EXHIBIT E

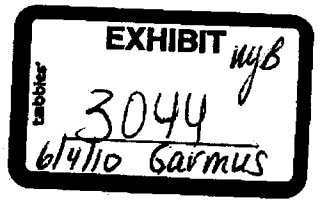
EXHIBIT nyB
tabbies
3044
6/4/10 Garmus

Notes re Response to Rebuttal report of David P. Garmus
Paul Pinto - May 19, 2010 - *Oracle v. SAP*

1. Introduction and summary response

Throughout Mr. Garmus' 29-page rebuttal report, he continuously criticizes my approach, while never actually testing it. He then offers an alternate approach based on a subjective hand-counting process, but never offers an alternative estimated cost for developing the products in question.

As part of my expert report, I provided Mr. Garmus with all information needed to recreate my analysis, a full explanation of my logic, and a working model (spreadsheet) that provided access to all algorithms and calculations that were applied. Defendants also requested that I also provide their Experts with access to the code-counting utilities, and an in-depth description of the computing environment that was used during my analysis, which was promptly provided. The Defense's experts were given everything that was required in order to replicate, test, and confirm or deny my findings, yet the results of such a test were never presented in Mr. Garmus' rebuttal report.

2. Estimating Experience and Right to use IFPUG Material

Mr. Garmus calls into question my credentials as an Estimator, on page 7 of his report, where he writes: "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. 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."

My personal expertise is centered around commercial software, with a focus on estimating and sizing software products. I have served as a senior executive with 2 large-scale commercial software Publishers (Infor and Epicor), as well as within a consulting company that specialized in developing commercial software (NIIT). As such, I have personally been involved in and conducted no less than 100 estimating efforts where I have applied a variety of estimating models and techniques including Function Point analysis and COCOMO analysis. Since 2000, my experience with commercial software includes the sizing and estimating of development, enhancement, and/or maintenance costs associated with the following partial list of commercial software products: Baan, BlackBaud, BPCS, CoKinetic, CRS, CyberLife, E4SE, EnQuesta, Enterprise, Epicor 9, Epiphany, ESRI, Firstwave, Hanson, Infinium, Interworld, ITSM, NSB, Scala, SEI Investments, SoftBrands, Sun Systems, Sytline, System21, TranDotCom, Vantage (CSC), Vantage (Epicor), and Workbrain. I also have considerable experience in estimating and sizing new application development, modernization, enhancement, and maintenance efforts for a variety of internal-use (non-commercial) applications for the following partial list of companies: AIG, Chubb Insurance, Corporate Express, CSFB, Cushman & Wakefield, Earthlink, Guardian Life, ING, KPMG, Lehman Brothers, Nielson Media Research, Office Depot, Sabre Holdings, SEI Investments, Thrivent Financial, TIAA Cref, Utica National, and Woodman of the World.

To assist me in performing my analysis, I engaged a team of professional estimators from NIIT Technologies Inc (niit-tech.com). NIIT is $220M/year, offshore-based consulting company that specializes in software development, system integration, and managed services. NIIT employs over 4,200 consultants and is publically traded on the New York (NIITTECH.NS) and Bombay stock exchanges (NIITTECH.BO). As one of India's top I.T. consulting companies, NIIT carries the following certifications:

- CMMi Level 5 Version 1.2
- People CMM Level 5
- ISO 27001: 2005
- ISO 9001: 2000

- ISO/IEC 20000-1: 2005
- As well as being an active member of IFPUG (membership number 21037 and 21039).

The estimating team that I used, serves as NIIT's internal estimating team, and as such develops over 100 estimates each year, for customer projects that include new application development, modernization, enhancements, and maintenance. During the 5-year period from 2001 to 2006, when I was employed as a Senior Vice President at NIIT, this team operated under my direction for all estimates that were produced for U.S.-based opportunities.

NIIT's estimating team adheres to rigorous standards, and assumes full responsibility for winning business opportunities through aggressive estimating and pricing, while ensuring that the engagement can ultimately be delivered within the established estimates. This team does not develop estimates for the sake of perpetuating estimating methodologies. This team develops estimates with the express purpose of bidding on business opportunities, and ensuring that when an opportunity is won, that it can be delivered within the promised timeline, price, and quality estimates.

As an active member of IFPUG, NIIT cannot be criticized for their use of the cited material in performing their work while engaged on this effort. Moreover, in exercising an abundance of caution, I personally became a member of IFPUG for the membership year of July 1, 2009 through June 30th 2010.

3. Mr. Garmus' interpretation of the goal of producing an exact replica

Mr. Garmus writes, on page 1 of his report: "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 software development project as proposed by Mr. Pinto would results in the creation of four exact replicas of the four Oracle suites of products is essentially zero (i.e., it is essentially impossible)."

Mr, Garmus has misinterpreted my writings, as cited on page 5 of my report, where I note that "In light of SAP TN's use of the underlying JD Edwards, PeopleSoft and Siebel enterprise software applications (in addition to using the fixes, patches, and updates for these applications) in providing support for its customers, I have quantified what it would have cost Defendants to independently create the underlying applications - and not just particular fixes, patches, and updates - for the Oracle products identified herein. "

At no time do I assert that it is prudent for SAP TN to attempt to develop the exact suite of products in questions.

4. Mr. Garmus' Narrow Interpretation of the Scope of Copying

Mr. Garmus writes, on page 6 of his report, "I disagree with Mr. Pinto's claims concerning "Additional Value," since I do not believe that is relevant to this case. 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."

Mr. Garmus contends that I improperly valued all code from HRMS/JDE/Siebel products in my report because "TN provided maintenances services for only a limited number of the many applications comprising the four suites of products [that I] sized/valued."

To determine the scope of my analysis, I worked with counsel to identify the products at issue. I adopted a very conservative posture and limited my analysis to the most current versions of these products in the time period at issue to serve as a proxy for assessing the value of what was stolen.

My analysis does not look at individual modules TN technically supported - or what that even means - because TN holds itself out as able to service everything. It appears that Garmus has focused specifically on what TN has contracted with customers to support. However I understand that TN offered to support everything and I understand that TN possessed more Oracle IP than what they contracted with customers to support. I also understand that other Oracle experts and lay witnesses will testify about issues related to TN's copying of Oracle's IP, and that there will be evidence introduced at trial that will address (among other things) proof that TN took more intellectual property from Oracle than what Mr. Garmus contends TN contracted with customers to support.

I do know, however, that my underlying assumption about the extent of TN's infringing activities was reasonable. First, I understood from Mr. Mandia that TN had copies of Oracle software on its servers reflecting substantial or entire copies of various Oracle product versions and that TN copied and used entire product lines. I also understood that the versions I quantified in my report were versions of these products that TN supported. From my industry expertise I know that each version of a product line is rolled into the subsequent version and improved upon, and thus it was reasonable - although very very conservative - to quantify the latest supported versions.

In response to Mr. Garmus's criticisms, I followed up further to be able to rebut Mr. Garmus's allegations that TN only used certain modules within these product lines. Mr. Garmus incorrectly limits what he says should be quantified to modules that customers contracted with TN to support. TN downloaded materials from customer connection beyond what customers paid TN to support, installed modules on its own servers as part of a customer's environment that exceeded what customers paid TN to support, and through its "retrofit" module used releases on which it was NOT supporting its customers to retrofit fixes for releases on which it was supporting its customers. All of these facts reinforce the reasonableness of my approach and undermine Mr. Garmus' approach, which I understand is contradicted by the facts in this case, or at best irrelevant because TN copied for than it formally contracted to support.

Further, TN "used" all of this copyrighted material in many ways. The reality is that TN was marketed and sold globally and across all products, and thus all of the product lines should be included in license and measured as part of my analysis. It is unrealistic that think that TN would or could have excluded (for example) "Global Payroll for Italy" or "Global Payroll for Spain" from its line of supported products when TN was marketing itself as a global company that offered a comprehensive support solution for its customers. Many of these customers were also global, and generally appear to be sophisticated business entities, and I do not believe they would sign up for support believing that TN could only service a portion of a product. That is undoubtedly why TN marketed itself as able to service the entire product. In addition, TN was trying to gain market share. Companies are not simply not going to switch to a support provider that could support their global payroll for one country and not another, or some of their PSFT CRM modules and not others, and so forth. This is consistent with TN's own marketing material.

In sum, based on my many years of industry experience, I am confident that TN needed to have all of these product lines and not just a subset to sell its services, which did not differentiate by product module and which affirmatively claimed the ability to service the entire product. My position was reinforced by my industry experience, additional discovery materials, and conversation with Juergen Rottler, Head of Oracle Support Services.

✓ See "Discovery Materials" section of New Materials Considered list (PIN-000120)

5. The Ten-Step Process

Mr. Garmus writes, on page 1 of his report, "That Approach and that Analysis are not recognized steps in any FPA. Consequently, any function point value that he (Pinto) achieved through his ten-step analysis is totally contrived

and without any cognizable basis. The Function Point values he (Pinto) reported are completely fabricated, and his (Pinto's) analysis and process would not be recognized by any expert in the field as being a legitimate means of valuing or assessing anything using FPA." He then goes on to outline his 5-step process, on page 16 of his report,

"The following are steps recognized by IFPUG in conducting FPA:

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

Mr. Garmus' 5-steps do not address how an Estimator would convert application size to effort, and effort to cost. His 5-steps do not describe the techniques that are to be employed, and thereby do not address the variety of choices an Expert must make in determining how to size a specific application. The selected techniques should and will vary, based on the availability of application-specific information. In this case, where the source code is available, the use of "backfiring" is the most appropriate technique for determining the size of the application in terms of function points. Mr. Garmus' cited 5-steps outline the activities that are required to perform a manual hand-count of the number of function points, which ignores the existence of the source code, and instead relies on the User Manuals as the sole source of input.

The 10-step process described in my report represents a detailed description of the activities and techniques that are employed in the real-world, on a regular basis. Industry references to the required steps involved in an estimating process, exist at varying levels of detail, and may or may not include the use of specific techniques as part their description. In evidence of this fact, please see an article titled Software Cost Estimation in 2002, by Capers Jones, where he outlines a 10-step process for estimating software costs (PIN000101). In further evidence, please see a whitepaper titled Software Cost Estimation, by Hareton Leung and Zhang Fan, where the authors describe a 7-step process for converting application size into cost estimates (PIN000100). In providing yet further evidence, please see the book titled Software Sizing, Estimation, and Risk Management, by Daniel Golorath and Michael W. Evans, where on pages 35-147, the authors describe a 10-step process, along with a number of iterated sub-processes associate with sizing and developing a cost estimate for software development efforts (TOC included as PIN000102).

Also, Mr. Donald Reifer, another expert retained by the Defense in this matter, has performed an in-depth review of the ten-step process. While Mr. Reifer does not agree with the results I reached while going through the ten-step process, he adopts and uses the process repeatedly without citing any omissions or leaps of faith.

- ✓ Software Cost Estimation in 2002, by Capers Jones, where he outlines a 10-step process (PIN000101)
- ✓ whitepaper titled Software Cost Estimation, by Hareton Leung and Zhang Fan, where the authors describe a 7-step process (PIN000100)
- ✓ book titled Software Sizing, Estimation, and Risk Management, by Daniel Golorath and Michael W. Evans, where on pages 35-147, the authors describe a 10-step process, copy of TOC (PIN000102)

6. Mr. Garmus' assertion of cost per function point

On page 9 of his report, Mr. Garmus writes: "During the years 2003 to 2007, costs were often estimated for development projects at a value of $400 to $1,200 per Function Point", but he provides no insight to the basis for this calculation or any citation associated with its source. In my personal experience, the per function point cost of product development, varies greatly based a number of factors (e.g. cost of resources, project risk, development language...). I have estimated and delivered product development efforts that have exceeded $3,000 per function point.

While there is no single acknowledged source for the cost per function point metric, Capers Jones of SPR provides a benchmark that is based on research derived from 150,000 projects. On slide 17 of his presentation titled: Software Benchmarks , Mr. Jones cites the average, fully-burdened cost per function point to be $1,700, for commercial software development (PIN000105). On slide 18 of a NASA presentation: Function Point Analysis for the NASA CEVGN, the presenter highlights that the cost per function point could be as high as $6,000, depending on the development language. (PIN000114)

In my analysis of the hybrid staffing models for the 4 products in question, I estimated the cost per function point to range between $2,401 and $2,927. In order to provide a like-for-like comparison between the industry metrics and my estimate, I need to remove the costs components that are specifically attributable to localizing and translating the products into 21 languages, because this cost represents an fringe scenario, and would therefore reside outside of the scope of an industry metric. With this component removed from the equation, my estimated cost per function point ranges between $1,435 and $1,501, for the hybrid staffing model.

- ✓ On slide 17 of his presentation titled: Software Benchmarks , Mr. Jones cites the average cost per function point (PIN000105)
- ✓ On slide 18 of Function Point Analysis for the NASA CEVGN (PIN000114)

7. Mr. Garmus' use of IFPUG as a shield

On page 21 of his report, Mr. Garmus' writes "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. Garmus inappropriately takes issue with my paraphrased definitions, which were supplied to provide the lay-person Reader with a common language description of a function point. My paraphrases are reasonable when viewed from this perspective, and the debate about the accuracy of my definitions is unnecessary and immaterial to results of my analysis.

The estimating methodologies provided through IFPUG, as with all reputed estimating methodologies, are intended to be used as a guideline, as opposed to being applied in a rote manner, as Mr. Garmus has asserted. All estimating methodologies are acknowledged as providing a baseline of meaningful thoughts that must then be evaluated for their specific applicability to the situation at hand. No estimating methodology can be viewed, or appropriately implemented as a recipe, without applying the Estimator's knowledge of the domain that is to be estimated. While IFPUG is acknowledged as an esteemed, member-lead organization, it by no means represents the only valid estimating approach.

Given the requirement to assess the R&D costs that were avoided by SAP TN, by stealing rather than constructing the software, it makes sense to approach this effort by using the results of the development process (source code) as the basis, as opposed to viewing that as a new application development effort, which is repeatedly asserted by Mr. Garmus.

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

"The models' ability to weight variables more correctly, to reduce biases, and to produce consistent estimates may consequently have been insufficient to compensate for the low quality of the models and their inability to use all of the relevant contextual information. The software development community is, consequently, still in a position

where the evidence supports neither a replacement of models with expert judgment, nor a replacement of expert judgment with models. If, as suggested in MacDonell and Shepperd (2003), there is a high degree of independence between estimates based on common effort estimation models and expert judgment, and it is difficult to devise rules for selecting the most accurate estimation method, the solution seems to be to use a combination of models and experts."

On page v. of the Function Point Counting Manual, the author writes "This document is meant to be a living one. We must recognize how to count new environments as they are introduced. We need to be able to do this in the context of maintaining the validity of the counts we have already made. This will not be an easy task, yet it is an essential one if we are to be able to measure the progress we are making in delivering value to the users and to the organizations they represent."

- ✓ Page 8 Software Cost Estimation in 2002 (PIN000101)
- ✓ Magne Jørgensen, of the Simula Research Laboratory, in his white paper titled: Forecasting of Software Development Work Effort: Evidence on Expert Judgment and Formal Models (PIN000113)
- ✓ On page v and xi, of the Function Point Counting Manual (PIN0007)

8. The benefits of backfiring

Mr. Garmus denounces the use of "backfiring" on page 15 of his report, where he writes: "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."

IFPUG's position on backfiring is not surprising, given that IFPUG's central source of revenue revolves around training and certifying Specialists to perform hand-counts of function points. If IFPUG were to endorse Backfiring, it would greatly diminish their position with their current subscribers.

Backfiring is commercially recognized, commonly used, and well-recognized as a viable method for determining the size of an application in terms of function points. Given the luxury of having access to application source code, it would be negligent to ignore the actual end-product as the basis for estimating the avoided development costs. By mechanically counting the number of logical source lines of code, through the use of a series of customized code-counting utilities to determine the logical source lines of code, I was able to apply a series of proven backfiring metrics to establish an accurate size as measured in function points. This method of measurement is objective, repeatable, and able to be tested.

As page 5 of Capers Jones' whitepaper: Software Cost Estimation in 2002 explains, "After function points became available in 1978, size could be expressed using either function points or LOC metrics, and converted between the two. As of 2001, sizing is a standard feature in more than 30 commercial software cost estimation tools." (PIN000101)

As further evidence, please see the book titled: Software Sizing, Estimation, and Risk Management, by Daniel Golorath and Michael W. Evans, on page 63, where the authors describe when to Source Lines of Code as an appropriate estimating metric: "SLOC has been the dominant method for sizing complicated, real time or embedded systems and works well for hand-generated systems in general. Use lines of code when SLOC-based historical data exists, when the development organization is comfortable with SLOC estimates, when add-ons to existing systems allow counting of actual SLOC in a system, and as a relatively easy check on other methods. The great strength of SLOC is that it is easy to obtain. All other factors aside, it remains a fairly accurate predictor of development effort... SLOC counts provide a firm indication of the volume of software generated, which is the first critical step for making comparisons and predictions." (PIN000102)

Mr. Garmus professes that he does not recognize backfiring as a viable technique for accurately determining the size of an application as expressed in function points. While, DCG, the company for which Mr. Garmus is a co-founder, has clearly used Backfiring at a number of its clients, as noted in an interview with Mr. Garmus' long-time partner and cofounder of DCG, David Herron. On page 4 of this 2006 Computer Aid interview, Mr. Herron is noted as stating "Another important factor is that lines of code can only be counted when they are available, and that is typically very late in the lifecycle. Because function points measure functionality requested by the user, they can be defined during the very early stages of development. As a result of this, function points can be used as a sizing metric to estimate project performance early in the life cycle. I would never suggest abandoning one measure for the other. In an organization where people are comfortable counting lines of code and they have no need for external comparisons or early lifecycle estimating, there is nothing wrong with what they are doing. Capers Jones has conducted studies that show correlations between function points and lines of code in relation to language complexity. For example, a Java or a C+++ program can be expected to generate a certain number of lines of code per function point. There is a process based on this correlation, known as backfiring, that can be used in high level estimating. We often modify the backfiring approach to capture specific correlations realized at an individual client site. We refer to this process as an approximation backfiring method. Using this method, we are able to accurately estimate the number of function points by taking into account language complexity and lines of code generated in a specific client environment." (PIN000112)

Further evidence of DCG's use of backfiring, is cited on page 4 of a whitepaper titled: A Short History of Lines of Code (LOC) Metrics, where the author writes: "There are tables available from several consulting companies such as David Consulting, Gartner Group, and Software Productivity Research (SPR) that provide values for source code statements per function point for hundreds of programming languages." (PIN000109) For obvious reasons, DCG recommends the use of certified hand-counters as opposed to Backfiring. Yet DCG still provides LOC-to-FP conversion tables, as cited on page 3 and 4 of the DCG Brochure: DCG Software Sizing with Function Points PDF. DCG also goes on to write "backfiring should be used only when sizing an organization's installed applications."

Backfiring is the only reasonable technique that could be applied in estimating the size of the applications in question, any other method, specifically the alternate technique offered by Mr. Garmus, is prone to subjective interpretations, unrepeatable, and difficult to defend.

- ✓ page 5 of Capers Jones' whitepaper: Software Cost Estimation in 2002 (PIN000101)
- ✓ book titled: Software Sizing, Estimation, and Risk Management, by Daniel Golorath and Michael W. Evans, on page 63 (PIN000102)
- ✓ On page 4 of this 2006 Computer Aid interview (PIN000112)
- ✓ page 4 of a whitepaper titled: A Short History of Lines of Code (LOC) Metrics (PIN000109)
- ✓ pages 3 and 4 of DCG Brochure: DCG Software Sizing with Function Points PDF (available online)

11. Productive Hours

On page 1 of Mr. Garmus' report, he writes: "Mr. Pinto's derivation of productive hours of effort and his estimated development cost are erroneous and baseless, as documented in this report," but he never cites an alternate derivation of productive hours, nor does he ever develop an alternate cost estimate. I can only assume that Mr. Garmus is alluding to my use of 144 productive hours per month, as opposed to the IFPUG suggested guideline of 152 productive hours per month. My selection to use 144 productive hours per month is addressed as part of my notes in response to Mr. Reifer's rebuttal report.

12. Mr. Garmus' proposed alternate hand-counting approach

On page 27 of his report, Mr. Garmus writes "As an exercise to demonstrate how to properly perform an FPA, I analyzed the Accounts Payable module of JD Edwards EnterpriseOne 8.0. 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)"

It is simply not viable to estimate each individual module, by reviewing the user documentation, and hand-counting the number of function points. Besides being prone to subjective interpretations of the written word, this method would disallow the Estimator insight into the common functionality (reused code), and an appreciation of the localization and language translation requirements. Mr. Garmus repeatedly attempts to redesign the scope of the estimating effort so as to conform to his proposed alternate approach.

As further evidence, please see the book titled: Software Sizing, Estimation, and Risk Management, by Daniel Golorath and Michael W. Evans, on page 64, where the authors write: "Counting function points is a sophisticated method that cannot be done automatically. There are few shortcuts; you must ensure that it will be done properly by assigning adequately trained and experienced personnel. If you have counts from previous similar projects, be sure to study those counts carefully to ensure that the work performed on your current project in consistent with the method used on those projects...it is very important that the counts be performed correctly and consistently... By using the SEER function-based sizing method, function points can be estimated without conducting detailed counts. Finally, it is important to understand that a combination of SLOC and function-based sizing can be the most appropriate way of estimating the size of an existing system..." (PIN000102)

Mr. Garmus' proposed alternate approach would be worthy of consideration if I was being asked to estimate a new application development effort where I only had access to a set of functional requirements. In reality, I had access to the actual code base that was produced, which is the manifested end result of the development effort. It does not make good sense to extrapolate functional size from the User Documentation, when I can interpolate functional size from the existing source code.

In the early stages of my analysis, I considered hand-counting as an option, but quickly dismissed it. It is simply not viable to accurately and objectively hand-count the function points for all of the applications in question, by using all of the available User Manuals, in 21 languages. This approach would require a huge team of multi-lingual Function Point Specialists, and would result in considerable over-counting of functionality that is common between modules, and common between localized versions of modules.

As a test of Mr. Garmus' proposed alternate approach, I engaged a certified Function Point Specialist, Steve Neundorf from NIIT, to perform the same hand-count, based on the same User Manuals that were used by Mr. Garmus. The results of this analysis are shown below:

Table #1: PIN000108

	EnterpriseOne 8.0 Accounts Payable	**Peoplesoft Global Payroll for U.S. 8.9**
Garmus' Hand-count	1,173 Adjusted Function Points	1,458 Adjusted Function Points
Neuendorf's Hand-count	1,069 Adjusted Function Points	1,682 Adjusted Function Points
Variance	8.9% below Garmus' estimate	15.4% above Garmus' Estimate

As is evident from this data, the alternate approach proposed by Mr. Garmus is indeed subjective, unrepeatable, unable to be defended, and therefore an inappropriate estimating technique to be used in this matter. My conclusion is in alignment with a number of statements that can be found in the book titled: Software Sizing, Estimation, and Risk Management, by Daniel Golorath and Michael W. Evans, on pages 113 - 115, where the authors write: "Function point counts vary with counters...the more experience a counter has, the better he or she

will understand the technique and be able to converge on a reliable estimate...Keep careful track of requirements and resulting function point counts to make sure that counts have not been replicated."

- ✓ book titled: Software Sizing, Estimation, and Risk Management, by Daniel Golorath and Michael W. Evans, on page 64 (PIN000102)
- ✓ book titled: Software Sizing, Estimation, and Risk Management, by Daniel Golorath and Michael W. Evans, on pages 113 – 115 (PIN000102)
- ✓ Table 1 from Surrebuttal Calculations (PIN000108)