EXHIBIT G

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.

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.

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.

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

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**

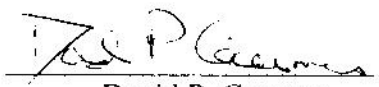
[36] This was one of the modules with respect to which sufficient user documentation was provided by Oracle, from which an FPA could be conducted. See TN-OR02989993, TN (Hard Drive).29 at BU01_G\BU01_LOGICAL_IMAGE_43-76\PeopleSoft Enterprise Documentation.

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