Application Portfolio Management, the basics
How much software do I have

Marcel Rispens, Rabobank
Frank Vogelezang, Sogeti

Abstract

After two external benchmarks, the Software Application Support division decided that the only way to quantitatively manage application support was by establishing a sufficient estimate of the size of the application portfolio. Based on criteria from Gartner's Application Benchmark a selection was made regarding which applications made up the application portfolio. Five different methods were used to size approximately three hundred applications, each with its own precision and cost efficiency: Gartner Fast FPA estimation, Backfiring from LoC for some older PL/1 and Assembler applications, detailed counts with NESMA FPA and COSMIC for newly developed systems and backtracking from budget for less important smaller applications. The size estimations were made by regular support personnel with little training in functional size measurement. A sample selection was reviewed by an experienced consultant, in order to detect possible pitfalls and ambiguities. The results from the review led to a re-evaluation of most of the FPA-estimations, with a higher precision and greater consistency. The results of these size estimates can now be used to compare parts of the portfolio and to quantitatively manage the (support of the) portfolio.

1. Introduction

Rabobank is a cooperation of over two hundred local banks that jointly form one of the largest retail banks in the Netherlands. Rabobank Nederland supports the local banks with central procurement services, back-office processing, infrastructure services and IT. The IT and infrastructure services are provided by Groep ICT. Within Groep ICT, the B&E Application Support department is responsible for supporting about three hundred applications. The department's mission statement is to deliver application services to Rabobank's local banks and central departments in compliance with agreed service levels and by achieving Operational Excellence.

Early in 2005, four separate application support departments merged to form the current Application Support department. It was a merger of four very different types of departments into a single organization. One of the departments was based in the formerly centralized IT division, while others supported applications in very close cooperation with business departments. The four departments had very different cultures, some of which are still visible today.

2. Benchmarking performance

To get a clear image of the current performance of the different parts of the newly formed Application Support department we began benchmarking its performance. In order to benchmark the performance and success of the merger every year, an external organization benchmarks the Application Support department against a representative peer group of other financial companies.

2.1. Benchmark 2005

The first benchmark by Maturity was performed shortly after the merger in the spring of 2005. We needed an indication on how we performed in Application Support regarding cost, productivity and quality when compared to others. For this benchmark it was necessary to have an idea of the size of the software. Size is the primary input for many models that estimate and control cost and effort [1]. Although size has multiple dimensions, only functional size has been used for benchmarking purposes, because it is the dimension of size that is best understood and for which there are clearly defined standards. The assumption is that, on average, functional size is also a good representation of non-functional size.

This was the moment that the Application Support department was introduced to functional size measurement. Functional size measurement has already been used to estimate projects within the Rabobank software development department [2]. For the Application Support department this was a new phenomenon. There was no overall picture of the size of the application portfolio of Rabobank.

In order to perform the benchmark, it was necessary to be able to compare the applications. Consequently, applications were sized in various ways. Some applications were sized in Lines of Code, some were estimated by our own department, using a proprietary method from Maturity, and for some applications an actual function point count was made. The choice of the method was based on the availability of detailed documentation of the application's functionality.

The results of this benchmark gave the Application Support department a critical, alarming picture of the performance. Targets for cost reduction and process improvement were defined. The three major benchmark indicators (cost per function point, function points supported per full-time equivalent and defects per 1,000 function points) were very useful, but at that time we did not implement benchmarking as management information. The Application Support department considered the management information at that moment as sufficient and decided that regular external benchmarks would be sufficient to realize the cost reduction and process improvement targets.

2.2. Benchmark 2006

In the spring of 2006 the Application Support department initiated a new benchmark by Gartner. Groep ICT had started to use Gartner's Total Cost of Ownership (TCO) model [3]. A benchmark by Gartner itself would give the best fit with that model because of unambiguous definitions and combined benchmark results from different departments.

The Application Support department had learned from the first benchmark and wanted to improve the quality of the delivered data. For this benchmark we estimated functions points using the Gartner Fast FPA estimation method only. With this method, the functional size of an application can be estimated from characteristics that can be derived from the application (also see section 3.1.1). The results of this benchmark were better: we scored better on quality and the financial and productivity indicators were slightly better.

Gartner advised us to take the next step with regard to Performance management and Application Portfolio Management: implementing a metrics program with key indicators based on function points, cost and productivity for the application portfolio as a whole. This data would be used as a start for Application Portfolio Management.

As a result, we began the metrics program. The Application Support department supports approximately three hundred applications, ranging from very large to very small ones. At that point in time we had function point estimations for 120 of the most prominent applications, adding up to approximately 400,000 function points. The size of the rest of the portfolio was estimated by backtracking from budget (see section 3.1.5). We estimated the size of our portfolio at 530,000 function points.

The first results from our metrics program were rather curious: we had extremely cheap or expensive applications, with extremely low or high productivity. Most likely this was the result of inaccurate function point estimates. Functional size estimation is not a core competence of our department. As part of the benchmark, a number of our employees received function point instructions and were taught how to use the MS Excel tool provided by Gartner.

However, this was not a sufficient basis to build performance management on. The quality of function point estimates had to be validated.

2.3. Benchmark 2007

This is an interesting thought in the case of the next benchmark. We may assume that peer group data will be of more or less the same quality as ours. This makes it possible to perform a comparative analysis. But what if we improve data quality on function points by validating the size estimates and providing our application support staff with better tools? As we will show, this can have serious consequences for the size of the application portfolio and consequently for comparability within our peer group.

This spring we will once again perform an external benchmark. The benchmarks in 2005 and 2006 were time-consuming projects in which support teams had to estimate some of their applications on short notice. With our metrics program we now come close to delivering the data from our internal benchmarking process to Gartner on the fly, for an annual update of the external benchmark.

3. Improving the base metric: functional size

Both the 2005 and the 2006 benchmarks concluded that there was not enough knowledge about and experience with functional size measurement within the Application Support department to make size estimates of sufficient quality. As a result, the differences in the quality of the size estimates produced by the department were quite large. Earlier research shows that having novice estimators can lead to differences in functional size measurements of over 20% [4].

3.1. Current functional size measurements

After the 2006 benchmark, the functional size of all applications supported by the Application Support department was determined. Depending on the age, size and importance of the applications, one of the following functional size measurement methods was used:

3.1.1. Fast FPA estimation

For the largest part of the portfolio the functional size had been estimated with a fast FPA estimation technique. This technique counts a number of characteristics that can be distinguished in an installed application and translates this data into an estimate of the number of function points (IFPUG/NESMA) that would have been produced by a detailed analysis of functionality. The advantage of a Fast FPA estimation technique is that it can produce a size estimate in cases where available documentation is insufficient to make a detailed analysis or when there is insufficient time to make a detailed analysis [5].

There are several of these fast FPA estimation techniques. The one we used was provided by Gartner and used the following categories:

A. Logical structures, divided into two categories:
   - *maintained by the application* and
   - *used by the application (read-only).*
B. Logical inputs, divided into five categories:
   - *logical forms that maintain unique business data*,
   - *inputs from other applications*,
   - *unique controls that provide a list of items to select*,
   - *logical screens that browse, inquire or display existing data* and
   - *background processes initiated by the user*.
C. Logical outputs, divided into two categories:
   - *canned reports* and
   - *outputs to other applications.*

The values for the nine categories were determined by the Application Support staff and translated into function points by an MS Excel tool provided by Gartner.

107 applications with a total size of 389,838 fp were sized using this technique.

3.1.2. Backfiring

Several older applications, most of them written in PL/1 and some in Assembler or Cobol, were sized by using the SPR backfiring technique, in which the number of logical source statements is correlated with the number of function points [6]. Backfiring is a high-order estimation technique that is easily distorted by local development practices and the way functionality is delivered. In addition to this, there are over a dozen ways to count logical source statements. Consequently, it was only used to size some of the older applications containing very little user interfaces and with no available documentation. Comparability was assured by using the same source statement counting tool.

16 applications with a total size of 3,472 fp were sized using this technique.

3.1.3. Detailed FPA

Detailed function point counts were available for a few custom-built applications from external suppliers. Those counts were made in compliance with the NESMA standard, which produces nearly the same results as the current IFPUG standard [7].

3 applications with a total size of 8,460 fp were sized using this technique.

3.1.4. COSMIC

Applications that were developed within Rabobank recently, were sized with the COSMIC method [2]. Although the COSMIC method is based on different principles than both the IFPUG and NESMA FPA standards, size measurements from both methods can be transformed into one another using a simple algorithm [8]. The size derived with the COSMIC method was transformed to function points with this transformation algorithm.

1 application with a size of 2,554 cfp was sized using this technique.

3.1.5. Backtracking from budget

A large number of smaller applications were not sized with one of the four methods above, but were assigned a fictitious size. Using the prominent applications that had been sized, we could calculate the average cost per function point. With this average cost per function point we calculated the fictitious size by dividing the budget by this average cost per function point.

This was done because most of those applications are not monitored as a single application but in clusters of similar applications, and because most of these applications are relatively small (fewer than two hundred function points). A more detailed sizing effort would not be cost effective for these small applications.

179 applications with a total size of 101,454 fp were sized using this technique.

3.2. Reviews on functional size estimates

Most of the review effort went to the size estimates that were made with the Fast FPA estimation technique. Not only because of the large volume of estimates, but also because these size estimates were assumed to be most prone to error, as most of the size estimates were carried out by support professionals with a limited experience in functional size measurement. The detailed FPA and COSMIC size measurements were made by qualified functional size measurement professionals and were based on documentation [2]. The only corrections for these size measurements were due to scope creep since the last measurement. The accuracy of the backfiring method is largely determined by the quality of the tool with which the number of logical source statements is counted. A review will not improve that accuracy, because the inaccuracy is a result of the method itself and not of the use of the method.

3.2.1. Fast FPA estimation

The category of logical structures proved to be the most difficult for the Application Support staff, especially when logical documentation was incomplete or unavailable. In the 1998 British ICCE experiment – based on documented functionality – differences of up to 50% were observed for novice estimators rating logical files [9]. Estimates for this category could be rapidly improved with some environment-specific training. We did not perform a complete inter-counter consistency check, but decided that consistency should be enhanced by having the estimates reviewed by a single experienced consultant.

In the logical input category, the logical forms sometimes led to misinterpretation for applications that used one screen for each type of operation (e.g. create, update, delete). The Gartner Fast FPA estimation method presumes these operations to reside in a single physical screen. Another difficult category in the logical inputs was unique controls, because of the fact that they can have a number of different appearances and that one unique control may be used several times within the application. It can be hard to filter out the unique occurrences of these controls without documentation.

Corrections of the initial size estimations ranged from -60% to +130% with an average correction of about -30%. This meant that ***on average*** the applications that were sized with the Gartner Fast FPA estimation method were overrated by more than 40%. The results of the reviews on the functional size estimates are summarized in the table on the following page.

The main causes of these inflated ratings were a lack of experience with the method and the inability to apply the training – that had focused on desktop applications – to the applications to be sized, which were mostly mainframe applications or packaged software.

Table 1: Review results for Fast FPA estimation

Application	Initial size	After review	Correction
#01 Package as-is	8,102	4,100	-49%
#02 Package as-is	11,651	4,600	-61%
#03 Package as-is	336	410	+22%
#04 Package interface	142	330	+132%
#05 Custom built	3,578	3,400	-5%
#06 Package as-is	28,945	20,000	-31%
#07 Package customized	14,680	14,680	0%
#08 Custom built	3,302	2,412	-27%
#09 Custom built	11,786	8,000	-32%
#10 Package integrated	14,525	13,000	-10%
#11 Custom built	11,400	6,500	-43%
#12 Package integrated	10,325	6,500	-37%
#13 Package customized	35,650	15,200	-57%
#14 Custom built	2,270	2,554	+13%
#15 Custom built	3,578	3,400	-5%
#16 Custom built	1,290	1,650	+28%
#17 Custom built	2,389	3,400	+42%
#18 Custom built	2,234	1,800	-19%
#19 Custom built	1,539	830	-46%
#20 Custom built	12,726	12,500	-2%
#21 Custom built	1,222	1,500	+23%
#22 Custom built	2,412	2,130	-12%
TOTAL	184,082	128,896	-30%

3.3. New support tool

Keeping these main causes in mind, a new FPA estimation tool was developed based on the FPA method. In case of doubt the estimator could fall back on the general definitions of the FPA method. This new support tool has the following categories:

1. Data, divided into two categories:
   - *internal logical files* and
   - *external interface files.*
2. External inputs, divided into three categories:
   - *maintenance functions*,
   - *inbound interfaces*,
   - *background processes initiated by the user*.
3. External outputs, divided into four categories:
   - *inquiries*,
   - *reports*,
   - *outbound interfaces* and
   - *unique controls to select field values.*
4. External inquiries, divided into two categories:
   - *inquiries with a unique identifying key* and
   - *help functionality*.

Also we enriched the tool with help topics dealing with all learning points both from this validation and from the Q&A sessions of the 2006 benchmark. Finally, we included several rules of thumb as a first sanity check for the estimator. These rules of thumb signal when parts of the size estimate fall outside of the expected range.

The rules of thumb that were incorporated are:

- *expected size, based on the number of logical files,*
- *fraction of the size from logical files – should not exceed 40%*,
- *expected number of logical files, based on the number of maintenance functions,*
- *expected number of maintenance functions, based on the number of logical files* and
- *the expected number of select controls*.

Using the new tool and a targeted sizing instruction, we expect to improve the accuracy of functional size estimates for future benchmarks. A sample report from the tool is provided in the appendix.

3.4. Ready for 2007

With the reviewed size estimates and an improved estimation method, the base metric was accepted as a sufficient basis for building performance management upon. The first internal indicators have been put in place to provide management information on a quarterly basis. Benchmarking has now changed from being an annual hassle to produce figures to 'business as usual'. The internal indicators are shown in section 5.

4. Special sizing challenges: packaged software

Within the Rabobank organization, a large part of the software portfolio consists of packaged software, which poses some interesting sizing challenges for Application Support. Literature distinguishes six different categories within the packaged software domain [10]:

- *as-is, implemented without change*,
- *customized for this installation, within the package*,
- *as integrated part of an existing application in the organization*,
- *new functionality in addition to existing package functionality*,
- *new functionality to enable interfacing with other applications* and
- *unused functionality of the package*.

4.1. Packaged software used as-is

For packaged software that can be used out-of-the-box, without any customizing of the functionality, an estimate of the functional size is irrelevant for Application Support purposes. The cost of support for this type of packaged software is determined by factors such as license cost and number of users rather than the functional size of the software.

4.2. Packaged software, customized within the package

This category of packaged software can be divided into packages in which functionality can be switched on or off based on the company's needs, and software in which the behavior of the provided functionality can be parameterized.

Examples of the first category are packages that support ITIL-processes configured to support only those processes for which software support is implemented. Other examples of the first category, where it provides only selected special functionality, can be found in section 4.3. For this category it is important to size only the part of the packaged software that is in use, because that is the size that is relevant for the Application Support effort. Files and screens that are not in use may not be included in the size estimate.

ERP and CRM packages fall in the second category. Essentially the same rule applies as for the first category, but in this packaged software it is harder to establish which part of the functionality offered is actually being used. In the following sections we will present the results for one of our Siebel implementations and for three SAP applications.

4.2.1. Siebel

The Siebel CRM package has more than four thousand tables. A large number of them would not be counted as a logical file. To determine the number of actual logical files for the Rabobank implementation we started from the Responsibilities table. This table describes which user role has the right to use which implemented functionality. Only the business entities to which at least one user role was assigned were counted as logical files. We determined that only 160 business entities were used in this implementation.

We determined the number of applets that were used in this implementation using the same approach. We determined that 540 of the more than one thousand available applets were used. Siebel has two categories of applets: list applets and form applets. List applets represent one external output. Form applets represent three external inputs (create, update, delete) and one external output (read).

With this approach we estimated the functional size of the Siebel CRM implementation at Rabobank at 15,200 function points.

4.2.2. SAP

For SAP the functionality is harder to estimate. When detailed requirements are available a detailed count can be performed. For Software Engineering purposes this only has a predictive value on a large scale because there is no linear relation between the amount of effort needed to parameterize the package and the amount of functionality that is delivered. Some rules of thumb are available for large-scale installation of SAP functionality [11].

For Application Support purposes only the functionality available is relevant and a detailed count is not cost-effective, so we were looking for an estimation approach such as the one described for Siebel. It is not possible within SAP to easily determine which part of the more than forty thousand tables in the software package is used and which part is not used within an implementation.

Maya Daneva from the Twente University in the Netherlands has proposed a sizing method based on high-level SAP requirements [12]. Some important components of the proposed methods have no relevance to the Rabobank requirements structure, so the method could not be applied within the Rabobank environment.

From practice we had two rules of thumb that we could test with the data for the Rabobank implementations:

- *logical files make up about 25% of the total functional size* and
- *reports make up about 25% of the total functional size of a SAP implementation*.

The data that was available for the implementations was the number of active transactions and the number of active reports. From the more than 70,000 available transactions we determined that some 5,000 transactions and some 4,000 reports were active in one of the implementations. Transactions within SAP are generally equivalent to an external input (or output in case of a report transaction). Based on the number of transactions, we expected between 850 and 3,400 logical files, being 50-200% of the number of external inputs divided by three (create, delete and update of the same logical file).

If we assume the rules of thumb to be more or less correct, the number of logical files must be about the average of the expected lower and upper boundary. With this additional rule of thumb we estimated the functional size of the SAP financial administration implementation at Rabobank at close to 69,000 function points. A much smaller implementation for the human resource department was estimated at just over 9,000 function points.

The three rules of thumb could not be aligned for a business warehouse implementation of SAP. This indicates that these rules of thumb can only be applied if the SAP implementation covers a broad range of the available functionality.

4.3. Packaged software as integrated part of an existing application

For this kind of software, more or less the same rules apply as for the first category of software customized within the package. It is important to size only the part of the packaged software that is in use.

From an end-to-end perspective on functionality this means that end-to-end functionality from a user perspective can be delivered by two pieces of software. When the interaction between both parts is known they can be counted as one piece of software. For our purposes we had to estimate the size of both parts separately. Strictly speaking, this means that functionality that uses both parts of the software for end-to-end functionality will have an inflated rating. Because of the fact that these estimates were used for Application Support purposes – for which purposes two applications have to be maintained – we did not correct for this inflation.

4.4. New functionality

New functionality, whether it is implemented to realize additional functionality or for interfacing purposes, is usually considered to be a separate application within the Rabobank portfolio. This means that the regular rules for functional size measurement apply to this kind of functionality.

4.5. Unused functionality

At this moment we ignore the size of unused functionality of packaged software. Once Application Portfolio Management reaches higher levels of maturity, the fraction of unused functionality could become a more important factor for architectural decisions.

5. The first management indicators

The productivity indicator of the number of function points supported has been improved to over 1,500 function points per FTE, which is close to the benchmark productivity in our peer group. Our cost per function points has slightly decreased, but is still higher than the benchmark cost within our peer group.

We used the benchmark results to focus our improvement effort and we now use our own implemented indicators. In February of this year we completed our first internal benchmark. In a joint operation between Finance & Control and Quality management we presented new results concerning the internal benchmark indicators according to Gartner definitions to the Application Support department management. This internal benchmark will be repeated every quarter from now on.

5.1. Cost per function point

This indicator is based on the cost of hours spent on application support, including third-line support, hardware and software for testing, supporting activities and overhead, per function point. We are currently making a detailed analysis of the composition of the benchmark cost per function point in relation to our actual cost per function point. A good insight in this composition will limit the risk of drawing wrong conclusions. If the organizational overhead part of our cost per function point is substantially higher than the benchmark cost per function point of our peer group we need to take action on that issue, instead of modifying the number or quality of application support staff.

Focusing on the function point part, with our effort in acquiring more reliable –in our case substantially lower – functional size measurements, we are most likely doing better than our peer group, as we have mentioned before. Apart from this consideration, the cost per function point gives us more insight in our performance in financial terms.

5.2. Function points serviced per full-time equivalent

This indicator is based on the hours spent by application support, including third-line support and overhead hours of our own Application Support department. It is a productivity indicator that indicates how many application we can support per FTE. It is used to compare individual applications. There has to be a good explanation why some (groups of) applications require more support hours than average applications. We show the productivity results at (sub-) department level in our management indicators.

5.3. Defects per 1,000 function points

This quality indicator was more complex to implement than others. Rabobank ICT has organized IT support with an ITIL implementation, and consequently thinks in terms of “incidents” and “problems”. We had some difficulty to report in terms of defects. According to Gartner, defects are software errors. We are now able to abstract the right data from our existing ITIL administration. This had its impact on the benchmark results (in the first year the number was too high, and in the second year it was too low). When reported correctly, it is a very useful indicator for evaluating software quality.

6. Application Portfolio Management

Managing an application portfolio using a single indicator would be comparable to measuring the various functions within a city by measuring the size they occupy on a city map. Good Application Portfolio Management means managing the portfolio using a set of indicators that can be put into perspective. Our goal with Application Portfolio Management is to coordinate investments in new and existing applications, and this cannot be realized by Application Support alone.

The Application Support department defined six indicators that are needed to successfully support Application Portfolio Management. Application Support is collecting and analyzing data for each application to establish a “more-dimensional” view on an application or group of applications. A database was installed for this purpose. As a result of the first prototype results, the Service Management and Programme and Information Management departments were invited to join the initiative. We are currently trying to make a connection between Application Portfolio Management and Project Portfolio Management.

6.1. APM parameter - Size

Since 100% accuracy is not necessary, the size of most of our applications is determined with the Fast estimating method. As explained earlier, we need a reasonable estimation of the size for Application Portfolio Management.

6.2. APM parameter - Cost per function point

The way we determine cost per function point has already been described in section 5.1. This indicator is primarily used to put the other indicators in perspective.

6.3. APM parameter - Service level

Application Support is making its support more uniform along three support levels: Gold, Silver and Bronze. The Golden service level entails 24 × 7 application support and extensive continuity measures. The Silver service level entails application support during extended office hours and a certain level of continuity measures. The Bronze service level entails application support from 9 to 5 and simple back-up procedures.

6.4. APM parameter - Architectural fit

The amount of support effort needed reflects whether functionality, application and infrastructure comply with the architectural guidelines. In order to determine the APM parameter for architectural fit, we used a simple scoring table to review applications on architectural guideline compliance.

6.5. APM parameter - Complexity

Function points are not the only way to “size” applications. Function points only reflect the amount of functionality applications offer. Once again using a simple scoring table, we can score applications on infrastructural complexity, application complexity, organizational complexity and functional complexity. The scoring process is supported by a small number of guidelines.

6.6. APM parameter - Business importance

How important is an application for the business of the Rabobank organization. In order to establish this parameter, we use a scoring table from the Gartner benchmark. Application Support is currently ranking the business importance. In the future we will ask the Business departments to determine the business importance of applications themselves.

7. Results

We now have established an Application Portfolio Management process which enables us to make well-informed decisions about our application portfolio. We will further improve our parameters if this process proves to be successful.

We have made the functional sizing process reliable enough to function as base metric, with an accuracy that is enough for Application Portfolio Management purposes, but which is also cost efficient. An estimation method is sufficiently accurate to manage a whole portfolio of applications.

With an adequate estimation technique and review mechanism, functional size estimates can be performed by the same professionals that also support the applications. Size estimation can thus be implemented as one of the application support processes.

We see functional size measurement as the only practical metric for comparing applications. Functional size measurement alone is not sufficient to accurately predict support effort. To realistically predict support effort, one also has to take into account a complexity metric to make Application Portfolio Management really effective.

We believe that metrics should always support decisions in Application Portfolio Management and not replace the decision process.

It will take some time to determine whether our Application Portfolio Management initiative actually has an impact on major application life cycle decisions. We believe that the best way to make Application Portfolio Management work is to start with a simple set of reliable metrics that fit an existing best practice model (in our case Gartner's TCO model) and only expand it if expansion has a real added value for the decision makers.

References

[1] Chrissis, M.B., Konrad, M., Shrum, S., CMMI® Guidelines for process integration and product improvement, Pearson Education, Boston, 2003 *www.sei.cmu.edu/cmmi/publications/cmmi-book.html*

[2] Vogelezang, F.W. and Lesterhuis, A., Applicability of COSMIC Full Function Points in an administrative environment: Experiences of an early adopter, Proceedings of the 13th International Workshop on Software Measurement (IWSM 2003), September 23-25, Montréal (Canada) *www.lrgl.uqam.ca/workshops/iwsm2003*

[3] Harris, K. and McRory L., Use a health check to determine your application's 'Fitness for duty', Gartner Research ID G00137171, January 2006 *www.gartner.com*

[4] Rule, P.G., The importance of the size of software requirements, Proceedings of the 12th annual conference of the National Association of Software and Services Companies (NASSCOM 2001), february 7-10, Mumbai (India) *www.measuresw.com/library/Papers/Rule/the_import_of_size/v1c.html*

[5] Vogelezang, F.W. and Prins, T.G., Approximate size measurement with the COSMIC method – Factors of influence, Proceedings of the 4th Software Measurement European Forum (SMEF 2007), may 9-11, Roma (Italy) *www.iir-italy.it/smef2007*

[6] Jones, C., The SPR Programming Languages Table, version PLT2006b, February 2006 *www.spr.com/products/programming.shtm*

[7] NESMA, FPA according to NESMA and IFPUG, version 8, June 2004 *www.nesma.nl*

[8] Heeringen, H. van, Changing from FPA to COSMIC-FFP – A transition framework, Proceedings of the 4th Software Measurement European Forum (SMEF 2007), may 9-11, Roma (Italy) *www.iir-italy.it/smef2007*

[9] UKSMA, The Inter-counter Consistency Checking Experiment, 1998 *www.uksma.co.uk*

[10] Morris, P., Function points as part of a metrics program, in "IT Measurement: Practical advice from the experts", Jones, C. and Linthinicum D.S. (eds), Addison-Wesley, Boston, 2002 *www.ifpug.org*

[11] Jones, C., Software portfolio analysis with function point metrics, version 2, June 18, 1999 *www.spr.com/news/SoftwarePortfolioArticle.pdf*

[12] Daneva, M., Measuring reuse of SAP requirements: a model-based approach, Proceedings of the 5th ACM Symposium on Software Reusability, May 21-23, 1999, Los Angeles (USA) *ssr99.ifi.uni-klu.ac.at/ssr99*

Appendix Estimating tool report sample

FPA size estimation

based on data from the application

application:	the financial administration
size:	69000 function points
estimator:	Frank Vogelezang

			screens	functions	
> Data	> Internal Logical Files	More...	unknown 2125		Logical tables with 50 or less data elements
					Logical tables with more than 50 data elements
	> External Interface Files	More...	unknown		Logical tables with 50 or less data elements
					Logical tables with more than 50 data elements
> Input functions	> Maintenance functions (add, modify, delete)	More...		1320	Screens and/or functions with maximum 4 fields
				3130	Screens and/or functions with 5-15 fields
				650	Screens and/or functions with more than 15 fields
	> Inbound interfaces	More...	unknown 220		Interfaces with maximum 4 fields
					Interfaces with 5-15 fields
					Interfaces with more than 15 fields
	> Batch- or background processing	More...	unknown 1250		Processes with maximum 4 data elements
					Processes with 5-15 data elements
					Processes with more than 15 data elements
> Output functions	> List functions	More...	unknown 630		Screens and/or functions with maximum 5 fields
					Screens and/or functions with 6-20 fields
					Screens and/or functions with more than 20 fields
	> Reports	More...	unknown	1600	Reports with maximum 5 fields
				2600	Reports with 6-20 fields
					Reports with more than 20 fields
	> Outbound interfaces	More...	unknown 220		Interfaces with maximum 5 fields
				1	Interfaces with 6-20 fields
					Interfaces with more than 20 fields
	> Unique select controls to fill field values	More...	1300		Number of unique select controls within the application
> Query functions	> Query functions based on a unique key	More...	unknown 65		Functions with maximum 5 fields
					Functions with 6-20 fields
					Functions with more than 20 fields
	> Help functions	More...			Help function on application level (Y/N)
					Help function(s) on screen level (Y/N)
					Help function(s) on field level / tooltips (Y/N)

> Rules of thumb	> These rules of thumb are meant as assistance:		The rules act as a sanity check of completeness of the estimate.
	> What if this estimate does not comply to the rules of thumb:		This does not necessarily mean that the estimate is wrong. It is sensible to check why for the financial administration this rule of thumb does not apply
	> Size based on logical files:	More...	Based on the number of logical files of the financial administration (2125) and the number of logical files that the financial administration reads (0) a total size is expected for the financial administration between 53125 and 99875 function points. The current estimate is 69000 function points.
	> Contribution of logical files to size:	More...	The contribution of logical tables to the total size is as a rule not more than 40%. For the financial administration that contribution currently is 22%.
	> The number of logical files:	More...	Based on the number of maintenance functions and or screens of the financial administration a number of Internal Logical Files is expected between 850 and 3400. For the financial administration currently 2125 are identified.
	> The number of maintenance screens:	More...	Based on the number of Internal Logical Files of the financial administration a number of maintenance screens is expected between 1063 and 4250. Currently 0 are identified.
	> The number of maintenance functions:	More...	Based on the number of Internal Logical Files of the financial administration a number of maintenance functions is expected between 3188 and 12750. Currently 5100 are identified.
	> The number of select controls:	More...	Based on the number of maintenance screens, maintenance functions and list functions of the financial administration the number of select controls is expected not to exceed 2330. Currently 1300 are identified.