll SUI

EnterpriseOne 8.12 Project Management:

Advanced Order Configurator
Blend Management
Distribution Contracts
Government Contracting
Homebuilder Management
Promotions Management

EnterpriseOne 8.12 MFG & SCM:

DRP/MPS/MRP
M&D Complementary Product
Plant Manager Dashboard

EnterpriseOne 8.12 SM:

Grower Pricing & Payments
Grower Management
Operational Sourcing
Purchase Order Receipts/Routing
Voucher Match and Landed Cost[11]

G. Mr. Pinto's Inappropriate Claim Of Using FPA

In Section IV of his report, Mr. Pinto states that he analyzed the cost of development for the following "products" using FPA:

- 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")

It is important to recognize first that Mr. Pinto did not analyze anything using FPA, including the suites of products listed above. Consequently, any Function Point value he reported is completely fabricated and without substance.

Second, he should not have attempted to place a value on entire suites of products when TN utilized only a limited percentage of the applications contained in those suites of products.

It is certainly possible to count any of the applications contained in the suites of products listed above or to calculate smaller components of those applications involved in adaptive or corrective maintenance. Mr. Pinto did not properly count any of the applications. In order to demonstrate how a FPA actually is performed, I calculated a Function Point count of Accounts Payable by reviewing the EnterpriseOne 8.0 Accounts Payable PeopleBook and a Function Point count of Payroll by reviewing the PeopleSoft Enterprise Global Payroll for United States 8.9 PeopleBook.[12] If there were a reason to

[11] I understand that, in addition, TN also was not using various modules comprising the JD Edwards World and Siebel suites of products, such as Agent Portal, eAdvisor, Release Manager, SOA Enablement, Homebuilder Management, Distributed Data Processing. See TN-OR06515455, TN-OR07717977, Clarke Appendix L and Responses to Interrogatory 13.

[12] See JDE EnterpriseOne 8.0 Accounts Payable PeopleBook Vol. 1, SKU REL8EAP0502V1 May 2002 (TN-OR02989997, TN (Hard Drive).33 at BU01_G\JDE 36-44\Generic PeopleBooks - JDE\OneWorld\8.0\Financial Management\Accounts Payable\EnterpriseOne 8.0 Accounts Payable PeopleBooks\Accounts Payable V1.pdf); JDE EnterpriseOne 8.0 Accounts Payable PeopleBook Vol. 2, SKU REL8EAP0502V2 May 2002 (TN-OR02989997, TN (Hard Drive).33 at BU01_G\JDE 36-44\Generic PeopleBooks - JDE\OneWorld\8.0\Financial Management\Accounts Payable\EnterpriseOne 8.0 Accounts Payable PeopleBooks\Accounts Payable V2.pdf); and PeopleSoft Enterprise Global Payroll for United States 8.9 PeopleBook, SKU HRCS89MP1GPT-B 0405 April 2005 (TN-OR02989993, TN (Hard Drive).29 at BU01_G\BU01_LOGICAL_IMAGE_43-76\PeopleSoft Enterprise Documentation\PeopleSoft

size any applications using FPA, FPA could have been utilized, for example, to determine the size of applications used by TN or even to estimate the size of similar applications far more legitimately and accurately than by using the flawed method used by Mr. Pinto, namely, using Source Lines of Code as a basis. I note that, using FPA, it is also possible to produce an estimate of small changes in the functionality of an application or for a component of an application. Measuring the entirety of the four suites of products (as Mr. Pinto apparently attempted to do) seems to me to be irrelevant and unhelpful in this case.

H. Function Point Methodology Does Not Permit Backfiring

Mr. Pinto stated: "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."[13] Although Mr. Pinto used Source Lines of Code as a proxy for determining Function Points, IFPUG does not recognize this method because of its unreliability. Nowhere on IFPUG's referenced official Website nor in their written documentation does IFPUG permit or accept the practice of backfiring (using the size and complexity of the underlying Source Code) in order to calculate Function Point Counts. In fact, it is quite to the contrary as all previous suggestions made to IFPUG that backfiring be accepted as an alternative method of calculating Function Points have been rejected. Bill Gates of Microsoft stated that: "Measuring Software Productivity by lines of code is like measuring progress on an airplane by how much it weighs."[14]

Although Mr. Pinto used his own unapproved methodology, "Ten-Step Analysis To Determine The Cost Of Development Using Function Point Analysis," to determine a "Function Point value" for JD Edwards EnterpriseOne and PeopleSoft, he rejected it for the JD Edwards World and Siebel suite of products. It is not clear why Mr. Pinto would claim to be using Function Points in one case and then reject his own methodology to measure other systems. I have personally sized a significant number of PeopleSoft, JD Edwards, and Siebel applications at DCG client sites. It would have been a relatively easy task by an experienced counter to have sized in Function Points all of the applicable applications, and an accurate count could have been obtained. Even abbreviated methods in using FPA (e.g., FP Lite™) would have provided a significantly more reliable and well-founded count than the one proposed by Mr. Pinto. Appendix E examines the FP Lite™ methodology as a reasonable alternative to the detailed FPA method that in fact addressed many of the time requirement criticisms that have been levied in the past about FPA. Many Function Point Counters have utilized FP Lite™ or similar abbreviated methods. Mr. Pinto did not use FPA or any recognized abbreviated method.

Enterprise Documentation Libraries\PeopleSoft Enterprise HRMS and Campus Solutions 8.9 Maintenance Pack 1 PeopleBooks\Administering Budgets and Requirements Documentation Set\hrcs89mp1gpt-b0405.pdf). It is my understanding that these two applications are among the larger applications in terms of functional size. Although the FPA here is, I believe, illustrative generally of the overall process, it is not proper to extrapolate Function Point counts to any unlike applications in any of the suites of products. Proper FPA must be conducted on an application-by-application basis.

[13] Pinto Report, page 11.

[14] See http://c2.com/cgi/wiki?LinesOfCode

In my opinion, the accuracy of backfiring has widely varying results, and that range can occur only after lines of code have been specifically calibrated by application type, application complexity, reuse, source code language, and other factors to Function Points at the specific organization where it is being used and then only to determine the size of their overall portfolio. Mr. Pinto's count is a derived count. In my own experience, I have seen many derived counts that are a thousand times too high. Much more preferred abbreviated methods for sizing are addressed in Appendix F.

V. Mr. Pinto's "Ten-Step Analysis To Determine The Cost Of Development Using FPA"

Mr. Pinto's "Ten-Step Analysis" is not recognized industry practice. Consequently, any Function Point value he achieved through this Ten-Step Analysis is totally contrived and cannot be considered as being the result of FPA.[15]

The following are steps recognized by IFPUG in conducting FPA:

- Determine counting scope & boundary
- Identify functional user requirements
- Measure data functions
- Measure transactional functions
- Calculate the functional size (in Function Points)

Application Function Point counts measure installed applications. They evaluate the current functionality provided to end users by an application. An application is a cohesive collection of automated procedures and data supporting a business objective; it consists of one or more components, modules, or subsystems. Examples of applications include General Ledger, Accounts Payable, Accounts Receivable, Payroll, etc. Another example of an application breakdown within Microsoft Office™ would include MS Word™, MS Excel™, MS Access™, and MS PowerPoint™. An application's functional size is a measure of the functionality that an application provides to the user, determined by the Application Function Point count. This size provides a measure of the current functions the application provides the user. It is initialized when the Development Project Function Point count is completed and is updated every time a completed enhancement project alters the application's functions.

Function Point counts are calculated by application; however, Mr. Pinto incorrectly attempted to develop Function Point counts globally. In this case, since TN used only a limited percentage of the applications contained in the suites of products identified by Mr. Pinto, if there is any merit in conducting any FPA in this case, then it surely should have been done at the application level, not the global level chosen by Mr. Pinto. Mr. Pinto incorrectly attempted to assign a development cost for the entire suites of products.

[15] See Pinto Report, pages 13-34.

For purposes of demonstrating that a proper FPA could have been conducted with respect to the individual applications that comprise the four suites of products, I completed Application Function Point counts of Accounts Payable and Global Payroll for United States (EnterpriseOne 8.0).

I, as a recognized expert in the area in software measurement and estimation and particularly in FPA, have never encountered his "Ten-Step Analysis To Determine The Cost Of Development Using Function Point Analysis." I certainly do not agree in any way with its use in determining the cost of development. I also do not agree that the cost of developing entire similar suites of products is appropriate in a situation where an organization is not selling or developing a competing suite of products.

A. Mr. Pinto Improperly Applied FPA

IFPUG does not agree with Mr. Pinto's Step One: Identify And Group Source Code Components. Mr. Pinto did not apply FPA and incorrectly defined FPA, particularly as it applies to components and Source Code. Instead, Mr. Pinto used backfiring. IFPUG does not concur with the use of backfiring to extrapolate Function Points from underlying Source Code. I also understand that Mr. Reifer addresses in his expert report the issue of whether Mr. Pinto's Source Lines of Code counts are correct. Irrespective of whether they are incorrect (something I leave to Mr. Reifer to discuss), they in any event should not have been used as the basis for backfiring Function Points. IFPUG also does not agree with sizing based upon individual programs or modules; the scope of a Function Point count is based upon analysis of functional aspects of an application or a component of the application. An application boundary is the border between the application being measured and either external applications or the user domain. IFPUG has defined specific rules for identifying boundaries, and they have never included the grouping proposed by Mr. Pinto.

B. IFPUG Membership Does Not Agree With Mr. Pinto's Approach

It is not only IFPUG itself that does not agree with Mr. Pinto's approach. I have retrieved a number of comments below extracted from the IFPUG Website by various members and non-members related to extrapolating Function Points from underlying Source Code. These are not official positions, but they are simply comments submitted by individuals to the IFPUG Bulletin Board on the IFPUG Website:

- Posted on Wednesday, June 13, 2001 - 02:03 pm:

> You probably will not get anybody talking about the benefits of backfiring on this board! The position of the software metrics profession is that it simply does not work. Those tables of backfire coefficients that were published about twenty years ago were meant for very high level statistical discussions, never for simulating a FP count on a single application.
>
> The pitfalls of backfiring are that it has a standard deviation of about 1000%. Just think about assigning a project to a new developer fresh out of programming

school, then giving the identical project to a highly experienced programmer as a benchmark. The beginner will use about ten times as many lines of code as the expert.

Couple that with the fact that the backfire coefficients themselves were never derived very accurately. The figure generally used for Cobol is approximately 100 LOC = 1 FP. Backfire proponents (there are still a few of them around) have admitted that the average has turned out to be more like 300 LOC. And they never talk about the standard deviation in public!

I believe you will receive very few replies that disagree with these points. Function Points were developed so we no longer have to try to derive metrics from lines of code, not so we can glorify LOC and keep using them. LOC based metrics do not work, no matter how they are presented.

- Posted on Sunday, June 17, 2001 - 10:02 am:

I was recently part of a team whose task was to count a system that was believed to be 60,000 function points according to a backfire-based estimate provided by a vendor.

After counting the system was found to be slightly less than 10,000 function points.

- Posted on Wednesday, January 21, 2009 - 10:11 am:

I think this is a perfect example of why backfiring is a dangerous way to derive function points and should be done with care and strong caveats. No two organizations are going to have the same results. In fact I have seen extreme differences within the same development organization (4 to 1 difference in SLOC per FP developing similar applications in the same language). It really comes down to some of the difficulties of using SLOC as a sizing measure. For a more in-depth analysis, check out this article in Crosstalk Magazine:
http://www.stsc.hill.af.mil/crosstalk/2005/04/0504schofield.html

The conclusion of this article included the following:
“The purpose of this article is clear: Statistically significant variation in LOC counts render those counts undesirable for estimating and planning, and deceptive as an accurate portrayer of product size. To those left pondering, "What is a better approach for measuring software size?" despite criticisms, function point analysis, endorsed by International Organization for Standardization/ International Electrotechnical Commission 20926:2003, is used by thousands of companies worldwide to measure software size.”

- Posted on Wednesday, January 21, 2009 - 10:20 am:

Just two cents more to add to this thread, as another view on the reason why relevant deviations could be observed in backfiring applications with a posteriori data: LOCs are a product-level measure of the 'length' of the code, while FP (or another fsu) is a product-level measure of the 'functional size' of a software application.

Thus, different attributes for the software 'product' entity level.
We'd compare apples with oranges...

- Posted on Wednesday, February 23, 2005 - 02:35 am:

In general backfiring is discouraged because the degree of error tends to be so great. A lot has already been said on this board so I'm not going to repeat it. If you decide to go this route, you should not go it alone. A couple of the IFPUG consulting firms have a lot of experience in this area. One is SPR (http://www.spr.com/) which was originally Capers Jones company. Capers Jones is person best known for backfiring; however he's also known for his now famous quote "the use of lines of code metrics for productivity and quality studies to be regarded as professional malpractice starting in 1995."

The other company that has a great deal of experience in backfiring is the David Consulting Group (http://www.davidconsultinggroup.com/). They have an interesting page which discusses Industry Data and Lines of Code at http://www.davidconsultinggroup.com/indata.htm

- Posted on Sunday, April 29, 2007 - 01:36 pm:

---

a general suggestion is to avoid to use "backfiring", in particular if it is not a "controlled backfiring", done on your own historical data.
Two simple reasons:
(1) you cannot control the way a LOC (whatever the programming language) has been defined
(2) IFPUG FPA has not a particular fit for sizing a COTS such as SAP; in any case, a Functional Size Measurement Method (FSMM) can measure only the "functional" side of a software system, not ALL the system. Thus, also if backfiring is applied within COCOMO, the conversion from LOC (that's a length measure based on code) to FP (that's a functional measure based on requirements) is a practice that should be avoided.

You can take a look also @
http://www.geocities.com/lbu_measure/fpa/fpa.htm#p6a for further info on this point.

- Posted on Monday, November 18, 2002 - 09:10 am:

You will not find very many IFPUG members who encourage the use of LOC/FP conversion tables. The reason that FP were invented is precisely because LOC were a notoriously poor unit of measurement.

The conversion tables were created for academic purposes at a time when there simply were not enough real FP counts available to study. They may be useful for estimating the total number of FP on the entire planet, but I would not trust them for any sample smaller than that.

You have already discovered one of the conversion tables' major flaws: inaccuracy. Even the handful of people who find a place for LOC in their toolset have stated that the ratio for Cobol, for example, is probably 300LOC/FP instead of the 100 in the table.

Another major flaw is imprecision. The standard deviation of "backfired" FP,

as they are called, can be greater than the differences you are trying to measure.

I urge you to be extremely cautious about using backfired FP. Metrics derived from them may be misleading.

- Posted on Tuesday, January 07, 2003 - 06:56 pm:

Hi Anand, I have had the same problem as you have and perhaps is due to the lack of time to perform an accurate FP counting.

I agree with Carol and Gene, backfiring is risky, but I had used the following procedure to reduce these risks:

1.- Try not to use the capper jones factors, instead of that I have found another factors which are more updated, you can find this factors in the following page: http://www.qsm.com/FPGearing.html

2.- Because you can't trust only in gearing factors, you could do the following:
2.1 To apply the "the contribution of the ILFs" to the overall application, that means that the you have to perform a counting of the ILFs based on the IFPUG rules and then apply the 22.1% contribution. You can find and example of this in www.isbsg.org.au/html/fpsize.html

2.2 Another way is to perform a "backfiring sampling" and the idea of this is that you determine your own gearing factor. In this case, you will need the entire LOC counting, and select modules with low, average and high functionality.
This modules are counted using the FPA rules, and then the LOC counting has to be performed. An as you can imagine, from the FP counting of the functionality you have selected and with the LOC counting of that functionality, you can derive your own gearing factor.

3.- You have to consolidate the sizes you get by the backfiring and the "ILF contribution".

I have applied this steps in practice and I have found differences of less than 10%.

Of course, don't forget to perform the FP Counting as soon as you can. This steps can be used as a way to get a ballpark estimation, later on you will need the FP counting complete, to track the project, to establish the baseline, to perform an enhancement count, etc.

Luca Santillo, another expert in project management and software measurement, stated: "You should avoid converting Lines of Code to Function Points (and vice versa) because it can introduce strong errors in the estimation process."[16]

[16] "ESE: Enhanced Software Estimation", *IT Measurement, Practical Advice from the Experts*, IFPUG, Addison-Wesley, April 2002

The consistency of these comments relate to the fact that backfiring Function Points from Source Lines of Code or even using Source Lines of Code as an application measure is inappropriate and misleading.

C. Mr. Pinto Has Exhibited Total Unfamiliarity With FPA

Mr. Pinto's totally incorrect definitions of Internal Logical Files (ILFs), External Interface Files (EIFs), External Inputs (EIs), External Outputs (EOs), and External Inquiries (EQs) are evidence of his unfamiliarity with Function Points and FPA. Incorrect definitions invariably result in incorrect and unreliable Function Point counts. Mr. Pinto apparently did not even use his referenced copy of the IFPUG CPM to obtain the correct definitions.

ILFs

Mr. Pinto's definition of an ILF is that it "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."[17]

In contrast, the industry-accepted IFPUG definition of an ILF is: "An internal logical file (ILF) is a user identifiable group of logically related data or control information maintained within the boundary of the application. The primary intent of an ILF is to hold data maintained through one or more elementary processes of the application being counted." [18]

An ILF does not "maintain data"; it is overall data stored by an application and not "within the boundaries of a specific program/module". There is virtually nothing correct in Mr. Pinto's statement that, "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." The functional complexity of an ILF is based upon the number of unique Data Element Types (attributes) and the optional or mandatory subgroups of the ILF.[19]

EIFs

Mr. Pinto's definition of an EIF is that it "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

[17] Pinto Report, page 18.
[18] See the International Function Point Users Group (IFPUG) Function Point Counting Practices Manual (CPM), Release 4.2.1, Part 1, page 6-3 and Part 4, page G-4
[19] See Appendix D, which contains a copy of the International Function Point Users Group (IFPUG) Counting Practices Manual (CPM) Version 4.2 Function Point Counting Guidelines published by The David Consulting Group and approved by IFPUG.

of work."[20]

In contrast, the industry-accepted IFPUG definition of an EIF is: "An external interface file (EIF) is a user identifiable group of logically related data or control information referenced by the application, but maintained within the boundary of another application. The primary intent of an EIF is to hold data referenced through one or more elementary processes within the boundary of the application counted. This means an EIF counted for an application must be in an ILF in another application."[21]

There is virtually nothing correct in Mr. Pinto's statement that, "An 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." An EIF does not ever pass any information outside of a boundary; instead it obtains information maintained within the boundary of another application. The functional complexity of an EIF is based upon the number of unique Data Element Types (attributes) and the number of optional or mandatory subgroup of the EIF.[22]

EIs

Mr. Pinto's definition of an EI is that it "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."[23]

In contrast, the industry-accepted IFPUG definition of an EI is: "An external input (EI) is an elementary process that processes data or control information that comes from outside the application's boundary. The primary intent of an EI is to maintain one or more ILFs and/or to alter the behavior of the system."[24]

Mr. Pinto's statement that, "An EI controls information that is entered into the system" is incorrect; an EI is an elementary process that processes data or control information that comes from outside the application's boundary and maintains one or more ILFs and/or alters the behavior of the system." Mr. Pinto's statement that, "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," is totally incorrect; it is the group of data (and not each piece of data) that crosses the boundary that impacts the relative

[20] Pinto Report, page 18.
[21] See the International Function Point Users Group (IFPUG) Function Point Counting Practices Manual (CPM), Release 4.2.1, Part 1, page 6-3 and Part 4, page G-3
[22] See Appendix D, IFPUG CPM Version 4.2 Function Point Counting Guidelines.
[23] Pinto Report, p. 18.
[24] See the International Function Point Users Group (IFPUG) Function Point Counting Practices Manual (CPM), Release 4.2.1, Part 1, page 7-3 and Part 4, page G-3

functional value of the EI. The functional complexity of an EI is based upon the number of unique Data Element Types (attributes) and logical files referenced and/or updated.[25]

EOs

Mr. Pinto's definition of an EO is that it "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."[26]

In contrast, the industry-accepted IFPUG definition of an EO is: "An external output (EO) is an elementary process that sends data or control information outside the application's boundary. The primary intent of an external output is to present information to a user through processing logic other than, or in addition to, the retrieval of data or control information. The processing logic must contain at least one mathematical formula or calculation, or create derived data. An external output may also maintain one or more ILFs and/or alter the behavior of the system."[27]

Mr. Pinto's statement that, "An EO sends processed information outside of the program/module, so it can be viewed by a User," is incorrect. Data or control information is sent outside the application's boundary, not the program/module, and the data might not be viewed by a user. Mr. Pinto's statement that, "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," has nothing to do with the functional complexity of an EO; the number of calculations in inconsequential. The functional complexity of an EO is based upon the number of unique Data Element Types (attributes) and logical files referenced and/or updated.[28]

EQs

Mr. Pinto's definition of an EQ is that it "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."[29]

In contrast, the industry-accepted IFPUG definition of an EQ is: "An external inquiry (EQ) is an elementary process that sends data or control information outside the application boundary. The primary intent of an external inquiry is to present information to a user through the retrieval of data or control information from an ILF or EIF. The

[25] See Appendix D, IFPUG CPM Version 4.2 Function Point Counting Guidelines.
[26] Pinto Report, page 18.
[27] See the International Function Point Users Group (IFPUG) Function Point Counting Practices Manual (CPM), Release 4.2.1, Part 1, page 7-3 and Part 4, page G-3
[28] See Appendix D, IFPUG CPM Version 4.2 Function Point Counting Guidelines.
[29] Pinto Report, page 18.

processing logic contains no mathematical formulas or calculations, and creates no derived data. No ILF is maintained during the processing, nor is the behavior of the system altered."[30]

Mr. Pinto's definition of an EQ is that it "presents requested information to the User that is retrieved from the database, without any further processing" is partially correct; however, there can be further processing including performing validations, converting equivalent values, filtering data, analyzing conditions, resorting, and rearranging.[31] Mr. Pinto's statement that, "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," is totally incorrect. The functional complexity of an EQ is based upon the number of unique Data Element Types (attributes) and logical files referenced and/or updated.[32] Mr. Pinto's statement that, "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," has been totally fabricated and is totally incorrect.

D. Backfiring Is Not An Acceptable Approach

Mr. Pinto's conversion of Source Lines of Code to Function Points is not recognized by IFPUG as even an alternative methodology. Backfiring is totally unreliable for determining Function Points as it yields widely varying results.

E. IFPUG Does Not Recognize Mr. Pinto's Formula For Determining The Number Of Pages Of Documentation

There has been no recognized or approved study by IFPUG on the conversion of Function Points to Documentation, and the author quoted for Mr. Pinto's formula[33] had previously been expelled from IFPUG for some of his contrary views. There are huge variations in pages of documentation per function point varying from none for some applications to many for government applications. Much documentation is produced within the relative cost per Function Point. In order to provide its maintenance services, TN presumably did not need much of the documentation that Mr. Pinto included in his calculations. Neither, presumably, was there a need or use for the Quick Start Users Guides, Full Users Guides, Feature Users Guides, HELP Text, Training Course Materials, Instructor Guides, Reference Manuals, and Glossaries. In my opinion, TN would have used only the README Files, CD-ROM Information, Operator Guides and Maintenance Guides.

[30] See the International Function Point Users Group (IFPUG) Function Point Counting Practices Manual (CPM), Release 4.2.1, Part 1, page 7-3 and Part 4, page G-3

[31] See the International Function Point Users Group (IFPUG) Function Point Counting Practices Manual (CPM), Release 4.2.1, Part 1, page 7-8

[32] See Appendix D, IFPUG CPM Version 4.2 Function Point Counting Guidelines.

[33] Pinto Report, page 20, footnote 15, citing *Estimating Software Effort*, SoftwareMetrics.com website, http://www.softwaremetrics.com/Articles/estimating.htm.

In my opinion, there is no demonstrable need for the vast amount of translation (from English into some 21 other languages, as proposed by Mr. Pinto), since all of TN's personnel presumably could speak and work in the English language, and the employees of the clients to whom TN was providing its maintenance services presumably also understood English.[34]

F. I Do Not Concur With Mr. Pinto's Derivation of Productive Hours Of Effort Nor With His Determination Of Estimated Development Costs

Not only do I disagree with Mr. Pinto's derivation of Function Point Counts, I also do not agree with Mr. Pinto's derivation of productive hours of effort nor his estimated development cost. The conversion values used to convert Lines of Code into Function Points has never been approved, and the rates utilized are flawed. The delivery rates by Source Code Language used by Mr. Pinto are significantly different from the industry-standards, including those measured by DCG, the largest consulting group in the field of FPA. Many DCG consultants have served as members on the IFPUG Counting Practices Committee as compared to other consulting companies, such as SPR which currently has none and has had only one member since 1992. Mr. Pinto has never been a member of IFPUG or this committee, nor has anyone from his company, Sylvan VI.

In my personal experience at client sites during the years 2003 to 2007 when estimating development costs properly, costs were often estimated for development of this type at a value of $400 to $1,200 per Function Point. Costs varied significantly based upon the development location, which is why outsourcing and off-shore development were so popular then and remain so today. In contrast, Mr. Pinto has reflected a cost between $1,663 and $6,101 per Function Point as reflected in his Tables 22 and 23; such costs were inappropriate for these suites of products at that time or even today. I have no idea how and why he selected the 2009 hourly cost per developers, but Mr. Pinto's cost between $1,663 and $6,101 per Function Point were not realistic for the time period under consideration or even in 2009 or today.

Based on my detailed review of the Pinto Report, and on my own analysis, I believe that Mr. Pinto has significantly over-calculated the number of Function Points, priced development cost per Function Point far too high, and included vastly inflated the number of documentation pages and translation costs.

G. Mr. Pinto's Calculation Of Function Points Is Flawed

Mr. Pinto's calculation of Function Points is not only flawed, but in most cases it is not appropriate. TN was providing maintenance support and was not attempting to sell competing systems. Contrary to Mr. Pinto's suggestion, there was no need to develop the four suites of products in their entirety, with numerous applications, etc. Certainly, clients in this industry do not pay for the total cost of development when they purchase a license for use of applications. Also, based upon my own experience at DCG client sites, most

[34] See Section H, below.

users of the PeopleSoft, JD Edwards, and Siebel software do not utilize much of the functionality included in their applications.

H. There Was No Need To Translate Vast Quantities of Documentation

In my opinion, if TN were to create any documentation at all, it by and large only would have need for the README Files, CD-ROM Information, Operator Guides, and Maintenance Guides. Accordingly, at most, some – but certainly not all – of the documents referenced by Mr. Pinto might have needed to be translated, so Mr. Pinto's estimates requiring translation of extensive libraries of documentation are significantly overstated. However, Mr. Pinto suggests that every document – irrespective of its use or relevance to the particular applications supported by TN – should be translated into 21 languages. For the vast portion of the documentation, there was no need for translation from English, since TN's personnel presumably all spoke and worked in the English language, and their clients presumably were English speaking, at least with respect to technical matters.

VI. MY RESULTS

A. My Summary Of Analysis

In this report, I have demonstrated that Mr. Pinto did not apply any recognized form or version of FPA in his report. I have documented the IFPUG methodology for sizing in Function Points, and I have shown that Mr. Pinto's "Estimating Approach" and his "Ten-Step Analysis To Determine The Cost Of Development Using Function Point Analysis" did not analyze anything using FPA. The Function Point value he achieved through his Ten-Step Analysis is inconsistent with the IFPUG methodology or any approved sizing methodology. These results are completely fabricated and without substance. In addition to being derived by means of his unrecognized, unorthodox and repudiated approach (and thus suspect), they also seem to me to unsubstantiatable and vastly inflated.

Further, in my expert opinion, he should not have attempted to place a value on entire suites of products as TN utilized only a limited percentage of the applications contained in those suites of products.

Moreover, if – for the sake of argument – the Defendants actually were to independently develop all four of the Oracle software application suites recited in the Pinto Report, then, in order for the newly developed software to be completely useful to TN to support its customers, the newly developed software essentially would have to be an exact replica of the four, recited Oracle application suites (especially for the purpose of providing most tax updates, bug fixes, etc.). The probability that the Defendants (or anyone) could create four exact replicas of the four Oracle application suites from scratch is essentially zero (i.e., it is essentially impossible). Accordingly, in my opinion, determining the cost for independently developing the four underlying application suites is not appropriate for the case in question.

Mr. Pinto's derivation of productive hours of effort and his estimated development cost have been developed utilizing faulty logic as documented in this report and should be disregarded.

B. My Function Point Count Of EnterpriseOne 8.0 Accounts Payable

As an exercise to demonstrate how to properly perform an FPA, I analyzed the Accounts Payable module of JD Edwards EnterpriseOne 8.0.[35] I performed this analysis not by using Mr. Pinto's unknown and unconventional Ten-Step process, but instead by utilizing the industry-standard IFPUG FPA methodology, namely:

- Determine counting scope & boundary
- Identify functional user requirements
- Measure data functions
- Measure transactional functions
- Calculate the functional size (in Function Points)

Mr. Pinto's "Ten Steps" – apparently created by him from whole cloth – have no commonality with the above-mentioned FPA methodology. To illustrate the extent to which his "Ten Steps" differ from actual, industry-recognized FPA, I list his "Ten Steps" below:

Step One: Identify and Group Source Code Components
Step Two: Count the Number of Source Lines of Code
Step Three: Determine the Amount of Functionality
Step Four: Determine the Number of Pages of Documentation
Step Five: Derive the Productive Hours of Effort
Step Six: Distribute the Effort across the Product Development Life-Cycle
Step Seven: Allocate Productive Hours of Effort to Team Roles
Step Eight: Derive the Cost of Localization and Documentation Translation
Step Nine: Apply Hourly Rates to Determine the Development Costs
Step Ten: Analyze the Estimated Development Costs

As an internationally recognized expert in FPA, and based on my many years of real-world FPA experience, it is my opinion that following Mr. Pinto's "Ten Steps" would not result in any sort of correct, meaningful, useful, or substantiatable Function Point size.

My results for my FPA of the Accounts Payable module of JD Edwards EnterpriseOne 8.0 can be found below.

[35] This was one of the modules with respect to which sufficient user documentation was provided by Oracle, from which an FPA could be conducted. See TN-OR02989997, TN (Hard Drive).33 at BU01_G\JDE 36-44\Generic PeopleBooks – JDE.

Type	Low	Avg	High	Total
EI	51 x 3 +	42 x 4 +	16 x 6 =	417
EO	3 x 4 +	34 x 5 +	0 x 7 =	182
EQ	25 x 3 +	35 x 4 +	6 x 6 =	251
ILF	7 x 7 +	5 x10 +	3 x15 =	144
EIF	0 x 5 +	0 x 7 +	0 x10 =	0
	Total			**994**
	Value Adjustment Factor			**1.18**
	Total Adjusted Function Points			**1173**

C. My Function Point Count Of PeopleSoft Enterprise Global Payroll For US 8.9

As an exercise to demonstrate how to properly perform an FPA, I analyzed the Global Payroll for the US module of PeopleSoft HRMS 8.9.[36] I performed this analysis not by using Mr. Pinto's unknown and unconventional Ten-Step process, but instead by utilizing the industry-standard IFPUG FPA methodology.

My results for my FPA of the Global Payroll for the US module of PeopleSoft HRMS 8.9 can be found below.

Type	Low	Avg	High	Total
EI	35 x 3 +	78 x 4 +	11 x 6 =	483
EO	2 x 4 +	11 x 5 +	21 x 7 =	210
EQ	31 x 3 +	35 x 4 +	4 x 6 =	257
ILF	5 x 7 +	8 x10 +	5 x15 =	190
EIF	2 x 5 +	8 x 7 +	3 x10 =	96
	Total			**1236**
	Value Adjustment Factor			**1.18**
	Total Adjusted Function Points			**1458**

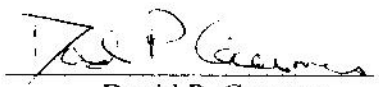
[36] This was one of the modules with respect to which sufficient user documentation was provided by Oracle, from which an FPA could be conducted. See TN-OR02989993, TN (Hard Drive).29 at BU01_G\BU01_LOGICAL_IMAGE_43-76\PeopleSoft Enterprise Documentation.

OPTION TO REVISE AND SIGNATURE

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

I, David P. Garmus, 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: March 26, 2010



David P. Garmus