Automating the Estimation of Project Size from Software Design Tools Using Modified Function Points.

Jason Ceddia, Martin Dick
School of Computer Science and Software Engineering
Monash University
900 Dandenong Rd. East Caulfield, Melbourne, Australia
jceddia{mdick}@infotech.monash.edu.au

Abstract

Final year students in the Bachelor of Computing complete an industry project where they work in teams to build an IT system for an external client. Grading projects in these circumstances is difficult because of the huge variability of projects and clients. A method of ameliorating some of the variation is to perform a function point count on the projects. Due to the large number of projects and the changing scope of projects a method of automatically counting function points has been devised that uses the output from design tools that students have used. Principally the method counts use cases and database tables. The method has been successful in that no statistical difference in function point counts was found regardless of the implementation environments of systems. However, the first function point count produced during the design phase resulted in values that are lower than expected. The reason for this is that there are omissions from the design. The students will perform another at the user testing stage. The average function point count is 270 with a standard deviation of 130. Currently, the method also assumes that the students are following a traditional waterfall development model. The paper discusses two issues (a) proposing a metric for project size and (b) automating the production of that metric.[1]

1 Introduction

Students enrolled in the Bachelor of Computing at Monash University do an industrial experience project in their third and final year. The students' work in groups of five and each group has an individual project for a client outside the University. A full description can be found in Hagan et.al. (Hagan 1999). This project unit is different from project units taught in some other degrees in that each project is different (Berztiss 1997; Fincher 1998; Daniels 1999; Chan 2001; Chamillard and Braun 2002). Project units are usually graded with a group mark component and an individual mark component. This means that each student in the group may receive a grade that is different from the other team members. Approaches to monitoring individuals are discussed in (Collofello 1999; Utting 1999). Allocating the group mark usually has a component based on the actual product produced (Hagan 1999; Chamillard and Braun 2002). Allocating a grade to these projects is difficult because of their variety and students have expressed concerns that they may be disadvantaged in their final result because they have had a 'difficult' project.

To ameliorate these factors, modified function point counting is being used to help gauge the size of the projects. Software metrics come in many varieties from post project completion, via counting source lines of code to pre project development via function point counting (K.H.Moller and D.J.Paulish 1993; Moller and Paulish 1993; Fenton and S.L.Pfleeger 1997). An overview of methods for estimating project size is (Agarwal, Kumar et al. 2001). However, as noted by Moller and Paulish ."The important point is not the unit used, but the fact that this measurement be well defined and applied consistently." (K.H.Moller and D.J.Paulish 1993) page 40.

The original function point counting method was proposed by Allan Albrecht in 1979 (Albrecht 1979).The current industry standard is that proposed by the International Function Point Users Group (IFPUG) (IFPUG 2000). Many variations of the function point counting method have been proposed (Abiad, Haraty et al. 2000; Hastings and Sajeev 2001; Kusumoto, Imagawa et al. 2002). Kremer has reported on the validity of the Entity-Relationship approach to the IFPUG approach as well as inter-rater comparisons (Kemerer 1993)

While the types of projects the students do typically involve a database, this is not always the case. In the 2003 intake of students, 7 out of 63 projects did not have database functionality. Abiad et al. have proposed a method that determines the function point count based on the objects found in MS-Access type databases (Access is a trademark of the Microsoft Corporation) (Abiad, Haraty et al. 2000). Kusumoto et al have proposed a method of automatically counting function points from Java program source files (Kusumoto, Imagawa et al. 2002). The system proposed in this paper is more generic in that no database need be involved in the project and that any programming environment may be used in the development of the project.

In the past, the unit co-ordinators had broadly scoped projects to a system that included 12 to 15 database tables and 10 to 15 reports. Whilst this was an initial guide to project size, “function creep” could cause the projects’ to significantly increase in size. Virtually all projects experienced a change in scope from the initial specification to the final delivered product. Kemerer reports that the function point counting effort is approximately 1 hour per 100 function points (Kemerer 1993). Given the number of projects that are being dealt with – 63 for the semester 1 intake in 2003 – it is not feasible for the unit co-ordinators to manually count the function points multiple times during the project life.

To help the students and unit co-ordinators keep track of project size a method of automatically counting function points has been devised. We call the modified function point a ‘project point’ (PP). The method uses tools that the students have used in their second year system design unit and database design unit. These tools are Rational Rose and Gershwin respectively (GERSHWIN 1998; RationalSoftware 2002). During the analysis and design phases of the project the students produce use case diagrams via Rational Rose; these use cases are stored in a file that is essentially a text file. The entity relationship diagrams and subsequent database design is produced by Gershwin and stored in a file that is again a text file. By parsing these two files, a count of the number of use cases and tables is obtained. The students are then required to enter values for the various Value Adjustment Factors. An adjusted project point count is then recorded.

Students can repeat this procedure as often as they wish. The system used for project management, called WIER – Web Industrial Experience Resource- stores two sets of values; a first count and a final count. Features of WIER are discussed elsewhere (Ceddia 2001). The first project point count may be repeated twice and the final project point count may be repeated as often as required. The unit co-ordinators recognize that the issue of project complexity – as opposed to size – still needs to be addressed. Project complexity is discussed in (Fenton and S.L.Pfleeger 1997; Hastings and Sajeev 2001). Software size is described by Fenton in Hastings and Sajeev as a function of length, functionality and problem complexity (Hastings and Sajeev 2001). Where students have undertaken projects of high complexity, the unit co-ordinators have had to make allowances by generally reducing the scope of the project. For example, a group was required to develop a web interface for an industrial robot controller. The remainder of the paper is divided into the following sections: section 2 discusses the project point counting method used; section 3 discusses the types of systems encountered in the student projects; section 4 presents the results for the first count of the projects in 2003 and section 5 discusses results and future work.

2 Function Point Count

The IFPUG manual describes five types of fundamental functional elements namely:

External input (EI) - which is a logical transaction where data enters the application

External output (EO) - which is a logical transaction where data exits the application

External inquiry (EQ) - which is a logical transaction where an input requests a response from the application

Internal Logical File (ILF) – which is a logical group of data maintained by the application

External Interface Files (EIF) – which is a logical group of data referenced by the application but maintained by another application (IFPUG 2000).

Hastings comments “Functionality is defined in terms of transactions (EI, EO and EQ) operating on and accessing data (ILF and EIF)”page 60 (Hastings 2000). To differentiate between ILF and EIF there is the notion of a ‘system boundary’. The boundary indicates the border between the application being measured and the external applications or user domains. In the context of student projects nearly all applications are ‘stand alone’ and do not interface to any other applications. This is to be expected, as student projects do not involve mission critical applications. This means that there are no external interface files and all application data is held internally. Further all transactions are of user input or output to the current application.

Our simplified project point count is therefore based on two types of analysis: -

(i) Data from an Entity Relationship diagram, representing internal logical files i.e. each actual entity is one logical file. The complexity of a file/entity is dependant on the number of attributes in each entity. Low complexity =7 (0 to 19 attributes), Average complexity = 10 (20 to 49 attributes) and high complexity = 15 (50 or more attributes). This is the data count (DC). For example, five entities with less than 19 attributes each, give a DC = 5 * 7 = 35. The students are required to produce an Entity Relationship diagram as an artefact of the analysis phase so it can be used in this estimation process. The Entity Relationship diagram is converted into a normalised database design in the design phase of the project; for these systems it is not a difficult transformation.

(ii) Transactions from a Use Case Diagram. Each use case represents one transaction either input or output. Each transaction is modelled as one use case. We will assume an average complexity of 4. This means that a transaction uses two tables (on average) and five to twenty attributes (on average). The Transaction count (TC) is given as: TC = 4 * *no. of cases.* In the automated counting process, all use cases are counted with the exception of use cases that are a generalisation. These use cases normally are developed as placeholders in the diagram and rarely have any functionality of their own.

Each project also has a VAF (value adjustment factor) based on other system characteristics of the project. Each characteristic is given a rating from (0 – not important) to (5 – very important); the rating is called the degree of influence (DI).

The following summarises the fourteen system characteristics to be rated and lists the formula to be used in calculating the unadjusted and adjusted project point count. (IFPUG 2000).

1. Data communications eg web connection
2. Distributed processing eg client/server
3. Performance eg min. response time
4. Heavily used configuration eg set up likely to change often such as dynamic web content
5. Transaction Rates
6. On line data entry
7. Design for end user efficiency
8. Online updates
9. Complex processing eg calculations or lookups
10. Usable in other applications
11. Installation ease
12. Operational Ease
13. Multiple sites
14. Facilitate change

The value for each system characteristic is summed to dive a Total Degree of Influence (TDI); this provides a Value Adjustment Factor of 0 to 70, which is then used in the following formula

Value Adjustment Factor (VAF) = (TDI*0.01)+0.65

This is then used to create the final project point count

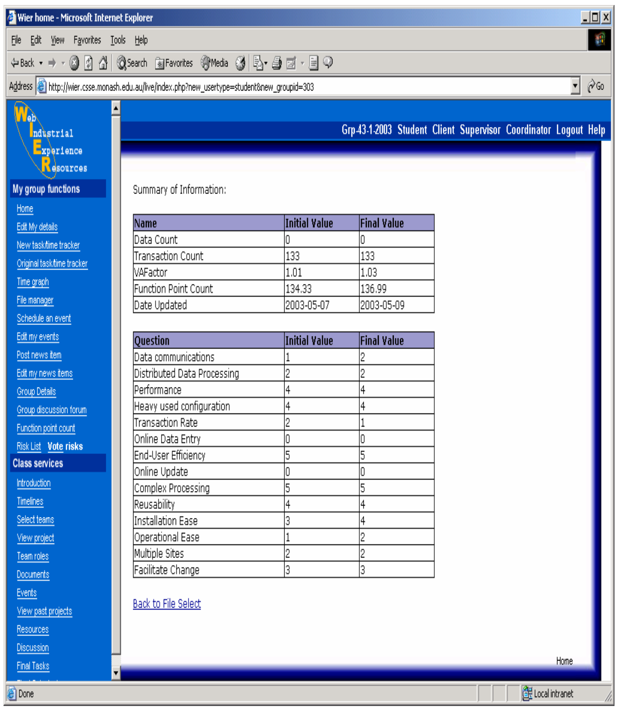
Project Point Count = (DC + TC) * VAF

The output from two system design tools is used to automate the counting process namely the database design tool (GERSHWIN 1998) and the use case design tool (RationalSoftware 2002). Both these products store their representations of the design in files that are essentially text files. As part of the project process, the students are required to upload these two design files to WIER. It is a straightforward matter to parse these files and look for the words “entities” and “object use case” to determine the number of tables and transactions. Note that super use cases are not included in the use case count as this is considered counting the use case twice. The students are then prompted to enter values for the general system characteristics to determine the value adjustment factor.

The students are required to do at least two project point counts at two different stages of the project. The first project point count is done in week 9 of semester 1; this coincides with the delivery of their functional specification document and marks the end of the major design phase. The second project point count is done in week 6 of semester 2 and coincides with the delivery of the beta version of the system. There is a major review of the system design by the project co-ordinators in system walkthroughs. Any errors detected are to be addressed before the system implementation and subsequent second project point count. The intention is to compare the two project point counts and determine whether there has been a change of functionality either by ‘function creep’ or decrease because of unexpected difficulties. A sample output of the two counts is shown in Figure 1. Appendix 1 shows a sample database design and use case diagram for a dynamic web site for a primary school. At the time of writing only the week 9 count has been reported.

3 System Types

In semester 1, 2003, 63 groups started the Industrial Experience project. The types of systems that are being built have been analysed in terms of a number of aspects. Each of these aspects will be used to determine if there is a statistically significant difference between the populations in the next section.



Summary of Information:

Name	Initial Value	Final Value
Data Count	0	0
Transaction Count	133	133
VAFactor	1.01	1.03
Function Point Count	134.33	136.99
Date Updated	2003-05-07	2003-05-09

Question	Initial Value	Final Value
Data communications	1	2
Distributed Data Processing	2	2
Performance	4	4
Heavy used configuration	4	4
Transaction Rate	2	1
Online Data Entry	0	0
End-User Efficiency	5	5
Online Update	0	0
Complex Processing	5	5
Reusability	4	4
Installation Ease	3	4
Operational Ease	1	2
Multiple Sites	2	2
Facilitate Change	3	3

Figure 1 Summary information of project point count

3.1 Language used

Language	Number	Percentage (rounded)
Access	3	4.8
C#	1	1.6
C++	2	3.2
Delphi	1	1.6
Foxpro	1	1.6
GTK	1	1.6
Java	5	7.9
Perl	1	1.6
PHP	17	27
VB.NET/ASP.NET	13	20.6
VB/ASP	17	27
WinRunner	1	1.6
Total	63	100.0

Table 1 Languages used in IE Projects

Language can have an impact on the difficulty of a project, here we have analysed the projects to also determine if the language may have an impact on the size of the project. Table 1 shows the distribution of languages used in the IE projects.

3.2 Whether a database was used

56 of the 63 projects are using a database (88.9%). This data was designed as we feel it is important to know whether having to use a database impacts on the size of the project and therefore its difficulty. It also demonstrates that this counting method can be applied to non database related projects.

3.3 Operating system used

In terms of the operating systems used by the projects to develop the software:

- Unix/Linux 20 projects (31.7%)
- Windows 43 projects (68.3%)

The operating system may affect the size of the project, so this factor was determined.

3.4 Whether the project is internal to Monash University or not

It is difficult to source all of the IE projects from external enterprises, so it is necessary for the School to use projects that have other parts of the University as the client. It is important that such internal client projects are equivalent to external client projects and size is a first point to use as a comparison. 48 of the 63 projects (76.2%) have external clients, while 15 projects have internal clients (23.8%).

3.5 Whether a web element is to be developed

Another aspect that could impact the size of a project is whether the project has to develop web pages. A sizeable majority of the projects have to develop web pages (50 out of 63 projects – 79.4%). In these projects the majority of the content for the web pages comes from a database and so is dynamically derived; even data like the 'contact us' details are stored in the database.

4 Results

50 of the 63 IE projects have submitted their initial function point counts. There are several reasons for the groups not having submitted their initial function point counts. An example is the two groups that are using the Extreme Programming (XP) software development methodology to develop the software. In these two particular cases, the possible story cards were not enumerated sufficiently by the client at the outset to generate a project point count. These projects should be able to generate a final project point count.

The two projects that submitted the largest initial project point counts were more than three times larger than the third largest project. Two projects reported much smaller initial project point counts than any other project. It was decided to remove these outliers from the statistical analysis as they seemed to indicate a misunderstanding on the part of the groups as to how to calculate function points. Table 2 shows a number of statistics for the project point count and the components that make up the project point count. As can be seen the variation is large for all elements.

	Data Count	**Transactions**	**VAF**	**Final PP Count**
Mean	19.4	243.3	1.00	261.5
Lower*	11.8	208.3	0.96	222.7
Upper*	26.9	278.4	1.03	300.3
Std Dev	25.5	118.0	0.12	130.7
Median	7	224	1.0	230.4
Min	0	49	0.79	81.8
Max	84	490	1.24	534.2

Table 2 Descriptive Statistics for Results

* Upper and Lower Bound for 95% Confidence Interval for Mean

An independent sample t-test was conducted and found that there were no statistically significant differences in the final project point count ($p>0.05$) for the use of databases in the project, whether Windows or Linux was used or whether a project was internal to the University or external to the University.

A statistically significant difference was found between projects that had to develop web pages and those that did not ($p=0.022$). Projects that had to develop web pages were significantly larger (mean = 284.4 project points as compared to 179.1 project points) than those that did not have to develop web pages.

5 Discussion

The results to date show that this method of calculating project size from data and transaction information is independent of operating system or development language used. This is consistent with Albrecht's original model (Albrecht 1979).

The heuristic used by the co-ordinators in scoping projects is 12 to 15 tables and 10 to 15 reports. This would give projects a size of between 402 and 525 points. The average point count shown in table 2 is 261.5 and is lower than expected.

At the time of writing, the second automatic project point count had not been completed by the students but it is expected that all groups should be able to complete the process. It is expected that the average project point count will increase for two reasons. Firstly, the sample size will increase and secondly, errors in the design should have been corrected. The example in appendix 1 illustrates this point. The example shows a web site for a primary school; most of the web page contents come from the database. The use case diagram shows many transactions as ‘viewing’ web pages; this is acceptable, as the data has to be retrieved from the database. However, there are only two use cases dealing with database changes; there should be up to 27 uses cases given that there are 9 tables and each table could have an add, edit and delete function. This is a significant underestimation of project size.

It was expected by the project co-ordinators that there would be a variation in project size. However, the standard deviation of 130 points shown in table two is very large and confirms that the project size be used as a moderator in grading the projects. A reason why Web based projects tend to be bigger is the requirement that the students provide the client with an interface to maintain all web page content so adding to the number of tables that are required by the system.

Another point highlighted by the statistics in table 2 is that the VAF averages out at 1. This means that it has no effect on the final point count and confirms a note by Morris in Hastings that “..the VAF provides little or no value.” (Hastings 2000) page 64. The project co-ordinators may dispense with computing the VAF in next year’s iteration of projects and so simplify the counting process even further.

A number of areas are to be explored in future work. These include: (i) dealing with data stored in XML files instead of database tables. The corresponding DTD (Data Type Definition) of the XML file could be parsed to determine the number of elements and so calculate a data count. (ii) The project point count be compared to the source lines of code of the final system as discussed by Jones (Jones 1995). Issues such as code produced by ‘wizards’ would need to be resolved. (iii) A man-hour per project point could be determined. The students are required to keep a time log in WIER for time spent on the various phases of the project. This could validate a heuristic used by the project co-ordinators that the project takes 1200 to 1500 man-hours. (iv) The issue of a project complexity measure is still to be addressed. One of the most complex projects for 2003 student group is the web interface for a robot controller. This recorded a project size of 166 – well under the average.

The authors consider the proposed method for counting project points to be simple enough to eliminate ‘human counter bias’; hence its automation. Inaccuracies with the project point counts will probably come from analysis and design errors made by the students as illustrated by the number of use cases depicted in the example in appendix 1. The project co-ordinators have data on 114 past projects from 2001and 2002 and plan to produce data as in tables 1 and 2 for further verification of the metric.

6 References

Abiad, S., R. A. Haraty, et al. (2000). Software metrics for small database applications. Proceedings of the 2000 ACM symposium on Applied computing, Como, Italy, ACM Press New York, NY, USA.

Agarwal, R., M. Kumar, et al. (2001). "Estimating Software Projects." ACM SIGSOFT Software Engineering Notes **Vol 26**(Issue 4): 60 - 67.

Albrecht, A. J. (1979). Measuring Application Development Productivity. Proceeding of the IBM Applications Development Symposium, GUIDE/SHARE., Monterey, California.

Berztiss, A. T. (1997). "Failproof team Projects in Software Engineering Courses". 27th Annual Frontiers in Education Conference.

Ceddia, J., Tucker, S., Clemence,C., Cambrell,A. (2001). "WIER-Implementing Artifact Reuse in an Educational Environment with Real Projects." 31st Annual Frontiers in Education Conference.

Chamillard, A. T. and K. A. Braun (2002). The software engineering capstone: structure and tradeoffs. Proceedings of the 33rd SIGCSE technical symposium on Computer science education, Cincinnati, Kentucky, ACM Press New York, NY, USA.

Chan, S. C. F., Ng, V.T.Y. and Wu, A.K.W. (2001). Cooperative/Collaberative Learning - Web based Management of Group Projects. International Conference on Computers in Education, Korea.

Collofello, J. S., and Hart, M. (1999). Monitoring Team Progress in a Software Engineering Project Class". In 29th Annual Frontiers in Education Conference.

Daniels, M., and Asplund, L. (1999). "Full Scale Industrial Work in a one semester course". In 29th Annual Frontiers in Education Conference.

Fenton, N. E. and S.L.Pfleeger (1997). Software Metrics: A Rigorous and Practical Approach. London, PWS.

Fincher, S., and Petre, M (1998). "Project-based learning practices in computer science education." In Frontiers in Education Conference.

GERSHWIN (1998). GERSHWIN Database design tool. Freeware product available to students at Monash University that was development by staff at Monash University.

Hagan, D. L., Tucker, S., and Ceddia, J. (1999). "Industrial Experience Projects: A Balance of Process and Product." Computer Science Education **9**(3): 215-229.

Hastings, T. and A. S. M. Sajeev (2001). "A Vector-Based Approach to Software Size Measurement and Effort Estimation." IEEE Transactions on Software Engineering **27**(4): 337-350.

Hastings, T. M. (2000). Measuring software size and predicting development effort of contempory software systems. School of computer science and software engineering. Melbourne, Australia, Monash University**:** 273.

IFPUG (2000). Function Point Counting Practices Manual, Release 4.1.

Jones, C. (1995). Programming languages table. Cambridge, Massachusetts, Software Productivity Research Inc.

K.H.Moller and D.J.Paulish (1993). Software Metrics: A Practitioner's Guide to Improved Product Development., Chapman & Hall.

Kemerer, C. F. (1993). Reliability of function points measurement: a field experiment. Communications of the ACM Volume 36 , Issue 2 (February pp.85 - 97), ACM Press New York, NY, USA.

Kusumoto, S., M. Imagawa, et al. (2002). Function point measurement from Java programs. Proceedings of the 24th international conference on Software Engineering., Orlando, Florida.

Moller, K. H. and D. J. Paulish (1993). Software Metrics: A Practitioner's Guide to Improved Product Development., Chapman & Hall.

RationalSoftware (2002). Rational Software http://www.rational.com/products/rose/prodinfo.jsp.

Utting, I. (1999). Negotiated Assesment Criteria and peer assessment in software engineering group project work: A case study. What have they learned? Assessment of Student learning in Higher Education. European Society for Engineering Education, SEFI, Brussels.

7 Appendix 1. Example:

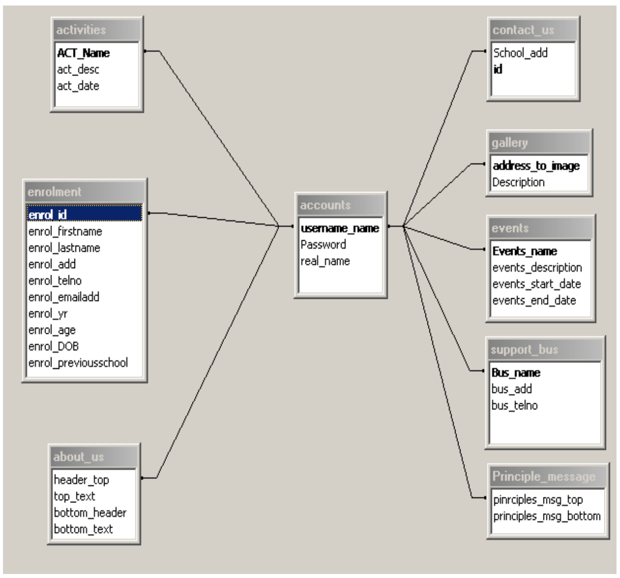


Figure 2 ER diagram showing 9 tables.

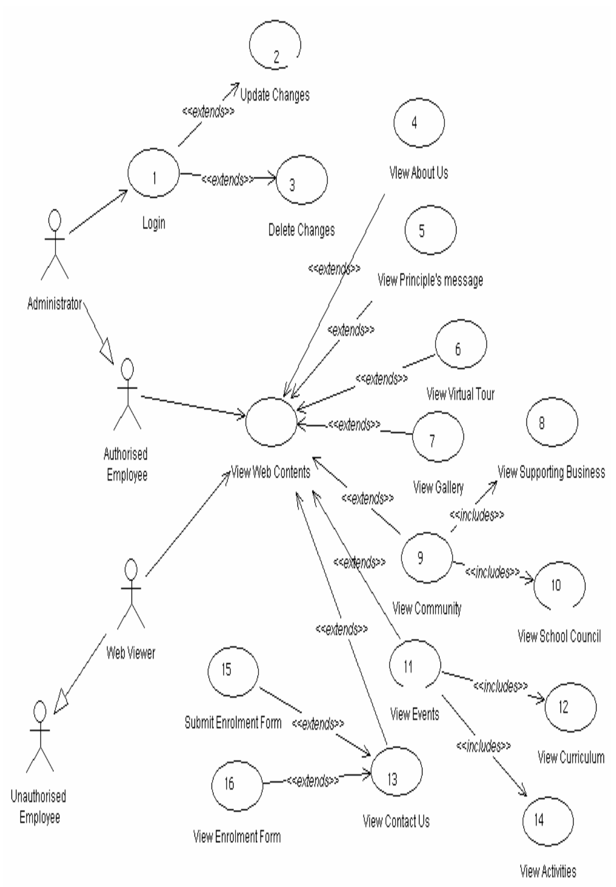


Figure 3 Case diagram showing 16 transactions.

General System Characteristics(GSC's)	**Degree of Influence(DI) 0(least)-5 (most)**
1 Data communications eg web connection	1
2 Distributed processing eg client/server	0
3 Performance eg min. response time	0
4 Heavily used configuration eg setup likely to change often such as dynamic web content	3
5 Transaction Rates	0
6 On line data entry	3
7 Design for end user efficiency	5
8 Online updates	4
9 Complex processing eg calns/ lookups	0
10. Usable in other applications	0
11 Installation ease	2
12 Operational Ease	4
13 Multiple sites	0
14 Facilitate change	3
Total Degrees of Influence (TDI)	25
Value Adjust Factor (VAF) = (TDI*0.01)+0.65	(25 * 0.01)+0.65 = 0.9
Data Count (DC) 9 tables	9 * 7 = 63
Transaction Count (TC) 16 transactions	16 * 7 = 112
Project Point Count = (DC + TC) * VAF	(63 + 112) * 0.9 = **157.5**

Table 3 Sample: project point count is 158 (rounding 157.5)

The sample count shown here is based on a project that provides a web site to a school. There is a database to hold some of the web content. There are two user types :- administrator and everybody else (world). The administrator is able to log in and change some of the database content, which is then reflected in the web pages. Everyone else has view only access.

This example also illustrates how projects can be 'underrated'. Use cases 2 and 3 refer to changing the site content that is stored in the database. For example the school principal (the system administrator) is able to add/edit school events, add/edit supporting businesses' details etc. These functions are barely shown while the view function is expanded to itemize each page in the site. There should be up to another 27 use cases – add, edit and delete for each of the 9 tables shown.