Using COSMIC-FFP for sizing, estimating and planning in an ERP environment

Frank Vogelezang

Sogeti Nederland B.V.

frank.vogelezang@sogeti.nl

Abstract

Triggered by new European legislation the Dutch Office for Regulations decided to renew major parts of their IT landscape with Oracle's E-Business Suite. They expect that this packaged solution offers the possibility of quick implementation of new business processes.

For the implementation of new regulations and the redesign of existing ones, a software factory was set up with three production lines implementing process-chains. Because of the nature of the documentation COSMIC-FFP was used to size the process-chains to be implemented. The measured functional size was used to support the cost estimation and the planning process.

This experience shows that COSMIC-FFP can be used to size, estimate and plan an ERP implementation with a high degree of parameterisation. Since this kind of implementation differs in a number of ways from an average implementation of packaged software future research is necessary.

Keywords

ERP, COSMIC-FFP, packaged applications, sizing, estimating, planning, control

1 Introduction: changing environments

1.1 Political change

Early 2003 the Commission of the European Communities proposed a reform of the common agricultural policy. [1] The main aims of this reform were to cut the link between production and direct payments (decoupling), to make those payments conditional on environmental, food safety, animal welfare, health and occupational safety standards (cross-compliance) and to increase EU support for rural development by a modulation of direct payments (from which small farmers would be exempted).

The proposed reform comes into force next year, which means that this year the information systems for financial support for the agricultural sector have to be revised drastically. In the Netherlands, the Office for Regulations (*Dienst Regelingen*) of the Ministry of Agriculture, Nature and Food Quality is responsible for carrying out all regulations with respect to the agricultural sector.

1.2 Technology change

Given the drastic revision of the financial support regulations, combined with the fact that a number of the current systems were at or near the end of their life-cycle, the Office for Regulations decided to build a new software environment to support the execution of financial regulations. This new environment had to be based on packaged software. In this way, the Office for Regulations expects to be able to implement new or changed business processes quicker than with custom-built software.

The Oracle E-Business Suite (EBS) has been chosen to be the basis for the new software environment. The EBS is a fully integrated, comprehensive suite of business applications, which can be implemented one at a time, as a predefined set for a special kind of business or as the complete suite. [2] The implementation of the EBS is further described in section 7. Next to the EBS basis specialised software and custom-built extensions to EBS are still being used.

1.3 Organisational change

It is expected that the reform of the common agricultural policy will be followed by other changes in regulations for the agricultural sector. Implementing an ERP system was more than an IT project. To achieve maximum flexibility for change the Office for Regulations decided not only to use one generic software environment, but also one generic working process for all regulations under the jurisdiction of the Ministry of Agriculture, Nature and Food Quality. What started as system renewal became a journey of change for the whole organisation with a far-reaching impact. The new working process was described in a Business Process Model. (BPM) The highest level of the BPM is presented in figure 1.

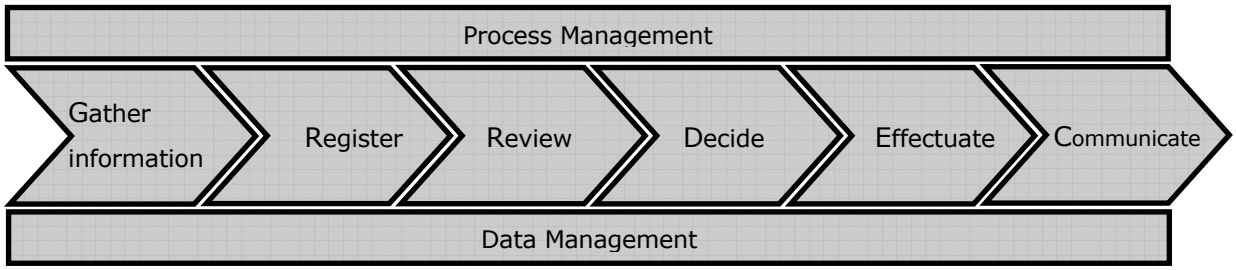

Figure 1: Business Process Model (BPM)

The Oracle E-Business Suite has been preconfigured to support the BPM so that new business processes – based on the BPM – could be implemented easily.

1.4 Design change

The new working process called for a different strategy for the design. Usually the design focusses on the functionality, but in this environment the design should have a very strong focus on the working process rather than the automated functionality. This design strategy also had consequences for the applicability of sizing metrics. (see section 3)

1.5 Production change

Many regulations will have to be implemented in the new environment over time. To do this economically efficient the implementation of new regulations has not been delegated to different projects. For software development projects, it is not unusual that the project teams are dismantled straight after the delivery of the developed software. The knowledge the project team has acquired fans out over the organisation(s) that formed the team. If the software requires a new release or maintenance, all the knowledge has to be re-established and fans out again. In this situation, it was more economic to set up a software factory. [3]

The critical success factors of a software factory are productivity and predictability. All other possible targets of a software factory (like price performance, reducing errors, shortening the time-to-market, reducing cost, less dependency on 'heroes') are derived from the two success factors. A software factory consists of four key elements:

- system engineers (all-round IT knowledge workers);
- standard working method;
- development- and maintenance tools;
- supporting processes and management.

A software factory can be split up into a number of production lines, either to be able to split up the workload or to facilitate different development environments. The Office for Regulations has a single development environment, so its software factory is split up into three production lines to balance the workload. (see section 6.1)

Regulations are implemented as sets of process-chains. A process-chain is a set of business functions that handles a certain event using all the steps of the BPM. By splitting up regulations into sets of process-chains, the workload can be balanced between the production lines.

2 COSMIC-FFP as sizing metric

All new regulations are designed to fit the BPM. One of the first design documents that is delivered is the process model. This document describes all process steps in terms of the BPM and gives an indication whether this is a manual process or a process that will be supported by the E-Business Suite. In this stage, some general information is known about the processes, but little is known about the data interaction. For planning purposes in this stage estimates were required about the expected size of the software support of the new regulation.

This posed a problem for the project management. The software support for a new regulation or a process-chain could not be estimated per EBS component, like a regular ERP-implementation, because this implementation does not implement EBS components, but process-chains based on EBS components. (see section 7) The lack of knowledge about data interaction and how process-chains relate to functions in this stage of the implementation meant it could not be estimated like a traditional custom-built software project by means of EQFPA. [4] But a programme of this magnitude without a way of forecasting the costs and managing the planning schedule was highly undesirable.

The project management decided to use COSMIC-FFP as a sizing technique for the process-chains that had to be implemented.

2.1 About COSMIC-FFP

COSMIC-FFP is a functional size measurement method that has been designed to be equally applicable to business application software, real-time software and infrastructure software. It is the first functional size measurement method that is designed to meet the principles of the ISO/IEC 14143 standard for functional size measurement. [5]

The basic principle of COSMIC-FFP is that the functional user requirements of a piece of software can be broken down into a number of functional processes that are independently executable sets of elementary actions that the software should perform in response to a triggering event. The elementary actions that software can perform are either data movements or data manipulations. COSMIC-FFP assumes that each data movement has an associated constant average amount of data manipulation. With this approximation, the functional processes can be broken down into a number of data movements. A data movement moves a unique set of data attributes (data group) where each included data attribute describes a complementary aspect of the same, single thing or concept (object of interest) about which the software is required to store and/or process data.

COSMIC-FFP distinguishes four different types of data movements:

- **Entry** – An entry is a data movement that moves a data group from a user across the software boundary into the functional process where it is required.
- **Write** – A write is a data movement that moves a data group from inside a functional process to persistent storage.
- **Read** – A read is a data movement that moves a data group from persistent storage to within the functional process, which requires it.
- **Exit** – An exit is a data movement that moves a data group from a functional process across the software boundary to the user that requires it.

The relation between the data movements is graphically represented in figure 2.

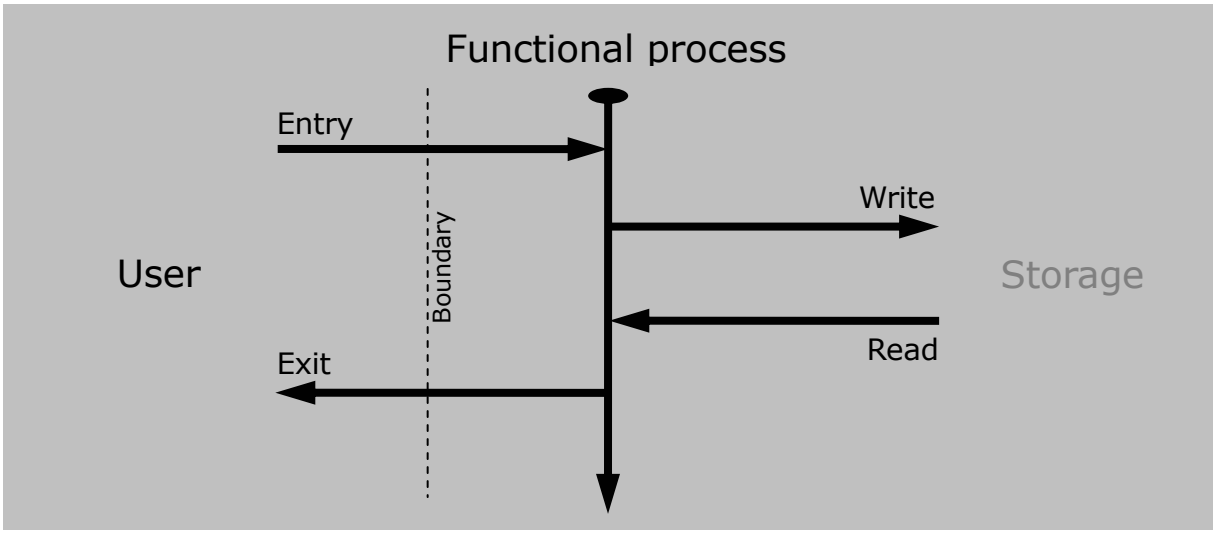

Figure 2: The relation between data movements in COSMIC-FFP

The value of a functional process is determined by the sum of the constituting data movements. The smallest functional process consists of two data movements: an Entry (containing the triggering event) and either a Write or an Exit (containing the action the functional process has to perform). Every identified data movement receives the value of 1 Cfsu (COSMIC functional sizing unit). The size of the smallest functional process is 2 Cfsu and increases with 1 Cfsu per additional data movement to an unlimited number.

2.2 Size measurement in early stages of development with COSMIC-FFP

To be able to make a functional size measurement in an early stage of development an approximation method has been developed: “approximate COSMIC-FFP”. This method counts the number of functional processes to obtain an estimate for the expected software size.[5] Because of the nature of the design process in this environment (see section 1.4) this approximation method is far more useful than FPA-based methods that rely on early knowledge of the data model. [4]

The expectation was that the size obtained using COSMIC-FFP would reflect the effort needed to implement the process-chains, although a lot of the functionality was not actually built but parameterised from existing functionality (see section 333). If there is enough information about the processes to classify them into categories, they can be classified as:

- Small e.g. retrieval of information about a single object of interest
- Medium e.g. storage of a single object of interest with some checks
- Large e.g. retrieval of information about multiple objects of interest
- Complex

This method is called “refined approximate COSMIC-FFP”. The values that are used for each category are:

- Small 4 Cfsu
- Medium 7 Cfsu
- Large 11 Cfsu
- Complex 24 Cfsu

These values are taken from the Measurement Manual, although there are indications that these values are environment dependent. [5,6] The precision of the “refined approximate COSMIC-FFP” is about 20-30%. [6] Given the fact that it takes quite a lot of experience data to calibrate the method for a given environment, it was decided that using the values from the Measurement Manual without a local calibration would be “precise” enough for planning and estimating purposes until enough data would be available for local calibration.

3 The advantage of COSMIC-FFP in this environment

In the first version of the process model not all data interaction is known. But the description of the process is detailed enough to classify a process into one of the four categories of the ‘refined approximate COSMIC-FFP”. Based on only the process model it is possible to get an early estimate of the size of a process-chain with an uncertainty of 20-30%.

Most early sizing techniques are based on information about the data [4]. In this environment the process model was designed before the data model, which means that FPA-based techniques can only be used in a later stage.

The EBS contains its own data model. The process-chains make use of the EBS data model and a specific data model for the Office for Regulations. The contribution of the specific data model to the functional size can be determined with FPA-based techniques, for the EBS data model this is quite difficult.

The exact structure and the way modules make use of the EBS data model is proprietary information and cannot be determined for sizing purposes. There are no counting rules when and how the data model must be taken into account for FPA-based techniques. With COSMIC-FFP the sizing is more or less independent of the structure of the data model. So with COSMIC-FFP the fact that a part of the data model is not known does not influence the quality of the functional size measurement.

4 Functional size measurement with COSMIC-FFP

In an early stage of development the size of a process-chain is estimated by classifying each automated process in the process model into one of the four categories of the "refined approximate COSMIC-FFP" and add together the sizes of all the processes. (see section 2.2)

When the process description contains enough information about the data interaction a detailed COSMIC-FFP functional size measurement can be made. As stated in section 2.1 the functional size is determined by the number of data movements. Each data movement moves one data group. The ability to recognize data groups determines the ability to determine the functional size in detail.

A data group is defined as:

> *A data group is a distinct, non empty, non ordered and non redundant set of data attributes where each included data attribute describes a complementary aspect of the same object of interest.* [5]

An object of interest is defined as:

> *An object of interest is identified from the point of view of the Functional User Requirements and may be any physical thing, as well as any conceptual objects or parts of conceptual objects in the world of the user about which the software is required to process and/or store data.* [5]

These definitions show that the ability to recognize data groups is not dependant on the data model, but on characteristics that can be determined without knowledge about the structure in which the data groups are stored. A detailed process model is sufficient to make a detailed functional size measurement.

5 Planning

The first step in using size estimates for planning the production of process-chains was to verify whether a relation can be determined between the measured size and the expended effort. It appeared that for the software factory we had to make a clear distinction between the direct effort and the support effort. The main components of the direct and support effort are given in table 1.

Direct effort	Support effort
Design administrative organization	Architecture
Design custom-built software	Project management
Set-up design	ERP set-up
Build custom software	Process improvement
System test	Quality control
Integration test	Metrics office

Table 1: Main components of direct effort and support effort

The support effort has a linear relation with time and has no dependency on the size of the process-chain(s) to be produced. This can be explained from the fact that these activities are mainly related to the processes with which the software is produced rather than with the (size of the) software itself.

5.1 Time to delivery

The time to delivery of a process chain has an exponential relation with the size of the software. [7,8] The power is a function of the number of production lines that is used:

$$\mathbf{Time}_{\mathrm{Delivery}} = \frac{\mathbf{Size}^{\mathbf{Power}}}{\mathbf{PL}}$$

where:

Time	= Time to delivery of the process-chain in months
Size	= Functional size in Cfsu
Power	= 0,20 for a single production line and 0,37 for two production lines
PL	= Number of deployed production lines (1 or 2)

The exponent values have been empirically calculated based on the figures of two releases of EBS functionality. No exponent values have been calculated for three production lines, because this would mean that the total software factory should be dedicated to one single process-chain, which is a non-desirable situation. A process-chain that is so large that it requires more than two production lines is too large to control efficiently and must be cut into smaller process-chains.

5.2 Working method of the software factory

The working method of the software factory is based on Oracle's Application Implementation Method (AIM Advantage™) for implementing packaged applications. [9] This method consists of six project phases:

- Definition
- Operations analysis
- Solution design
- Build
- Transition
- Production

In each of the phases, the project team will execute tasks in several processes. The full model contains 11 processes. For the tasks of the software factory that affect the direct effort three major processes are relevant:

- Design (including business requirements definition and -mapping)
- Build (including module design & build and documentation)
- Test (including business system testing and performance testing)

In figure 3, these major processes are mapped to the AIM Advantage™ phases:

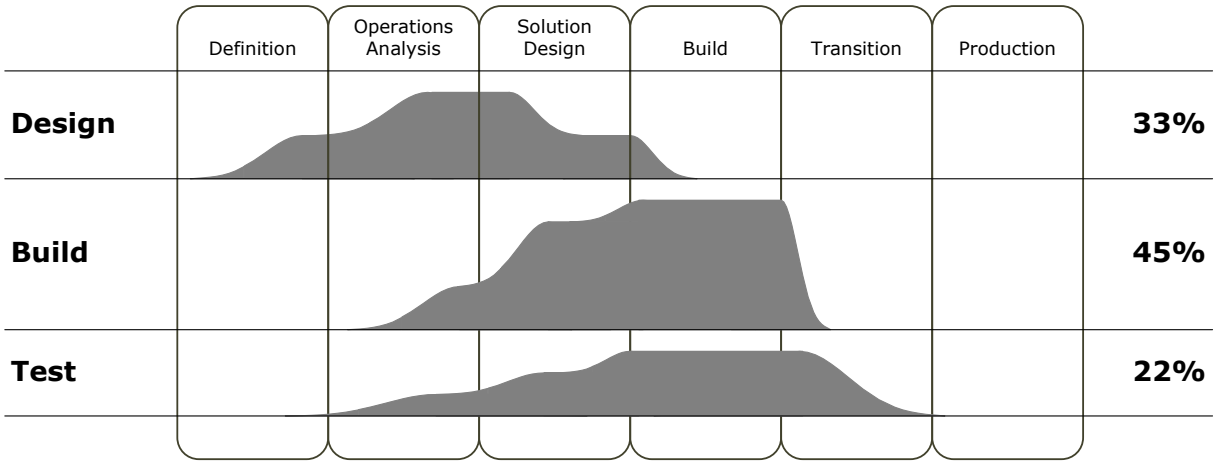

Figure 3: The working method of the software factory, related to direct effort

5.3 Planning production

The manpower build-up for the three major processes differs in time. This means that the production of a new process-chain can start before the production of a previous chain is completed. The production process can be planned in tiles, where the length of a tile is determined by the size of the process-chain to be produced.

5.4 Planning production line staffing size

A factor that is essential for the planning, is the staffing size of the production line. This size is scaled up or down according to the size and complexity of the process-chain to be produced. The size of a production line varies between 4 and 14 system engineers. This is consistent with general industry experiences for package customisations [7] in which a maximum team-size for package customisation is reported of 5 (median value) to 12,8 (75 percentile value).

The main factors that influence the staffing size of a production line are:

- the *size* of the process-chain(s) to be produced;
- the estimated amount of *reuse*;
- the *complexity* of the process-chain(s) to be produced;
- the relative *autonomy* of the process-chain(s) to be produced.

The *size* of the process-chain(s) is based on early estimates of the functional size from the first design of process model. This documentation is produced before the design activities of the software factory start.

The amount of *reuse* is currently determined as an expert estimate. The Office for Regulations is setting up a process-chain component library. This library will contain the components and their functional size. When this library is in effect the amount of reused can be determined more objectively.

The *complexity* is determined by an expert estimate of the number of quality plans[Q] that must be developed to produce the necessary user interaction.

The *autonomy* of a process chain is determined by the dependency on other process chains that make use of the same process-chain components.

At this moment we are not able to quantify all of these factors with enough precision to derive a formula to predict the required staffing size.

5.5 Client expectations

The early size estimates are very useful to manage client expectations. The implementation of the new regulations in the EBS environment is not only a system renewal, but also contains a lot of business process redesign. The Office for Regulations has to deal with both aspects at the same time. Some of the clients of the software factory do not have a good idea of the impact of the new processes in terms of EBS functionality. For these clients the size estimates proved to be a very effective tool in communicating that different process-chains can have a different software size.

[Q] A quality plan is a way to parameterise the Quality module for the required man-machine interaction.

6 Control

The functional size measurement is not only relevant for planning purposes, but also to provide management with information to control several aspects of the production process. At this point in time, the functional size measurement is used to control:

- stability rate
- direct cost (productivity)
- scope creep
- change management

6.1 Stability rate

The stability rate is the rate between the functional size to be produced per unit of time per production line. The target of the software factory is to produce at a level stability rate.

The stability rate can incline due to scope creep, which causes more production with the same staffing size (see section 6.3) and corrective maintenance on previous releases which causes a decrease of the effectively available staffing size. A small incline can be corrected by increasing the staffing of a production line. If the expectation is that the stability rate remains at a high level for a longer period of time an extra production line can be added. It takes approximately two months before a production line can produce at a regular stability rate.

The stability rate can decline due to late availability of new process models for process-chains. A small decline can be corrected by decreasing the staffing of a production line. If the expectation is that the stability rate remains at a low level for a longer period of time aproduction line can be dismantled.

6.2 Direct cost

The cost model for the direct cost within the Office for Regulations is on a fixed price per Cfsu to stimulate the software factory to improve productivity. With the formula for the time to delivery from section 5.1, the total cost formula can be described as a function of size and the number of production lines:

$$\mathbf{Cost} = \frac{\mathbf{Size}^{\mathbf{Power}}}{\mathbf{PL}} * \mathbf{Constant}_{\mathrm{Support}} + \mathbf{Size} * \mathbf{Constant}_{\mathrm{Direct}}$$

where:

Size	= Functional size in Cfsu
Power	= 0,20 for a single production line and 0,37 for two production lines
$\mathrm{Constant}_{\mathrm{Support}}$	= Daily support cost per production line in €
$\mathrm{Constant}_{\mathrm{Direct}}$	= Direct cost per Cfsu in €

The direct cost per Cfsu for has a target that is based on the productivity the software factory expects to reach after working process is stable. If the productivity improves, the cost per Cfsu decreases. The first process-chain was delivered early this year and the expectation is that the set productivity goal will be reached at the end of this year. Figure 4 gives an indication of the progress towards the productivity goal:

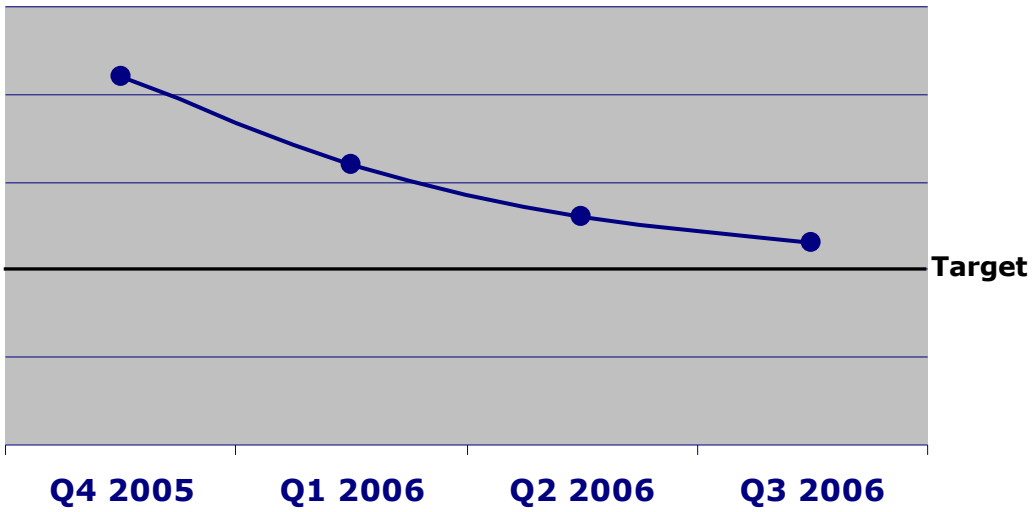

Figure 4: Cost per Cfsu versus the set target

In this environment, the support cost is a substantial part of the development cost of a process-chain. The support cost cannot be related to functional size, because there is no relation between the support cost and the functional size. The main factors influencing the support cost are:

- **Architecture** In parallel with the development of the process-chains, the working processes have to be redesigned. To support the business process redesign there is a heavy architectural support.
- **Project Management** Because of the parallel process redesign and system renewal, each production line has its own project manager. This means a relative high cost for project management in relation to the delivered functional size.
- **Promotion Management** The EBS environment has no tool support to promote parameterised components from one OTAP[!] environment to the next. This calls for a lot of manual labour to promote the components.

[!] OTAP = Separate environments for Build (*Ontwikkeling*), Test, Acceptance and Production

6.3 Scope creep

By comparing the early size estimate with the final functional size when the design is finished the scope creep can be determined. Scope creep is a measure for the stability of the requirements. Because of the fact that parallel to the system development there is also business process redesign, a high percentage scope creep can be expected, due to information latency between the parallel streams. At this moment, only the scope creep due to change requests is being measured. (see next section) This scope creep is around 13%. The total scope creep between the first functional size measurement and the delivered functionality is expected to be somewhat over 20%.

6.4 Change Management

With COSMIC-FFP, changes can be measured easily. [5] A substantial number of the process-chains implement new legislation that evolves after the first stage of design and generates a number of Requests for Change. To facilitate Change Management a simplified procedure has been developed so that each Request for Change can be sized by the developer that does the impact analysis:

- substantial change: 100% of the size of the functional process
- minor change: 50% of the size of the functional process
- deletion: 100% of the size of the functional process to delete
- new functionality: the size of a comparable functional process

The functional size that is determined by this procedure is used to evaluate the expert estimate of the impact of a Request for Change on the development of a process-chain. This procedure is now in a pilot stage.

6.5 Technology choice

The early size estimate is independent of the chosen technology. With the cost formula introduced in section 6.2 the cost of a process-chain produced with EBS technology can be calculated at an early stage. This cost can be compared with the cost of producing the same process-chain with another technology. This is only relevant for process-chains that will be used for a short period of time and are not obliged to comply with the strategic choice for the EBS environment. (see section 1.2)

7 Deviation from average packaged software implementations

Usually ERP projects implement one or more modules from a suite to support the primary business process of the customer. These kinds of implementations are usually estimated based on the relative weight of the module in relation to a core module that is used in most implementations. For ERP implementations, this core module is usually the General Ledger.

For the Office for Regulations, not only secondary business processes would be implemented, but also the primary business process. The Workflow module is preconfigured to support the BPM. The Quality module is used to combine the Workflow tasks with other used modules like Trading Community Architecture, Install Base, Service Contract and Advanced Pricing. [2]

This experience shows that COSMIC-FFP can be used for sizing and estimating this kind of ERP-project where the EBS modules are used as development environment instead of implementing them to support secondary processes.

8 Conclusions

The main reason to choose COSMIC-FFP as the sizing metric was based on the kind of documentation that is produced in the early stages of development. When only process information is available, (approximate) COSMIC-FFP has an advantage over other sizing metrics like function points.

This experience shows that functional size measurement can be used for sizing and estimating this kind of ERP implementation that requires a lot of customisation. This knowledge is mainly important for large-scale implementations for multinational companies that cannot change all their processes to the processes required by the ERP modules as-they-are and for special implementations that require a lot of non-standard functionality. Examples of this kind of implementations are typically found in the military.

9 Proposed further research

Further work on mapping the COSMIC-FFP concepts to ERP concepts would be beneficial for both fields. Based on this experience it cannot be concluded that COSMIC-FFP or any other kind of functional size measurement is suitable for estimating, planning and controlling regular ERP implementations. This experience only shows the applicability for a special kind of ERP implementation.

Further research should concentrate on the possibility of using functional size measurement to support the current practice of estimating ERP implementations based on the relation with a core module like General Ledger.

Affiliation

The author has been working as a practitioner and consultant within the area of software metrics since 1999. Within this area, he specialised in sizing and performance measurement programs within client organisations. He is a metrics consultant and scope manager for the centre of expertise in Sizing, Estimating and Control of Sogeti Nederland B.V. He is a member of the Measurement Practices Committee of COSMIC and a member of the COSMIC working group of NESMA.

REFERENCES

1 Commission of the European Communities, Explanatory memorandum: A long-term policy perspective for sustainable agriculture, COM(2003) 23 final, Brussels, http://ec.europa.eu/agriculture/capreform/memo_en.pdf

2 Oracle E-Business Suite, http://www.oracle.com/applications/e-business-suite.html

3 Kranenburg, K., Nelissen, F., Brouwer, J., De moderne softwarefabriek: De organisatie en werking van een software development & maintenance center, Ten Hagen en Stam, Den Haag, 1999 *only available in Dutch*

4 Meli, R., Early and Quick Function Point Analysis – From summary user requirements to project management, in "IT Measurement: Practical advice from the experts", Jones, C. and Linthinicum D.S. (eds), p 417-441, Addison-Wesley, Boston, 2002, http://www.ifpug.org/publications

5 Abran, A., Desharnais, J.M., Oligny, S., St-Pierre, D. and Symons, C. (eds), COSMIC-FFP Measurement Manual (The COSMIC implementation guide for ISO/IEC 19761:2003), version 2.2, january 2003, http://www.cosmicon.com

6 Vogelezang, F.W., Early estimating using COSMIC-FFP, Proceedings of the 2nd Software Metrics European Forum (SMEF), March 16-18 2005, Roma (Italy), http://www.iir-italy.it/smef2005

7 McConnell, S., Rapid development – Taming wild software schedules, Microsoft Press, 1996, http://www.stevemcconnell.com/rd.htm

8 Hill, P.R., The benchmark release 8 – Analyses of the factors that affect the project duration, quality & productivity of software development & enhancement projects & package customisation projects, January 2004, http://www.isbsg.org

9 Oracle, AIM Advantage™ : A comprehensive method and toolkit for implementing Oracle's packaged applications, Oracle White Paper, Revision 1.0, September 1999 http://www.oracle.com/consulting/collateral/AIMadvantage.pdf