International
Software
Benchmarking
Standards
Group

Software Development Projects in GOVERNMENT

performance, practices and predictions

This report provides a global view of the best practices of software development and acquisition by Government. It uses the ISBSG data repository of over 2,000 projects to examine current practices and productivity. It provides a series of Case Studies highlighting best practice within government.

Messages for Government Executives

This section is intended for those executive level managers who are the owners, sponsors and members of the project boards for software projects within government. Here we provide a snapshot of the state of software projects in the Government Sector and suggest some actions that management may take to help improve the chances that software projects will succeed.

An end is in sight for dismal success rates of software projects

According to the Standish Group CHAOS report of 2003:

- 15% of software projects are terminated before they produce anything
- 66% are considered to have failed
- Of those that do complete the average cost blowout is 43%
- The lost dollar value for USA projects in 2002 is estimated at US$38bn with another US$17bn in cost overruns.

However, this performance is a significant improvement on that reported in the previous years.

Over the past decade a global body-of-knowledge of software project histories has been created around standardised language and measurement concepts. This is driving a transformation in the way government and business estimates, plans and manages the acquisition of its software applications. Additionally, in most countries there has been a growth in companies specialising in software metrics that understand the new capabilities and provide services to government and business as to how these can be applied.

Some facts about software productivity in government

The International Benchmarking Standards Group, (ISBSG), is a not-for-profit organisation whose membership currently comprises the peak national software metrics associations of eleven countries. For over a decade the ISBSG has been developing standards and has established a repository of the project history of over 2,000 projects from 30 countries. This repository forms the basis of this report. It must be noted that projects in the ISBSG repository are not a random sample but probably represent the best 25% performing global software projects.

An analysis of Government sector projects indicates that they generally performed better than the non-Government sector projects. The highlights are as follows:

- The Government projects were 8% more productive than non-government projects. By that we mean that more functionality was delivered per developer hour.
- The speed of delivery of Government projects is slightly better, (measured by the number of function points delivered in a month)
- On average, Government projects are being delivered 37% later than their delivery estimate and 22% of projects exceed their cost estimate.

The analysis also revealed some interesting factors that affected software development productivity:

- Projects that were developed in-house were 14% more productive than outsourced projects.
- Project teams that remained stable, (i.e. they did not experience staff changes), were significantly more productive than those with staff turn-over.

Stop kidding yourself - work to realistic project expectations

In the past project boards have had no choice but to accept estimates of cost and duration based on little more than the project manager's 'best guess'. This was often driven by the business imperatives that dictated the date the software was to become operational and what the budget was to be. There was usually little understanding of the real size of the job and what performance was possible in that environment. Working to unrealistic estimates is a major factor in project failure.

With the new body-of-knowledge it is now relatively easy to obtain very early estimates of size, cost, effort and duration that are soundly based on project histories.

Be prepared to stop projects

While often feeling discomfort that a project may be going 'off the rails', without objective data to confirm this, Project Boards are often compelled to let projects run to inevitable failure.

Using techniques to monitor progress against the project plan, it is now possible for Project Boards to objectively track a project's progress and to take early remedial action. These services can be provided by industry specialists, known as Software Scope Managers. If a project's functional requirements are unclear, the project should be stopped until they are clarified and agreed.

Recognise and use the industry specialist skills

During the 90s it was widely thought that software metrics skills should merely be a part of the skill set of any capable project manager. With the sophistication of the body-of-knowledge and the tools and skills to apply it, this thinking is now outdated. A good analogy can be made with the building construction industry where a specialist profession of what is known variously as 'Cost Surveyors' or 'Quantity Surveyors' have a role throughout the project to support estimation and project planning, to track progress and to monitor and resolve scope creep.

Within software engineering, this role is being undertaken by Scope Managers who normally work for independent companies specialising in software metrics. Scope Managers also provide services as diverse as benchmarking an organisation's software development performance or valuing its software portfolio.

Consider buying software on the basis of cost-per-unit-delivered

Traditionally, any government wanting to acquire software did so in one of two ways: either by developing a requirements specification and inviting fixed price offers to deliver the software application, or by engaging developers and paying on a 'time-and-materials' basis.

The new capabilities have created a third option - buying software on the basis of cost-per—delivered-functional-unit. The best manifestation of this method is the *southernSCOPE* methodology that has proven to provide substantial benefits over the traditional approaches.

Table of Contents

Context

Acknowledgements

This report was sponsored by the Government of the State of Victoria, Australia with the intent of helping Governments worldwide to improve their management of software development projects.

ISBSG also acknowledges the following people for their advice, assistance and review of this report: David Cleary (Charismatek Software Metrics), Pam Morris (Total Metrics), George Ansell (ISBSG), Rob John (MTC), Pekka Forselius (STTF), John Best (Victorian Government), Prof. Chris Lokan, (UNSW – ADFA), and Terry Wright (ASMA).

Copyright Statement

Purpose of this Report

The ISBSG Sector Reports are seen as a way to redress the imbalance between current capability and current practice. These reports will document the current best management practices and the capabilities in a specific industry sector. The reports are also for suppliers to the target industries, so that they may better understand factors influencing the sector and better tailor their services. The case studies and analyses within the report will focus upon the particular sector in order to maximise the learning opportunities.

This report will focus on: *what the best Government organisations are doing and what the evolving body-of-knowledge will allow*.

This Report has been sponsored by the Government of the State of Victoria, in Australia, which for many years has pioneered the application of the new body-of-knowledge to improve its software management practices.

Compiled and edited by Peter R. Hill January 2004

About the International Software Standards Benchmarking Group (ISBSG)

The International Software Benchmarking Standards Group is a not-for-profit organisation that was established in 1997. Its formation was built upon several years of previous cooperation by a group of national software metrics associations that were trying to develop and promote the use of measurement to improve software processes and products for the benefit of both business and government.

The mission of the ISBSG is:

To help improve the management of IT resources by both business and government

through

the provision and exploitation of public repositories of software engineering knowledge which is standardised, verified, recent and representative of current technologies.

The current membership of ISBSG comprises the following:

Australia
Australian Software Metrics Association (ASMA) www.asma.org.au

Finland
FiSMA (Finland Software Metrics Association) www.fisma-network.org

Germany
DASMA (Deutschsprachige Anwendergruppe fur Software Metrik and Aufwandschatzung) www.dasma.de

India
NASSCOM (National Association of Software and Service Companies) www.nasscom.org

Italy
GUFPI - ISMA (Gruppo Utenti Function Point Italia - Italian Software Metrics Association) www.gufpi.org

Japan
JFPUG (Japan Function Point User Group) www.jfpug.gr.jp

Netherlands
NESMA (Netherlands Software Metrieken Gebruikers Associatie) www.nesma.nl

Spain
AEMES (Asociacion Espanola de Metricas de Software) www.aemes.org

Switzerland (Associate)
SWISMA (Swiss Software and Service Metrics Association) www.swisma.ch

United Kingdom
UKSMA (UK Software Metrics Association) www.uksma.co.uk

USA
IFPUG (International Function Point Users Group) www.ifpug.org

How this report is arranged

Analysing the ISBSG repository data to study both traditional and new technologies, and using resources of the ISBSG member countries, this report provides the following:

1. Messages for government executives. We provide a snapshot of the state of software projects in the Government Sector and suggest some action management may take to help improve the chances that software projects will succeed.. This section is aimed at non-IT executives.
2. Answers to some of the most common concerns of government relating to their management of software projects. For example:
   - ❖ *How do I reduce the chances of my software projects failing?*
   - ❖ *What are the most effective development infrastructure choices for the future?*
   - ❖ *How do I ensure good value for money from my outsourced functions?*
3. A detailed analysis of how the performance of the Government sector compares with other industry sectors, (cost, productivity, time to market and delivery rate).
4. An analysis of the various performance measures for projects within Government with comment on those factors that impact either positively or negatively on performance, (such as in-house versus outsourced, project size and personnel changes).
5. A report on the development infrastructure components used in the industry, looking at the changes over time and the likely future trends, (programming languages, tools, techniques, methodologies, platforms).
6. Case studies and practical solutions specifically focussing on Government software projects.

In this report we have drawn on the combined knowledge of the eleven international ISBSG Members to:

- ❖ Identify and document worldwide best practice case studies from within the Government sector
- ❖ Identify and document the current best available resources, methodologies and approaches
- ❖ Predict the capabilities that will become available as the body-of-knowledge continues to evolve.

A new body-of-knowledge is created

Introduction

In recent years huge strides have been made in planning and managing government software projects so that they deliver required business functionality within a defined budget and time. One Government has almost halved its outsourced software development costs on specific projects by applying the new body-of-knowledge that the industry now has available. It consistently delivers its projects on time and within budget. This report not only describes the state of Government Sector I.T. software development, using the ISBSG body-of-knowledge, but provides case studies and "learnings" that can be applied to gain the same successes as those enjoyed by the case study organisations.

Using a new approach, one Government has almost halved the cost of specific outsourced software development projects.

"Those who ignore history are doomed to repeat it". Software engineering now has a body of history-based knowledge that is providing new business benefits to organisations in all sectors. Government organisations that have not discovered the new capabilities available through this new body-of-knowledge are missing opportunities for greater predictability, a new level of control, the ability to manage changing requirements and improved performance from their software projects.

Background and the Need

Traditional Management of Software Applications

The scoreboard for software engineering in developing business applications is dismal[1].

- 15% of software projects are terminated before they produce anything
- 66% are considered to have failed
- Of those that do complete the average cost blowout is 43%
- The lost dollar value for USA projects in 2002 is estimated at US$38bn with another US$17bn in cost overruns.

Spending on software projects in the USA alone is estimated to be in excess of US$250bn annually. According to IDC, Governments account for 10% of global IT spending.

The price government and business pays for comparable software varies by a factor of more than 10:1.[2]

So why is the performance of the software engineering industry so poor? There are probably two primary reasons. Firstly, the software industry is a relatively young industry. Unlike most other engineering disciplines, until recently software engineering has not had access to mature and proven tools that allow it to predict, plan and measure progress and productivity. Secondly, again until relatively recently, there has been no reliable and broadly accepted technique for measuring the output of a software engineering project. The building industry talks about *square metres* of floor space, in road construction *kilometers* of highway; but

[1] Standish Group CHAOS Report, 25 March 2003

[2] From analysis of the ISBSG Repository

software engineering had no similar size measure. Over the last two decades these problems have been largely overcome.

The Evolution of Good Measurement Capability

In the late 70's the first of what are now known as software *functional size measurement methods* was developed. These methods measure the size of the functionality provided to the end user, irrespective of how the software is developed. They are now mature, being reliable, repeatable, accurate and consistent. The most commonly used functional units are known as *function points*. There is now an ISO standard covering functional size measures. To continue the analogy with the building industry, whereas the average house size is about 150 square metres the average software project size is about 300 function points.

This newfound ability to measure output laid the foundation for a whole new management capability.

Software development projects can be reliably sized, just like a building or a road projects.

The Software Project Body-of-knowledge

To meet its mission, the ISBSG sets standards to measure software development performance and using these standards has created a repository of more than 2,000 projects from over 30 countries.

Using this repository the ISBSG has produced a range of analyses, reports, tools and services which now offer business and government project management capabilities never before possible.[3] Although the capability now exists within software engineering to manage in the same way as the mature engineering disciplines, sadly, common practices within the software industry remain virtually unchanged from those practices of the '70s.

[3] Refer to Appendix C

Performance Analysis

In the following analyses of project performance we have attempted to answer some key questions:

- ❖ Am I paying too much for my software development?
- ❖ Is outsourcing better value than having software development done in-house?
- ❖ What level of productivity should I expect in the various development environments?
- ❖ What speed of delivery is realistic using various development languages and tools?
- ❖ What development language types and tools deliver the best results?
- ❖ What factors impact positively or negatively on Government software projects?
- ❖ What is the trend of programming language use over time?
- ❖ What is the accuracy of software project estimates?

It is important to remember that the data used for this analysis probably represents the best-performed 25% of software projects. The thinking behind this belief is that any organisation that decides to submit project data to the ISBSG is likely to be quite mature in its project development processes. Projects have generally been submitted by organisations seeking best practice. In addition, these projects have actually completed.

Am I paying too much for my software development?

There are two levels of answer to this question: First the performance of Government software projects, and secondly, the productivity and performance that you should expect from your development suppliers, whether in-house or outsourced.

The Government projects were 8% more productive than the non-government projects.

The Government projects were 8% more productive than the non-government projects and had a much smaller variance range from the best to the worst projects.

Government projects enjoyed a median productivity of delivering one function point of functionality in 9.7 hours of developer effort. This is a good indicator of the sort of productivity you should expect from your software development supplier, whether in-house or outsourced. However, this will vary depending upon the development environment. We have provided tables in the Appendix A so that you can establish a more accurate productivity expectation for your particular environment.

Is outsourcing better value than having software developed in-house?

The small sample of projects available for this analysis showed that the in-house developed projects were almost 14% more productive than the outsourced projects. Once again the Government projects compared favourably against the non-government projects, in-house Government projects being almost 27% more productive, outsourced projects being almost 30% more productive. (Refer to the tables in the Appendix A for the sample sizes and Function Point details).

Government projects developed in-house were almost 14% more productive than the outsourced projects

What level of productivity should I expect from the various development environments?

The ISBSG gathers data by three types of development platform, Mainframe, Midrange and PC. “Development platform” has been found to be the best indicator of the environment in which the software is being developed. So the term more correctly refers to the whole environment/process, not specifically the hardware platform.[4]

Government Mid-range based projects were twice as productive as similar non-Government projects.

So what does the productivity of the Government specific projects look like across the three platforms? The analysis for this report showed that once again, in total, the Government software projects were considerably more productive than the non-Government projects. Government Mainframe projects were 15% more productive, Mid-range projects were twice as productive and PC projects 40% more productive.

What speed of delivery is realistic using various development languages and tools?

Speed of delivery, has in recent years overtaken productivity as the most important factor of project development. Management emphasis has swung to rapid deployment, even at the expense of productivity and cost. The median speed of delivery of the Government projects in the ISBSG repository is 38.2 function points delivered per month. Government projects perform slightly worse than non-Government projects which show a median of 40.7 function points delivered per month.

Government projects have a median speed of delivery rate of 38.2 function points delivered per month.

It is worth noting that Upper CASE tools used for specification activities deliver improved speed of delivery performance, with the Government projects recording a median of 47.2 function points delivered per month compared to 51.7 for non-government projects.

What development language types and tools deliver the best results?

Development productivity differs with the development language type and individual languages. Government projects using 3GLs have a median productivity rate of 13.3 hours per function point. For 4GLs this improves to 5.2 hours per function point. There are too few government projects in the ISBSG repository that used Application Generators to provide a meaningful comparison. But the non-Government projects using Application Generators report an impressive productivity rate of 7.8 hours per function point.

Non-Government projects have slightly worse productivity than Government projects for 3GLs, (14.7) and 4GLs, (8.7).

What factors impact positively or negatively on Government software projects?

As well as development environment and language type, we look at two other factors that have an impact on the Government projects in the repository: changes of project personnel and the

[4] Refer to Appendix B: Notes on Development Platform and Development Process

size of the user base, (number of users, the number of locations and the number of business units), that a system is to be deployed in to.

Changes of project personnel have an adverse impact on productivity. Eighteen government projects indicate a number of personnel changes, (either of project manager or in the development team). For the ten projects where no changes occurred the median productivity is 6.7 hours per function point, which is substantially better than the 9.7 hours per function point for all government projects. For the eight projects where personnel changes occurred, (a single change in all cases except one), the median productivity is 9.0 hours per function point.

Government projects that did not experience project personnel changes were significantly more productive than those that did.

The size of the user base has an effect on productivity. One hundred and thirty government projects indicate the size of user base, (either number of users, locations or business units). The median productivity of projects with lower than average **number of users** is 10.5 hours per function point, while productivity of projects with higher than average number of users is 18.1 hours per function point.

Projects that cater for a larger **number of locations** are less productive. The median productivity of projects with less than the lower than average number of locations is 10.4 hours per function point, while productivity of projects with higher than average number of locations is 26.3 [MSOffice1]hours per function point.

The median productivity of projects does not vary with **number of business units**, being 10.4 hours per function point. This is not unreasonable if the functionality is constant across business units.

What is the trend of programming language and platform use over time?

The Government projects in the ISBSG repository have been implemented over the last decade (1991 to 2001). There is no discernible trend in project effort over this period but productivity has decreased. The average size of the Government projects being submitted to the Repository is reducing. This may be an artefact of the small number of projects per year available for analysis, however this trend has also been observed in non-government projects.

In the first half of the decade the programming languages being used were Clipper, Natural and PowerBuilder. Recent years have seen a move to Oracle, Notes, Visual Basic, Java and HTML. Cobol has been used throughout. Over this ten-year period there has been a noticeable change in the languages used and an apparent shift away from Application Generators. These trends are also apparent in non-government projects.

Recent years have seen a move to Oracle, Notes, Visual Basic and Java.

There is no discernable trend in development platform usage over the ten-year period other than an increasing use of Client/Server.

What is the accuracy of software project estimates?

Twenty-nine Government projects provide estimate data. Within this data-set, 24 give an estimate of implementation date ranging from a 14% earlier implementation than the estimate, to a 101% later than estimate, the average lateness being 37%.

On average, Government projects are being delivered 37% later than their estimate and 22% over their cost estimate.

Nineteen projects give an *effort estimate* ranging from zero difference (1 project), to 76% under-estimated. On average project effort has been under estimated by 26%.

Fourteen projects give a *cost estimate* ranging from 1 project coming in 3% under-estimate to 114% over-estimate. On average, project costs have been under estimated by 22%.

Best Practices, Resources and Methods

In this section we offer some ideas aimed at increasing your awareness of the new capabilities that may help you to improve your performance, get more for your money and achieve core business benefits on time and within your budget.

Apply 'reality checks' to your early estimates

Given the known likelihood of cost and time over-runs, the ability to check the reality of project estimates prior to funding approval is an opportunity too good to miss.

Such a 'reality check' is easily achieved. The estimates for the project being proposed are compared to the actual results of completed projects with similar characteristics from the ISBSG Repository. The results of such a comparison will provide a clear indication of the chances of the proposed project being delivered within its estimated effort (cost), and duration.

For example if the reality check indicates that only 20% of similar projects in the Repository were completed within a similar effort (cost), and/or duration, then it is highly unlikely that the estimates for the project being proposed are realistic.

Improve the overall accuracy of your estimates

If a builder does not know the approximate size of a building, then a reasonable estimate of cost and time to build cannot be provided. In the same vein, if you have no idea how big a proposed software project is, it is impossible to estimate a cost and time. Functional size measurement methods have been developed, matured and refined to the point where they now provide reliable, repeatable, accurate and consistent sizing of software development projects.

There are techniques that provide size estimates at the various stages of a project, from initial concept through to detailed requirements specifications[5]. This is very similar to the building example. A builder can provide you with an initial estimate based on the likely size of the proposed building and your broad description of its features. Such an estimate can provide an indication as to whether the building is within your budget. The estimate is then refined as more information is provided until eventually a fixed price might be quoted based on the architect's detailed plans.

Similar to Quantity/Cost Surveyors in the building construction industry, the IT industry now has its own specialists applying software metrics. These specialists provide advice on project and application sizing then they manage the scope of a project throughout its development. They are commonly referred to as Software Scope Managers.

Benchmark your current position

It is important to establish the current performance of software development in your organisation prior to considering any changes. The body-of-knowledge contained in the ISBSG Repository provides the data needed to perform such a benchmark.

When you benchmark your projects against those in the Repository it is important to select projects for comparison that have the same characteristics as your projects. The main characteristics might be:

- Government projects

[5] Refer to www.isbsg.org - Project Sizing - Sizing to Suit Your Needs

- Development Platform[6]
- Project Size (in functional units)
- Primary Programming Language
- Business Area (eg a government department)

When comparing projects against those in the Repository remember that the projects contained in the ISBSG Repository probably reflect the best performed twenty-five percent of the industry, given that most organisations that that have submitted projects are probably quite mature in their development processes.

Once you have established how your organisation compares to the ISBSG projects, you can then make rational decisions for changes or improvements to your development practices.

Acquire software on the basis of ‘cost per delivered functional unit’

Now that software projects can be reliably sized and a body-of-knowledge exists to provide productivity guidelines, there is no longer any excuse for “flying blind”. Once you have established the characteristics of a project, you and your prospective developer, can establish an expected productivity rate for the project using any in-house project history that is available plus the ISBSG history data. This will then allow you to agree to cost/price the project on a cost per functional unit delivered. This method has proven to be highly effective in allowing projects to be developed to a realistic budget and, in the process, removes the ongoing arguments between customer and developer about scope creep that are a normal part of the software engineering environment as it currently exists.

Use Scope Managers

Rethink the way your projects are managed. The two principal parties involved in a software project are the project customer and the developer. The well documented problems of project over-runs result from miscommunication, misunderstanding or misbehaviour within these parties. A Scope Manager is an independent party who specialises in understanding and applying the software metrics body-of-knowledge. This person is also able to act as a mediator and, if required, as a “referee”.

Use a Scope Manager to:

- Perform a preliminary functional sizing of the proposed project, using the ISBSG project history data plus other sources and experience, to provide an early but sound cost estimate and a realistic development timeframe for the project. The estimate will be based on a price per functional unit.
- Help the customer evaluate a project proposal, carefully checking that the price per functional unit being offered is realistic for the development environment in question. The ISBSG data is used to ascertain realistic effort hours required to deliver a functional unit for the required combination of platform, programming language etc. A very low price might indicate developer incompetence or naivety.
- Conduct a functional sizing of the Requirements Specification. This provides an opportunity to ensure that the Requirements Specification is complete. The scope manager will not be able to successfully size a project from an inadequate specification. Using the established functional unit size, the customer can then decide exactly what functionality will be needed to produce the required project outcomes, while meeting the budget and the delivery date.

[6] Refer to Appendix B for a description of the significance of Development Platform

- ❖ During the project ensure that changes to the scope are understood and that the project customer and the developer agree on their price and time impact. The customer might query a lack of functionality that it thought was to be delivered. The scope manager then acts as a "referee", stating whether the functionality in question was, or was not, in the specification and subsequent functional unit size. If it was included, then the developer has to deliver it. If it was not included, then it can be sized and the customer can decide whether it still wants the additional functionality, based on the known cost per functional unit. The same procedure applies if the developer claims that deliverable functionality was not included in the specification. This keeps both the customer and the developer "honest".

Consider the effects of Languages, Tools and Techniques

When deciding the best development environment for your organisation's software projects, there is no 'silver bullet' that will immediately provide a quantum leap in productivity and time-to-market. Each development environment should be tailored to the goals and objectives of an organisation.

However, the following provides a look at some of the more interesting findings regarding languages, tools and techniques:

CASE Tools – There is no evidence that case tools, by themselves, provide any productivity improvement. Of course the tools themselves may not be the problem. It may result from such factors as inadequate training, lack of developer experience or staff turnover on the project team.

Languages – ACCESS and Visual Basic provide the best productivity, with COBOL, JAVA and C++ offering only half the productivity, or worse, of Visual Basic. It is important to think beyond the language in isolation and consider the types of projects that the various languages might be used for and the platforms that they are normally associated with.

Techniques – Projects that utilise Object Oriented analysis record significantly better productivity than those that don't. Prototyping, Rapid Application Development and Timeboxing are the other techniques that stand out as offering considerable productivity improvements. The trade-off between productivity and quality has to be considered. The use of some techniques might lower productivity but result in a higher quality or a more robust system.

Over the Horizon

In this section we present some ideas about possible future developments in software project management practices and methods resulting from the availability of reliable project sizing methods and the growing body of software metrics knowledge.

Software project governance

Business executives who are charged with approving and funding new software intensive projects now have a means to 'reality check' the project estimates being presented to them. Such executives can now ask their Project Managers to extract relevant metrics from like projects in the ISBSG repository to provide them with a comparative means to determine the extent of exposure or project risk. They can compare the presented project estimates with similar international projects' durations, development effort, productivity rate and the like. In support of this governance need, the ISBSG is providing a basic 'reality checking' tool as a free service on its web site. Deeper analysis is possible utilising the complete body-of-knowledge, including the source project metrics available via the range of ISBSG products offered[7].

Using external Scope Managers will become the norm

In the future the project manager will be relieved of the burden of sole responsibility for providing estimates of a project's size, cost and duration. A Scope Manager will be employed for the life of a project to provide initial estimates. They will then provide an ongoing project monitoring role where they will take an external view of the project and advise the project manager how the project appears to be tracking. They will also mediate between the project manager and the customer on matters of changes to scope.

Fixed price development contracts will be used less

Organisations will abandon lump sum fixed price quotations for software based solely on the best guess of experienced developers. Functional sizing will provide a consistent base on which price per functional unit quotations can be provided. This ability to define an objective size and cost of functional components of a system, will allow informed decisions to be made about the need for each component based on the known cost and time impact. It is likely that projects will become smaller and be delivered on time as they deliver core functionality and abandon the "nice to have" scope creep that causes cost and time blow out.

Growth of a software metrics support services industry

Organisations will seek outside support services to provide functional sizing and scope management. This will cause consulting companies to realign their consultancy offerings to provide whole-of-project scope managers with the certification, skills and experience to deliver services from initial functional sizing through to successful project delivery. The impact of this will be the practical application of software metrics in mainstream day-to-day project development. Accreditation of software metrics practitioners will evolve.

Software metrics requirement for CMMI & SPICE certification

More and more organisations will seek recognition of their mature development process capabilities and demonstrate this achievement either through CMMI or SPICE certification or at least conformance to the requirements. This will increase the demand for and use of the

[7] Refer Appendix C

body-of-knowledge and the services of specialists in functional sizing and software metrics. CMMI & SPICE demand a knowledge and use of metrics well beyond the current level of practice in the majority of organisations.

Executives will cut their losses earlier

Executives will terminate more software projects before they turn into major business disasters.

Why is it that more 'run away' projects are not stopped before they become disasters? Perhaps senior executives are simply not being given independent factual information on which they can base a tough business decision. The ability to obtain early, low-cost reality checks and the increasing use of independent Scope Managers will have a major impact. Senior executives will be able to understand the real chances of a project's success or failure well before they have 'backed themselves into a corner'. If a project does go off the rails the evidence will be obvious and unbiased. Making a decision to close the project before it becomes a disaster should be much easier than in the past.

Better ways to manage packaged software implementation

There will be an increasing demand for data and processes to better manage package implementations. As more and more use is made of package software, performance history metrics will be gathered and published to help organisations size, plan and manage package implementations, which have in the past suffered delivery problems as bad as or worse than software development projects.

Threat to management capability for outsourcing contracts

Outsourced development projects will be compromised by the lack of in-house personnel capable of managing the outsourcer.

With the heavy cost cutting, downsizing and move to outsourcing, many organisations have underestimated the need to actively manage outsourced projects and have unwittingly lost personnel capable of managing these projects. There will be a need to employ either in-house or consulting resources to ensure the successful delivery of outsourced projects. There will also be a demand for a price per unit approach for the non-software development components of projects to complement the price per functional unit for software.

Improved emphasis on management of software maintenance

Management and pricing of software maintenance will become a higher priority as companies seek to control their overall software costs. Functional size methods are already being employed to improve cost management of software maintenance.

Sizing projects accurately early in the life cycle

Project sizing techniques and tools will continue to be improved to provide more accurate project sizing at the modelling or requirements gathering stages. This will allow decisions about project feasibility to be made earlier in the life of the project.

Best Practice Case Studies

CASE STUDY

Victorian State Government – Australia. A standard new approach.

The problem

The Victorian Government suffered the same problems with its software development projects as the majority of businesses. It regularly exceeded its budgets and rarely met time deadlines.

The Victorian Government's previous approaches to paying for software development were either based on a fixed price proposal using specified requirements or paying an agreed rate for inputs such as time and materials. These approaches were unsuccessful in achieving projects that consistently delivered the required functionality on time and within the agreed cost. (The average cost blow out in the industry at the time was 84%).

The Victorian Government set out to find an approach to manage outsourced software development that would address many of the reasons why software projects fail .

A solution

In an attempt to improve the success of its software projects, in 1996 the Victorian Government developed a new project management approach to software development, (now called *southern*SCOPE), to help it avoid software development acquisition budget and delivery blowouts. This method was trialed and refined on a number of government projects and has proven to be extremely successful.

The approach is similar to the building industry that develops its initial costings based on a square metre of floor space, or the road construction industry, which costs projects by the kilometre.

The methodology that the Victorian Government developed allows its project managers to invite tenders for software development projects based on cost-per-delivered-functional-unit, rather than on how much time has been spent on the development.

The functional size unit used has been the *function point.* This is a unit of measurement that puts a number to the amount of functionality delivered to the users by a software application. *Function points* are to a software application what *square metres* are to a house. Whereas the average house is about 150 square metres, the average software project is about 300 function points.

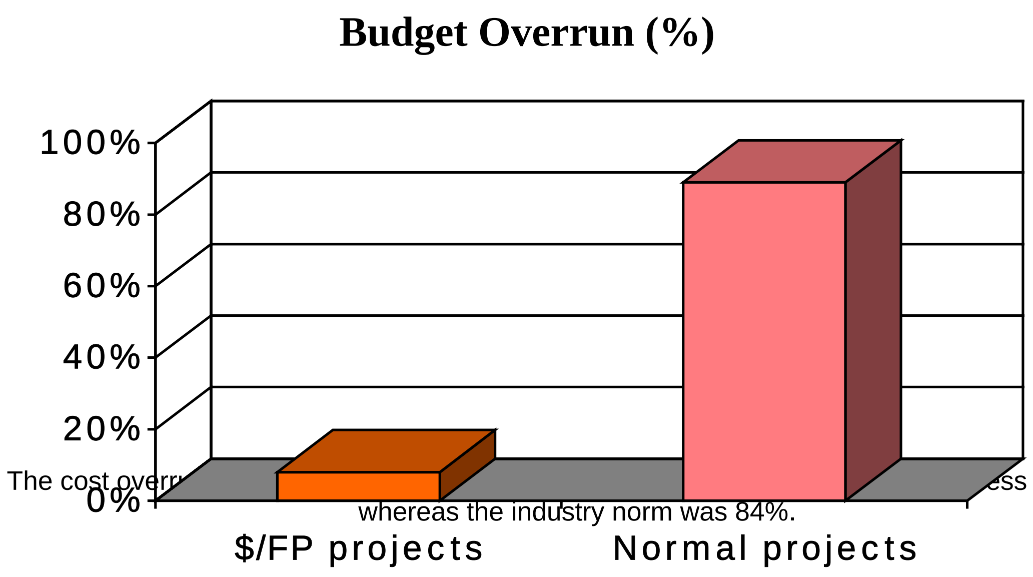
How does the method work?

These are the eight steps of the *southern*SCOPE method.

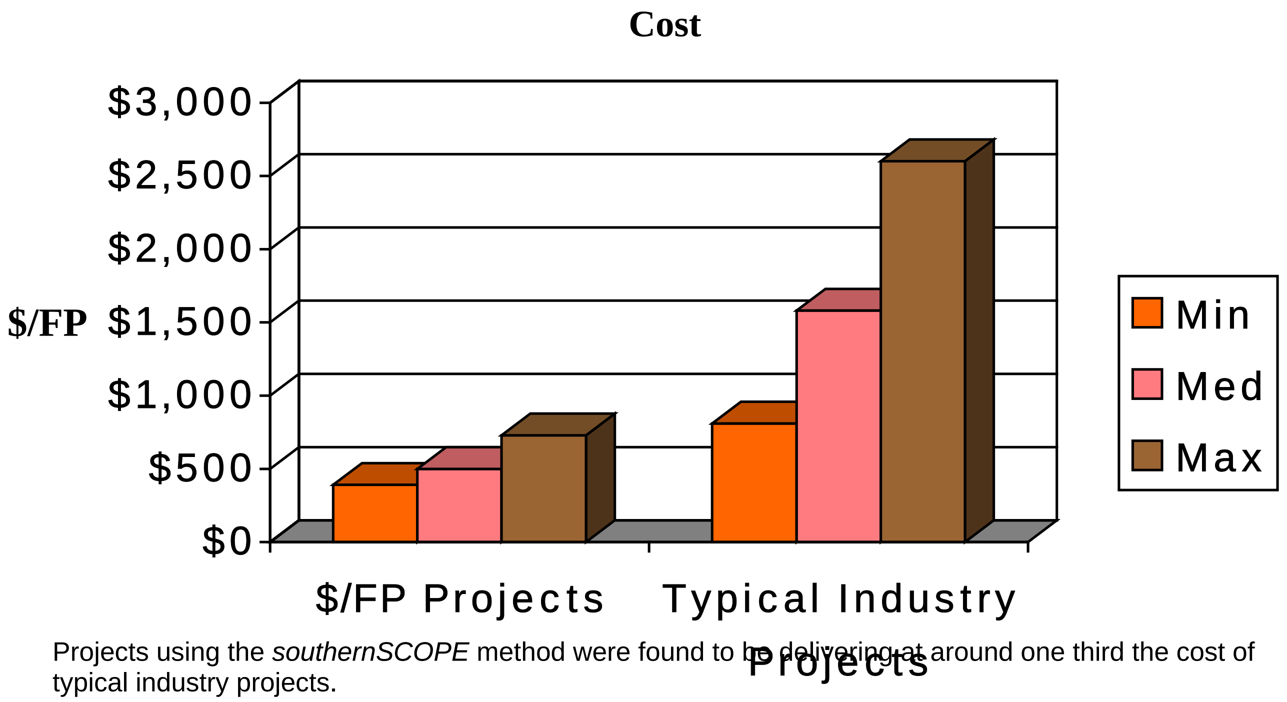
1. A Victorian Government department identifies that a need exists to acquire application software and engages a 'scope manager'. The scope manager is an independent person who specialises in software measurement.
2. The scope manager performs a preliminary function point count to size the proposed project and using the ISBSG project history data, provides an early but sound cost estimate and a realistic development timeframe for the project.
3. The department prepares a short document outlining the need for the software and the constraints of the project and invites proposals to develop the software based on a price per function point basis.
4. The department selects a proposal, carefully checking that the price per function point being offered is realistic for the development environment in question. (The ISBSG data is used to ascertain realistic effort hours required to deliver a function point for the required combination of platform, programming language etc.). A very low price might indicate developer incompetence or naivety. The department then engages a developer, agreeing payment for the software based on the dollars per function point of the software delivered.
5. The development begins with a phase of analysis that produces a Requirements Specification.
6. The scope manager conducts a function point count on the Requirements Specification. This provides an opportunity to ensure that the Requirements Specification is complete. The scope manager will not be able to successfully size a project from an inadequate specification. Using the established function point size, the customer decides exactly what functionality will be needed to produce the project outcomes while meeting the budget and the delivery date.
7. During the project, the scope manager ensures changes to the scope are understood and that the department and the developer agree on their price and time impact. The department might query a lack of functionality that it thought was to be delivered. The scope manager then acts as a “referee”, stating whether the functionality in question was, or was not, in the specification and subsequent function point size. If it was included, then the developer has to deliver it. If it was not included then it can be sized and the department can decide whether it still wants the additional functionality, based on the known cost per function point. The same procedure applies if the developer claims that deliverable functionality was not included in the specification. This keeps both the department and the developer “honest”.
8. At the conclusion of the project the department makes payment to the developer on the basis of software delivered plus the agreed changes at the cost per function point that was quoted.

Has it been successful?

The Victorian State Government commissioned a study to review the success or otherwise of this new methodology. The following charts show the results of that study:



The cost overr ess than 10% whereas the industry norm was 84%.



Projects using the *southernSCOPE* method were found to be delivering at around one third the cost of typical industry projects.

According to the Standish Group, the six most common causes of project cost overruns are:

- Lack of user involvement
- Incomplete requirements
- Changing requirements
- Lack of executive support
- Developer incompetence
- Unrealistic expectations

It is believed that the reasons why the projects using *southernSCOPE* are successful is that this approach eliminates or reduces five of the six factors of the above causes - the only possible exception being the 'lack of executive support'.

The review also found that projects using this approach:

- Resulted in high customer satisfaction as it resulted in the delivery of functionality well focused on the primary business objectives
- Completed successfully with minimal or no overrun of time or budget
- Provided software value-for-money within the top 25% of industry best practice

Additional information

The Victorian State Government has developed extensive documentation and material for its *southern SCOPE* methodology and made it available in the public domain. Visit :

http://www.mmv.vic.gov.au/southernscope for additional information.

The *southernSCOPE* method is now receiving widespread attention on a global basis as a viable alternative to fixed price tendering by both government and business.

CASE STUDY

Finland – Ministry of Justice - Prison Information System

The problem

"When I reviewed this project for the first time, I thought that the sooner it could be cancelled the better", said JJ, one of the FiSMA's[8] scope managers.

It happened in 2001. JJ was working for SupCo Plc[9], one of the biggest independent software suppliers in Finland, operating in several business areas with several sites around the Scandinavia. Another department of the company had started a large tailored software project three years earlier. The objective was to develop a complex Prison Information System, (PSS), for an external customer, an agency of the Ministry of Justice. In 2001 the project seemed to be in serious trouble, almost "mission impossible". Both the supplier and customer representatives were very pessimistic about the future and concerned that the system would never be delivered, resulting in a complete loss of their investment.

In the end the project wasn't cancelled, phased development is continuing and now progressing very well. The planned completion date is early 2005. The final price will be four times greater than the original "fixed price". Because of huge overrun in schedule and budget, one would expect the customer to be furious and feeling cheated. But this is not the case. Both the customer and developer are now pretty content with the current status. They both report that PSS is progressing very well. All the people involved believe, that based on the current plan, the Prison Information System will now be delivered on time and on budget. So what happened to turn around such a disastrous situation? In this case study we will investigate the key factors that turned the project around.

This story was told by Ahti Lempiö and Martti Karjalainen from Ministry of Justice and SP and JJ from SupCo. It was recorded and edited by Senior Scope Manager Pekka Forselius from FiSMA.

Background of the project

The Finland Criminal Sanctions Agency had an old prison information system running on a Wang platform. It was developed by an external supplier company, which no longer existed. For several years the only maintenance and support available was from a single external consultant. Support for the Wang development environment was no longer available. The system was becoming a bigger and bigger risk as the end of the millennium approached. In addition, there had been a lot of changes in the law plus new business needs, especially related to the integration with other government databases and European Union systems. The users at Criminal Sanctions Agency had a list of required improvements. So, there was a real user need to replace the old system.

Personnel at the Criminal Sanctions Agency had limited knowledge of the various software systems available. In addition to the old prison service system, they were using a couple of small administrative applications developed in Lotus Notes. They liked those applications and thought that it would be beneficial to have the same simple architecture for the new PSS system. This became one of their technical requirements, which later proved to be a mistake. But their rationale was understandable.

[8] FiSMA, Finnish Software Measurement Association, a network of organisations improving their software development management systematically, through measurements, data collection and analysis.

[9] A fictitious name has been substituted for this Case Study

The requirements specification document for the new PSS system was developed by a consultant from a large, international consulting and development company. It later became apparent that this requirements specification was inadequate for fixed price development outsourcing.

In early 1998 the Criminal Sanctions Agency received tenders from five suppliers. (It is interesting to note that the company which had specified the requirements did not bid, despite being invited).

SupCo Plc was one of the five tenderers. This company had a lot of experience with banking, manufacturing and retail sectors. They were seeking to extend their client portfolio into the public sector. SupCo Plc had little experience of Lotus Notes development, but they had skills with many other tools so they considered this to be a very minor problem.

The Prison Administration Director remembers how surprised he was to see the two most convincing and professional tenders, so close to each other. These two were evaluated more thoroughly and finally the contract was negotiated with SupCo, which was then given six months to improve the requirements specification.

The contract was very detailed and formal, no doubt influenced by the customer being the Ministry of Justice. All kinds of penalties for project over-runs were included but nothing very specific or measurable about the system scope or quality requirements.

From this position the PSS project was launched in March 1998.

The Target System Structure and Project Organisation

The Prison Service System consists of two main subsystems and a large database. The subsystems are called Sanction Execution System and Prison Operation and Control System. They are both very large and used by several different types of user groups. The overall structure of the system is illustrated in figure 1.

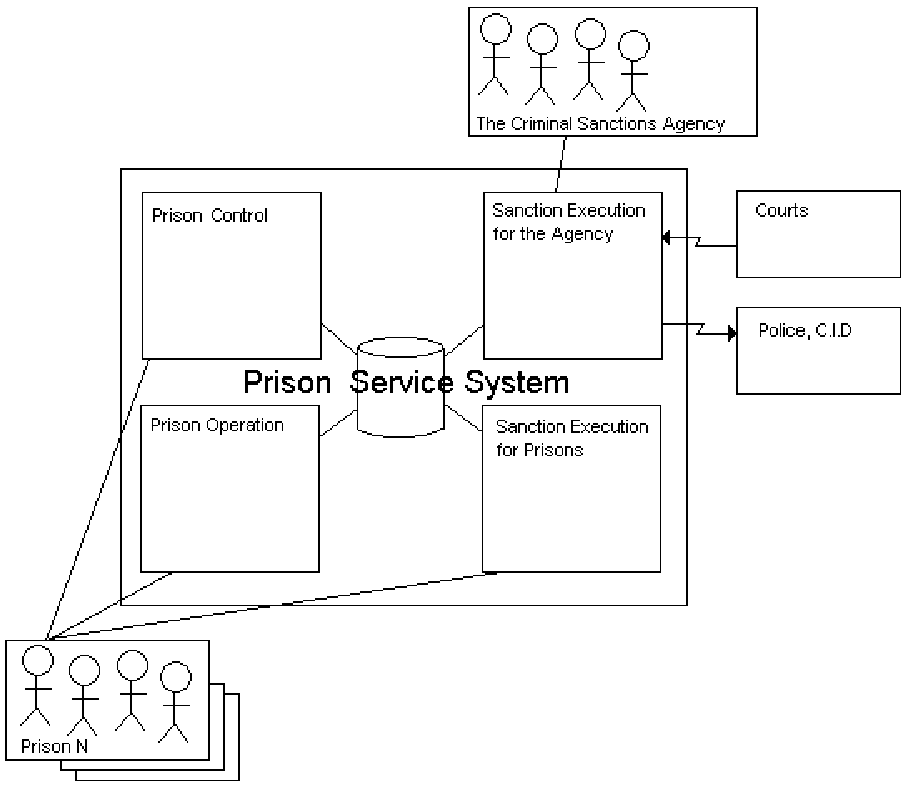

The Criminal Sanctions Agency
Prison Control
Sanction Execution for the Agency
Courts
Police, C.I.D
Prison Service System
Prison Operation
Sanction Execution for Prisons
Prison N

Figure 1. Original Architecture of the Prison Service System (PSS)

The project was organised in a normal way, having two separate project groups, one for the supplier and another for the customer representatives. The Steering Committee consisted of top management representatives from both parties. The original schedule was very tight and ambitious. The intent was to get everything implemented before year 2000. The original 1998 project structure is shown in figure 2.

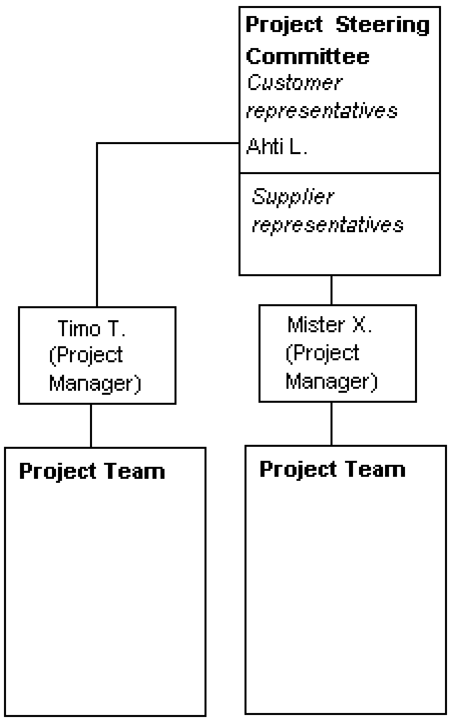


Figure 2. Organisation of the PSS Project in 1998

The project suffered serious changes to senior project personnel (1998-2001)

When a project manager realises, that he or she is perhaps out of their depth, they will very quickly look for a less risky position. This happened three times in PSS project during the first 24 months. Not just project managers, but also other "experts" from the supplier organisation disappeared one after another in that period.

JJ remembers that the software market was very buoyant in that time and it was very easy to find new jobs. He also recalled that most of the original team members were very young and impatient to improve their chances of promotion. When they realised that the project would take much longer than expected, they left.

A change of architecture and tools

The system was now in trouble. However, the top management of SupCo was very committed to the PSS project and didn't want to abandon it. They believed the technicians who told them that the Lotus Notes architecture would never work for this project. They had a number of discussions with several customer representatives. They negotiated, and having displayed a good understanding of the system and a firm commitment, the representatives of the Ministry decided to continue with them. These negotiations took two months, during which time there was no progress on the development.

The result of the negotiations was that the system architecture and development tools were changed. The PSS would now be a three-tier system developed in MS Visual Basic utilising a relational database. SupCo had a lot of developers with skills and experience in this architecture. Figure 3 illustrates the new structure of the Prison Service System.

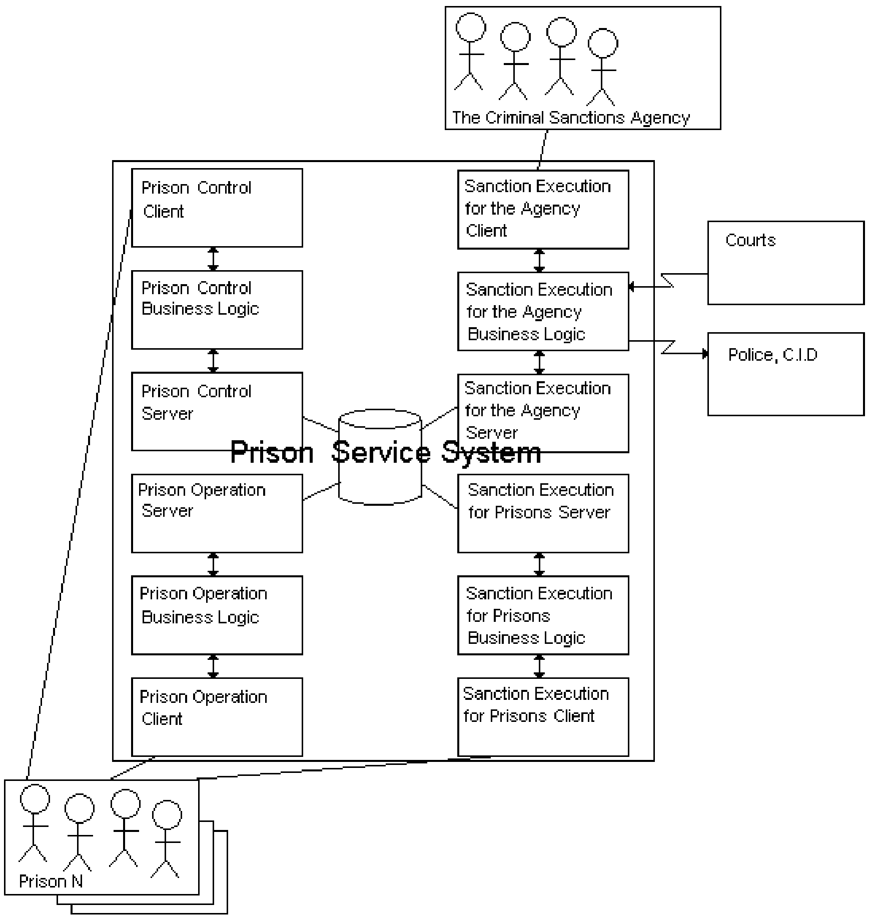

The Criminal Sanctions Agency
Prison Control Client
Prison Control Business Logic
Prison Control Server
Prison Operation Server
Prison Operation Business Logic
Prison Operation Client
Sanction Execution for the Agency Client
Sanction Execution for the Agency Business Logic
Sanction Execution for the Agency Server
Sanction Execution for Prisons Server
Sanction Execution for Prisons Business Logic
Sanction Execution for Prisons Client
Prison Service System
Courts
Police, C.I.D
Prison N

Figure 3. The New Architecture of the Prison Service System (PSS)

The consequences of changing the architecture were huge. There was only one independent application in the original solution. Of course it was much larger than any of twelve independent pieces of software needed for the new PSS system, but its functional size was markedly smaller than the total size of the three-tier system.

The difference resulted from all the interfaces between the peer components. On the other hand, this kind of structure improved flexibility and maintainability of the system.

In addition to the size changes, there were other impacts. The PSS project also needed new technical staff and a new schedule.

The project was further impacted by the need to upgrade the old Wang-environment, because of year 2000 problems. The old system had to be centralised, so that all the services which had been distributed to local servers were moved to the upgraded Wang-mainframe. The usability and efficiency of the Wang-system deteriorated. This upgrade project interrupted the main PSS project, because most of the customer representatives were involved with the upgrade.

The PPS development was started from the beginning again. The delivery library of the PSS system started to grow, but the problems were not over yet. The scope wasn't managed, consequently the requirements were creeping every day. All kinds of new features and automatic business rules that "should be there", according to the users, continually popped up. This meant returning to earlier phases to redesign. As a result, no completed functionality could be delivered, which was frustrating to both the developers and the customers.

More people left the project. Personnel replacement incurred lost time and money. This meant more and more excuses and explanations. The supplier top management was in serious trouble. It was at this stage that the current Director of Government e-business solutions, SP joined the Steering Committee. Getting control over the project wasn't easy. He remembers that the negotiations between the organisations were extremely serious, taking place at the highest executive level in mid 2001.

The turning point

A new project manager took responsibility for the project in 2001. A change of the project manager wasn't big news to the client, as this was the fourth change of project manager on the PSS project. But this change proved to be the crucial one. After a short introductory period the new project manager started to question the scope of the system. There were different, contradictory, documents, but the full list of all required functionality was missing. The project manager also recognised that the planned system was incredibly large.

In June 2001 the project manager had some rough functional size measures done on both the current and original specifications. He noticed quite big differences. He then proposed establishing the functional size of the system at detailed level. Initially this was not agreed to because the customer representatives didn't understand how this would help. They did not understand functional size measurement. They thought that it was something of use to the supplier only. They were even less interested when they were told that functional size measurement of a large system would take several working days.

The project manager didn't abandon his idea and continued educating the customer about its benefits. Finally, the client representatives agreed to functional size measurement. This change of position occurred because they had asked for help from the IT Office of the Ministry and were advised by Martti Karjalainen that functional size measurement would be useful.

The function point counting was started, using Experience Pro software and Experience 2.0 Function Point Analysis, (FPA), method[10]. Initially there were some problems due to a lack of experience in counting multi-component systems. There are some important differences between these functional size measurement methods, especially related to the algorithmic and manipulation user requirements. Pekka Forselius from the Finnish Software Measurement Association was asked to assess the measurement and give independent, external guidance if needed. The personnel of the I.T. Office were given a two day Estimating and Measurement Training Course. Since this training took place there have been no problems with the team members co-operation in size measurement[11]. The consultant then helped the supplier to find the best fitting delivery rate, (in hours per function point), using both the Experience Database and the ISBSG data disk, which are the two largest common project datasets in the world.

The project organisation had changed a lot since it started in 1998. The new, stronger organisation is illustrated in figure 4. As the figure shows, neither the customer nor the

[10] Experience 2.0 Function Point Analysis method is the most popular size measurement method in Finland. It was developed by FiSMA in the middle of 1990's. It is not a standard, but it is conformant to the ISO/IEC 14143-1 standard of Functional Size measurement.

[11] JJ and the new project manager CH participated the same Estimating and Measurement training already in 1997.

supplier scope manager, are members of the Steering Committee. They are simply advisory members, participating as needed.

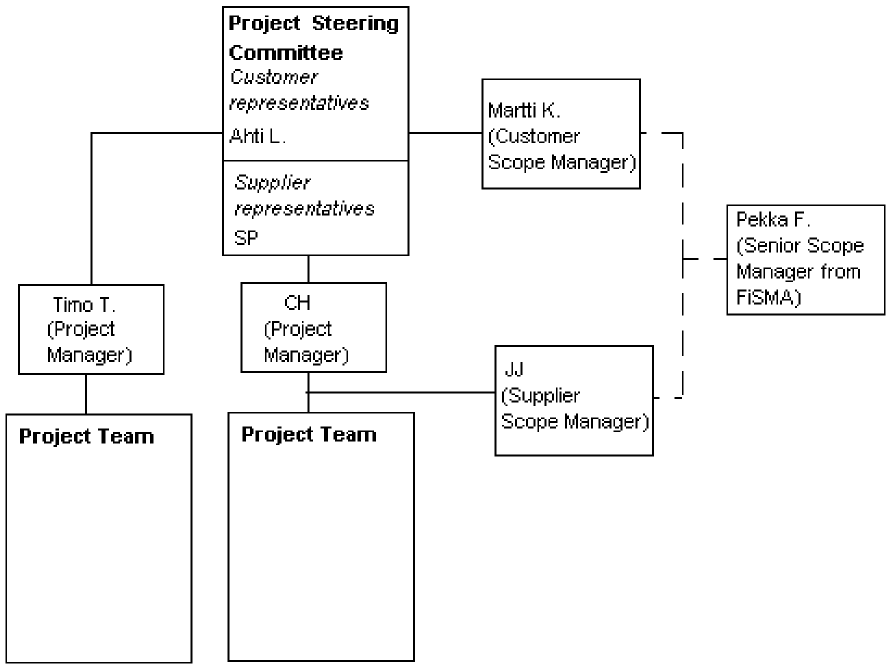

Project Steering Committee
Customer representatives
Ahti L.
Supplier representatives
SP
Martti K. (Customer Scope Manager)
Pekka F. (Senior Scope Manager from FiSMA)
Timo T. (Project Manager)
CH (Project Manager)
JJ (Supplier Scope Manager)
Project Team
Project Team

Figure 4. Organisation of the PSS Project in 2002

A period of progress

Following the first functional sizing all schedule and cost estimates given by the supplier project manager have been accurate and well explained. There have been some minor changes in both directions. The scope has changed only very slightly. Using functional size measurement at a detailed level was a real breakthrough.

As JJ said:

"The participants of the project found an agreed way to manage scope. All of them understand the consequences of scope changes now. In practice the project changed from fixed price model to a model based on a number of Euros per function point delivered."

SP believes that delivering Phase 2 on time and within budget regained the trust of the customer. The customer has been convinced of the skills and capabilities of the current project team, which has remained stable since the new project manager started. All the development is undertaken on-site where the project manager is resident. The team has proven its capability so well, that SupCo slightly lowered the price per function point after the second phase, which was a positive indication of the growing partnership. SP hopes that good results and improving productivity will keep the motivation of the development team as high as it is now.

The last phases of the project have been planned. The progress is steady and under control. All the interviewees are convinced that the project is back on track. Now, in the end of 2003, PSS is an exceptionally well managed project.

Lessons learned

As the only person who has been involved from the beginning to the end, Ahti has probably seen the most. In his opinion, it would not have made any difference to select another supplier in 1998, or to change supplier in 1999. As he understands now, all the tenders they received in

1998 were very loose, based more on expectations and guessing than on facts. He thinks that SupCo has tried very hard in difficult circumstances. Some other suppliers would probably have given up and walked away. Regarding the decision to finally apply FiSMA estimating and measurement methods he believes:

“Without function point counting and scope management support from the IT Office of the Ministry of Justice, this project could never have succeeded.”

He thinks that the ability and capability of the IT Office to support professional scope management has improved remarkably. The Ministry of Justice is also reorganising its structure. The role of IT Office is changing, so that it will be easier for the agencies to ask for help in the future. As a result, this kind of large system development project will not struggle for several years before getting back on track. Ahti still believes that in addition to their own measurement knowledge and skills, external, independent scope managers will be needed to balance and stabilise large projects.

The only big disappointment to Ahti was, that in spite of very strict and demanding project contract, with very clear penalties specified, the expected fixed price wasn’t fixed at all. Fixed price contracts do not work in these circumstances. This kind of combination of an incomplete requirements specification, scope creep and wrong technical architecture will never be managed by contracts or penalties because it is impossible to establish who is responsible for the failures. He suggested that the use of an external scope manager from the beginning of the project would probably have helped avoid most of the problems.

SP has been involved almost all the time since the system architecture re-design. He has seen both the bad and good times of the PSS project. He confirms the opinions of Ahti about the importance of functional size measurement in this case. He emphasizes the importance of functional sizing at a detailed level. The Prison Service System includes an exceptional number of algorithmic rules, most of which were not specified in the original requirements specification document. This kind of omission makes fixed price contracting pointless. Another unexpected characteristic of the PSS system was the complexity of the user interface. Rough function point counting doesn’t catch this kind of “extra functionality” and can result in a large error in the total size. Detailed functional sizing is essential to cover this kind of project characteristic.

Summary of the lessons learned

- ❖ If possible, ensure that the requirements specifications are complete and contain all the required functionality
- ❖ Have in-house consulting support available to the project customer to provide advice if the project runs into trouble
- ❖ Where requirements are difficult to define in total prior to the start of a project, or there is the possibility that requirements will be added, do not attempt to work on a fixed price quotation basis
- ❖ Use an external scope manager from the beginning of the project to provide balance and objectivity
- ❖ Be very wary of rough functional size estimates, use these only as a very broad, initial indication of the possible size of the project
- ❖ Be aware of unusual functional complexities, in this case a high number of algorithmic rules or a complex user interface
- ❖ Spend the time and money needed to establish an accurate functional size from detailed specifications. (This is a good opportunity to check the completeness of the

specifications, if a detailed functional size measurement cannot be completed from the specifications provided then the specifications need more work).

The current situation

Martti is very happy with the current status of PSS project. He has noticed increasing interest in the scope manager concept and functional size measurement among the staff of the Ministry of Justice. He believes that this kind of approach will become more popular after its success on this project.

Another, very interesting observation made by Martti was the change in attitude of the customer project manager. Initially he didn't get involved in the functional sizing at all, because he thought that it was not his business. Another colleague stood in for him. When the second sizing was started, he tried to avoid it again, but Martti and the colleague from the first counting demanded that he participate. Finally he responded, fortunately. He noticed that functional size measurement wasn't difficult at all. It involved discussion about issues which he knew very well and could actually contribute to better than the others. Martti commented that the project manager was now the first person proposing to start the next use of sizing.

One issue that has not yet been discussed in the PSS project is maintenance. Martti is of the opinion that it will be possible to use function points as the basis of a maintenance contract. It will be interesting to see how this develops.

Functional sizing accuracy review

JJ produced the next review of the functional size measurements of the PSS project:

Year	Measurement	Input	Result
1998 May	Rough Function Point count done by the supplier expert	Original requirements specification document	3,000 fp
1999 Feb.	Rough Function Point count done by the supplier expert	Requirements specified by the selected supplier	5,000 fp
2001 Nov.	Rough Function Point approximation done by the new project manager	Requirements specified after changing the architecture	12,000 fp
2002 Mar.	Detailed Function Point Analysis with the customer by the new project manager, supported by all three scope managers	Requirements specified after changing the architecture and after 3 years scope creep	20,500 fp
2003 Oct.	Detailed Function Point Analysis with the customer	Current requirements specification document	20,500 fp

The change of the project size from the first rough count to the first detailed count is massive. Hopefully this acts as a warning. You must not put too much trust in rough size estimates. Another important point is that just counting the number of function points is not sufficient. The estimate may be quite good, if the scope doesn't creep, but a thorough functional size measurement, providing the complete list of functional user requirements, establishes the best known basis for a successful relationship between the customer and supplier. This is a completely new level of software development project management compared to the common, current practice.

Finally, both the supplier and the customer of the Prison Service System development project know the final total price of the project. As explained, the project is almost four times bigger than the original "guesstimate", but what does this mean? Is it expensive? No, it's not. The final Euro price per function point is very competitive compared to other published examples that use a price per function point approach, (for example southernSCOPE projects).

CASE STUDY

Victorian State Government, Australia – Department of Human Services - Telephone Information and Reception Service

The problem

The Victorian State Government has a policy to increase the range of services and information available electronically to the citizens of Victoria. In 2002 the Department of Human Services, (DHS), undertook to redevelop the reception processes of its regional offices. This redevelopment project was called *Telephone and Information Reception Ser*vice, (TIRS). This system was a pilot implementation in the Southern Metropolitan Region of Melbourne.

The system

The system included the following primary components, which were based on existing Lotus Notes infrastructure The following features were included on a single screen:

- "While You Were Out" visitor and telephone message recording and time-based message escalation process
- Integrated view through to the corporate telephone listing
- Client to allocated Worker look-up
- Information kit and brochure mail/email facilities
- Online local services information and online system help
- Desktop quick links to Internet and Intranet resources
- Desktop search facility.

Where relevant, existing Lotus Notes features were customised/enhanced to minimise keystrokes and to provide for faster and more efficient message forwarding. Customised Lotus Notes features were also incorporated into TIRS.

The approach

At the outset DHS proposed to use the TIRS development to trial the Victorian Government's southernSCOPE methodology for acquiring software. The expected benefits of using southernSCOPE were:

- Early estimates of project size and cost to assist in tender evaluation
- A value for money software solution
- A mechanism to control the project budget by controlling project scope
- An independent audit of project specifications and design to ensure system functionality met the business objectives
- Effective change management procedures
- A 'hassle free' negotiated development

- A quality software product

The project

Groupware Consulting Pty Ltd was engaged as the software developer and Total Metrics Pty Ltd performed the role of the independent Scope Manager. This was the first southernSCOPE development for both the client and supplier organisations, consequently the project closely followed the methodology under the guidance of the Scope Manager.

The results

The project was completed on time and on budget with all parties agreeing the development was 'hassle free'. All the expected benefits were realised.

The learnings

Factors that were recognised as contributing to the success of the project include:

- The relatively small size of the development team and the project (239 adjusted function points).
- The additional effort expended in the analysis phase to ensure that the Functional Requirements Specification,(FRS), was complete, accurate and consistent.
- The build phase was delayed until all parties approved the FRS document and a detailed baseline function point count was completed.
- Commitment by the client to minimise functional change in order to ensure a timely delivery.
- The technical skills of the development team.
- The technical and procedural discipline maintained through the support of the Scope Manager.
- The commitment of all parties to the southernSCOPE trial.
- The clarity of the processes, ease in tendering and bidding for the client and the vendors.

Rather than rushing into development, the DHS, working with developer Groupware Consulting and Total Metrics as scope managers, spent some 55 per cent of the project's time in the analysis and design stages and just 34 per cent of its time in actual development. DHS project manager Chris Carter says spending so much time on design paid off, (industry figures suggest an average of 35 per cent for analysis and design).

CASE STUDY

Victorian State Government, Australia - Student Management System

The problem

Three previous attempts at defining requirements for a whole of Technical and Further Education – Higher Education Centres, (TAFE), student management system had resulted in three disparate, voluminous and sometimes unrelated sets of documentation in varying formats that related to the perceived requirements of both TAFE central and distributed management needs. In two of these cases, software development had been commenced then abandoned when budgets were exhausted and delivery was obviously not imminent. In some instances individual TAFE units had proceeded alone to develop partial solutions to meet their own needs.

The Office of Training and Further Education, (OTFE), was determined to address this problem and implement an overall Student Management System. However, given the disastrous history of previous attempts, an approach was needed to determine the scope of the final application and the practicality of achieving an implementation within a pre-determined budget and a fixed time frame.

Scope Manager role

A decision was made to employ a Scope Manager. CHARISMATEK Software Metrics was appointed as the Scope Manager. This initially involved collecting and outlining functional requirements in a manner that would facilitate analysis, evaluation, prioritisation and cost forecasting. The ongoing role of Scope Manager to oversee the practical implementation of a development process based on a cost per function point, followed the initial development of the conceptual model.

Project attributes

- **Requirements** were in a number of disparate places, the two existing applications and three additional sets of Requirement Specifications.
- **Software Size** was unknown.
- **Project Schedule** was fixed at 24 months. This is a very difficult target for any substantial project and this project had proven difficult in the past.

The approach

In order to determine one common view of the software based on five different sources of requirements information, functional requirements were collected and defined within a functional model by the Scope Manager. This model, and associated search and query facilities, were used to identify common and/or duplicate functionality and to enable first cut priorities to be linked to broad sections or individual functions. The end user of each function (or group) was also identified.

An analyst, highly experienced in the field of Student Management Systems, then reviewed the model with the Scope Manager. Around 9,000 function points were documented. The older functionality could easily be identified and marked as not required for the future. Large amounts of duplicate functionality across multiple administration systems could also be identified. Ultimately, around 6,500 function points of functionality were determined to be desirable in the new Student Management System. Of these, around 3,000 function points were mandatory; 1,500 were highly desirable and another 2,000 function points were desirable.

At this point, whilst we had some idea of the functionality needed and hence probable size and cost implications, the process of fully identifying and documenting requirements still had to be undertaken as part of the project.

Tender Responses were requested on the basis of a dollar per function point price for a solution in the order of 2,500-4,000 function points. By including the phase of Requirements Definition, the phase that determines the final size of the solution, as well as the build and delivery phases within the one tender, much time (and money) was saved.

The supplier selected offered a package based option. The package was based on source code from the selected supplier who could thus provide native changes and additions using their own development environment. On this basis, they were able to provide an extremely competitive price per function point. The client was happy with the look and feel of the software. It was the responsibility of the supplier to match requirements.

However, there were some constraints on the supplier.

Rules of engagement

- ❖ All functionality defined within the functional model was to be addressed and prioritised, with the client having the right to select any set of functionality deemed most appropriate.
- ❖ The client would only pay for functionality that was defined as a requirement, (and hence signed off, etc.).
- ❖ Documentation of functionality was adjusted slightly to facilitate sizing processes from requirements documentation collected as part of development process.
- ❖ The Scope Manager was the sole arbiter. (Note: this does require a substantial degree of trust by both parties, so it is highly recommended that any Scope Manager be pre-agreed by both parties).

The tactics

The supplier was in direct contact with the users and collected requirements. This gave them the opportunity, which they took advantage of, to promote the functionality that they could implement ‘as is’ and thus achieve ‘easy’ function points and payment milestones. However, the natural honesty and professionalism of the staff made sure this did not go too far and some sensible review by the project management team ensured this was not an issue.

The client, on the other hand, requested enhancements to the functionality offered by the package. This meant that the supplier was being forced to change more functionality than originally anticipated. Some negotiation was required.

The results

Despite the normal project difficulties for an implementation of this size, the project went remarkably well. After the Phase One implementation was successfully completed within time and budget, (with an acceptable profit for the supplier), Phase Two followed using the same process.

The client received a very good Student Management System which was completed and implemented within the specified time and budget. The supplier made an acceptable profit.

However, the supplier refused to proceed beyond the original upper limit of size, (4,000 function points), on the basis of the original contract. The proportion of new and changed functionality was much higher for enhancement work than for the original implementation, (where around 60% was unchanged from the package). Given this different mix of work, a different rate per function point was required.

What have we learnt?

A lot was learnt from this project.

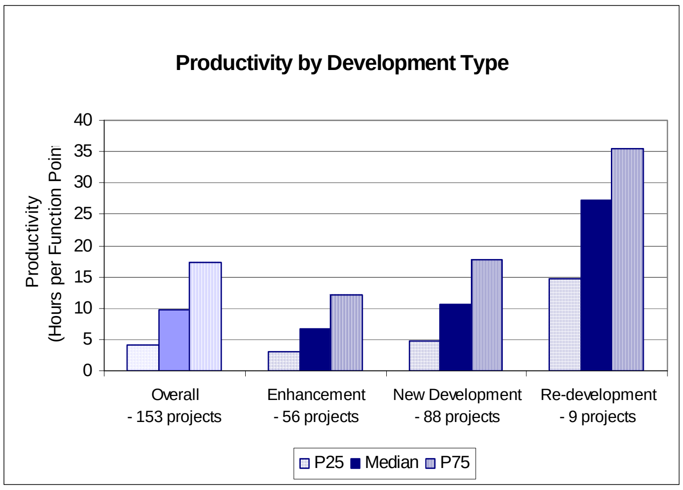
The critical points were:

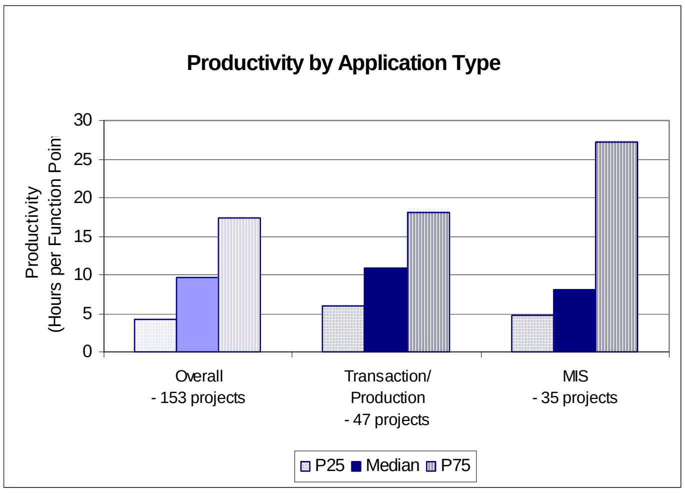
- The use of a price per function point approach is a practical and effective way to integrate full life cycle development charging mechanisms to the ultimate benefit of all parties. It has the potential to enable one group to manage a project from requirements to implementation whilst ensuring a reasonable and consistent charging and control mechanism is in place.
- The need for multiple productivity price rate options for suppliers of different tasks/technologies. Whilst the price per function point approach provided an excellent basis for client and supplier to work together for some time, the rigid nature of using just the one rate for all work ultimately damaged the relationship.
- The potential for the quantified functional model to be used as a basis for clarifying functional requirements across a myriad and complex web of needs. This means of representing and communication software capabilities enabled a wide range of people to interact effectively, using the same simple representations.
- The ability of the graphical functional model, and its associated function point size, to operate as an effective means of communicating priorities and cost implications when defining software needs.

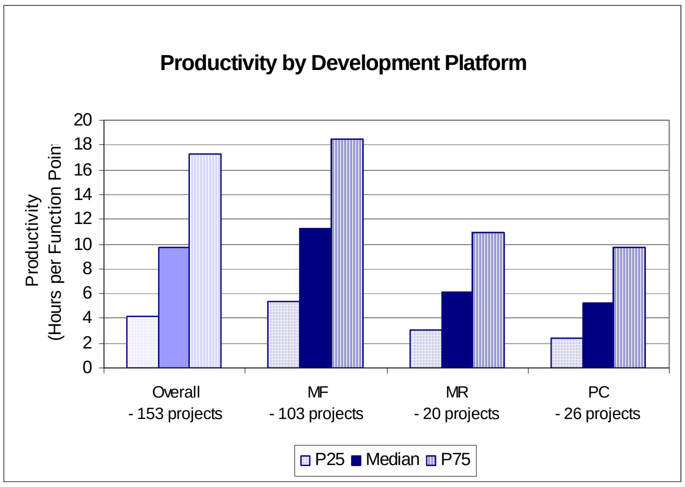
Appendix A – Tables of Government Project Performance

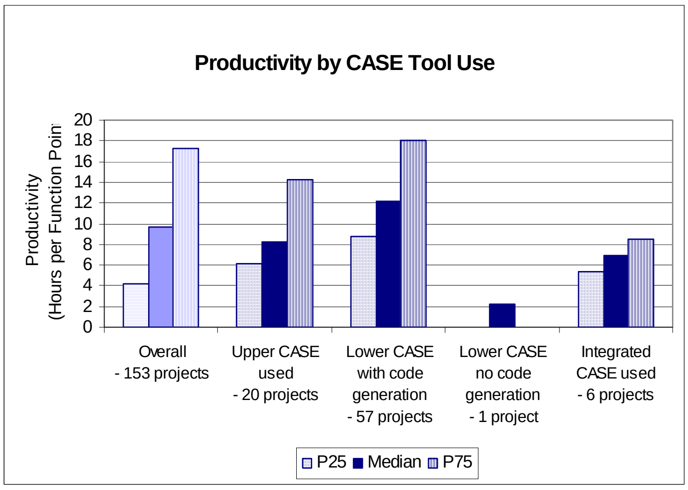
Overall productivity for Government projects, shown in hours per function point delivered:

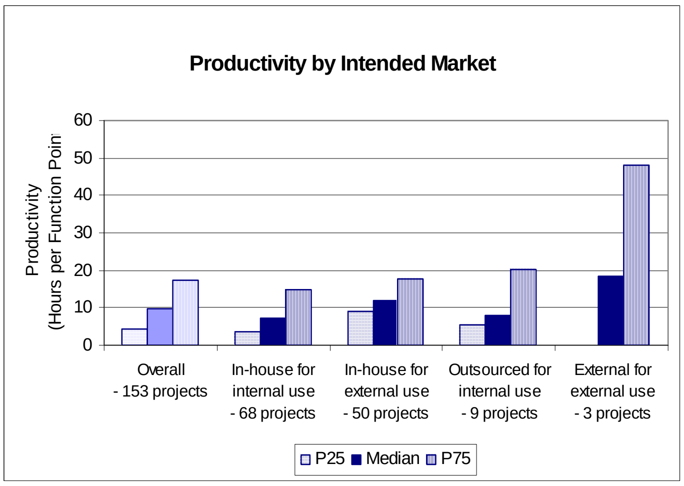
		n	P25	Median	P75	Mean	Stdev
Government projects							
All		153	4.2	9.7	17.3	13.7	14.6
Productivity by development type:							
Enhancement		56	3.1	6.6	12.1	9.9	9.9
New Development		88	4.8	10.5	17.7	15.1	16.5
Re-development		9	14.8	27.2	35.4	23.8	13.2
Productivity by intended market:							
In-house for internal unit		68	3.7	7.1	14.9	14.0	17.6
In-house for external unit		50	9.1	12.0	17.7	15.0	10.3
Outsourced for internal unit		9	5.3	8.1	20.3	13.3	12.3
External for external unit		3		18.3	48.0	25.3	20.1
Productivity by application type:							
Transaction/Production		47	5.9	10.9	18.1	16.8	18.6
MIS		35	4.8	8.0	27.2	11.5	11.0
Productivity by language type:							
3GL		72	9.1	13.3	24.8	18.3	16.1
4GL		75	2.7	5.2	10.5	8.7	9.7
ApG		4	2.5	17.8	72.0	27.5	30.5
Productivity by development platform:							
MF		103	5.4	11.3	18.5	16.1	16.0
MR		20	3.1	6.1	10.9	8.4	8.6
PC		26	2.4	5.3	9.7	8.7	10.4
Productivity by primary language:							
COBOL		61	9.1	13.4	24.8	18.6	16.4
NATURAL		26	2.4	4.1	10.1	6.3	5.6
Other 4GL		15	4.1	12.1	35.4	18.0	16.3
Productivity by CASE tool:							
Upper CASE Used		20	6.1	8.2	14.2	10.8	10.8
Lower CASE with code generation		57	8.8	12.1	18.1	17.8	17.6
Lower CASE no code generation		1		2.2		2.2	0.0
Integrated CASE Used		6	5.3	6.9	8.5	7.1	2.5
Productivity by development technique:							
Data Modeling		85	5.9	11.1	18.0	16.4	16.5
Joint Application Development		29	4.5	9.7	20.3	18.0	21.8
Multifunctional Teams		23	2.4	6.5	20.3	15.8	22.5
Process Modeling		77	5.8	10.6	17.1	15.3	16.3
Prototyping		20	4.4	8.0	16.7	14.8	21.8

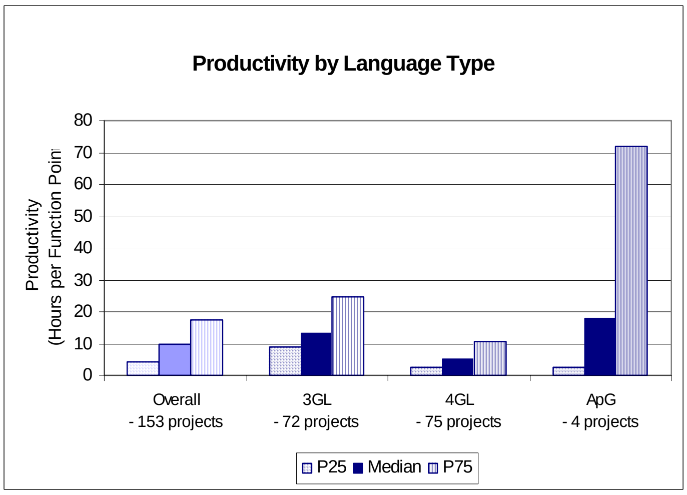

Productivity by Development Type
Productivity (Hours per Function Point)
40
35
30
25
20
15
10
5
0
Overall - 153 projects
Enhancement - 56 projects
New Development - 88 projects
Re-development - 9 projects
P25
Median
P75

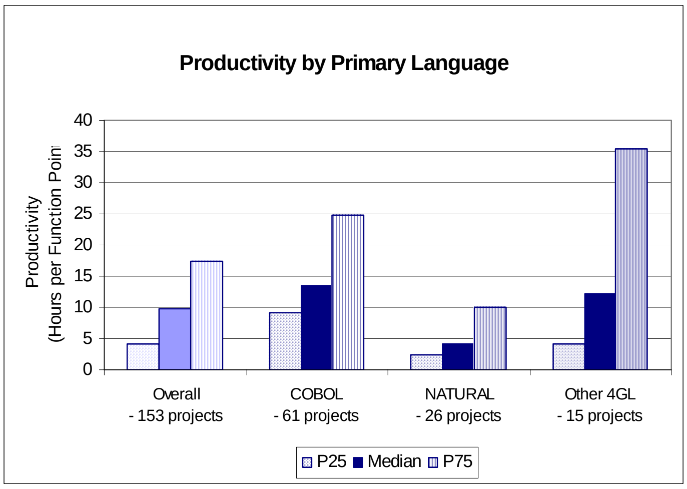

Productivity by Application Type
Productivity (Hours per Function Point)
30
25
20
15
10
5
0
Overall - 153 projects
Transaction/ Production - 47 projects
MIS - 35 projects
P25
Median
P75

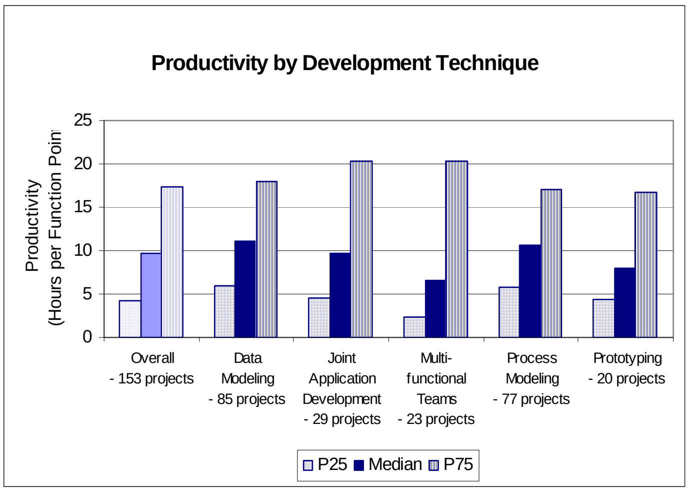

Productivity by Development Platform
Productivity (Hours per Function Poin
20
18
16
14
12
10
8
6
4
2
0
Overall - 153 projects
MF - 103 projects
MR - 20 projects
PC - 26 projects
P25
Median
P75


Productivity by CASE Tool Use
Productivity (Hours per Function Poin
20
18
16
14
12
10
8
6
4
2
0
Overall - 153 projects
Upper CASE used - 20 projects
Lower CASE with code generation - 57 projects
Lower CASE no code generation - 1 project
Integrated CASE used - 6 projects
P25
Median
P75


Productivity by Intended Market
Productivity (Hours per Function Poin
60
50
40
30
20
10
0
Overall - 153 projects
In-house for internal use - 68 projects
In-house for external use - 50 projects
Outsourced for internal use - 9 projects
External for external use - 3 projects
P25
Median
P75


Productivity by Language Type
Productivity (Hours per Function Poin
80
70
60
50
40
30
20
10
0
Overall - 153 projects
3GL - 72 projects
4GL - 75 projects
ApG - 4 projects
P25
Median
P75


Productivity by Primary Language
Productivity (Hours per Function Point)
40
35
30
25
20
15
10
5
0
Overall - 153 projects
COBOL - 61 projects
NATURAL - 26 projects
Other 4GL - 15 projects
P25
Median
P75


Productivity by Development Technique
Productivity (Hours per Function Point)
25
20
15
10
5
0
Overall - 153 projects
Data Modeling - 85 projects
Joint Application Development - 29 projects
Multi-functional Teams - 23 projects
Process Modeling - 77 projects
Prototyping - 20 projects
P25
Median
P75

Appendix B – Notes on Development Platform & Development Process

Development platform and development process

In this report you will find a number of references to the project attribute: "Development Platform". To date, "development platform" has been the best indicator of the environment in which the project is being developed. So the term more correctly refers to the whole environment/process, not specifically the hardware platform.

Development platform actually indicates development process and environment

There are two likely reasons for the major differences in productivity between PC, mid-range and mainframe development projects. These are:

- ❖ Development process, such as how the software was specified, designed, tested and documented;
- ❖ Business environment, such as number of business stakeholders and numbers of users.

The ISBSG performed a detailed analysis of the differences between the PC, mid-range, and mainframe projects, which was published in *Worldwide Software Development, The Benchmark, Release 5*. This analysis showed that, as expected, mainframe projects had more business units involved and supported a larger numbers of concurrent users. Such factors would result in worse, (higher), hour per function point productivity values because of the additional effort required to communicate with and obtain input from, a larger number of people.

The ISBSG's analysis revealed that mainframe projects make more frequent use of methodologies. The methodologies used on mainframe projects are likely to be purchased, but possibly then applied with some customisation. In contrast, PC projects make infrequent use of methodologies and the methodologies used on PC projects are likely to be written in-house.

Purchased methodologies are always comprehensive and detailed, projects that follow them tend to produce a wide range of documents, such as specifications, designs, plans, change and issue lists and test cases. In contrast, in-house methodologies tend to focus only on key parts of a software project's lifecycle. Projects that follow in-house methodologies, or no methodology at all, tend to produce fewer documents. A software project that produces fewer documents is likely to have a better, (lower), hour per function point value than a project that produces many documents.

Of course, software projects produce documents in order to communicate with multiple business units and to avoid the cost of rework resulting from poor specification, design and planning. The ISBSG project database does not yet have sufficient information to identify the appropriate balance between the effort on specifications, design, plans etc. and rework to correct defects.

Considering the development platform process/environment when using the ISBSG data

When using the ISBSG data on productivity, (Project Delivery Rate), for benchmarking or estimating, you need to consider the development process and client/user factors of your projects. Most importantly, you need to examine the following issues:

- ❖ What process or methodology do your projects follow? Is their process similar to the purchased methodologies with multiple documents covering planning, specification, design, and test, activities? Or do they not follow a methodology and thus probably not produce many documents? Or are they somewhere in between?

- What effort is required to communicate with client/customer business units and stakeholders? Are there many stakeholders and end-users involved, or just one or two?

When considering these factors, you need to identify whether your projects are similar to the ‘typical’ PC projects, or the ‘typical’ mainframe projects, or somewhere in between.

Appendix C – ISBSG Products

A series of products have been developed from the ISBSG Repository to help I.T. practitioners and business better manage software projects. The following table provides a guide the ISBSG products, their audience, use and the benefits that they can provide. All products are available from the ISBSG web site, (www.isbsg.org), or from the ISBSG member organisations in each country.

Current Products & Services	**Type**	**Purpose**	**Customer Benefit**
Estimation Toolkit	Book & CD	Educate I.T. practitioners about what is involved in estimation, the approaches available and the data available.	1. Education in macro estimation 2. Practice in estimation 3. Reference during estimation process 4. Estimation tools
Benchmark R8 (There is no Benchmark R7)	Book	Provide easy to understand analyses of the Repository, to assist I.T. practitioners in their work. Special analyses: Duration, Package customization, Quality, COSMIC projects	Knowledge to assist in project planning, decision making and control.
Benchmark R6	Book	Provide easy to understand analyses of the Repository, to assist I.T. practitioners in their work. Special analyses: Productivity, Effort, Team size, Cost.	Knowledge to assist in project planning, decision making and control.
The Software Metrics Compendium	Book	Provide updated statistical analysis of the ISBSG data using the latest project data.	A metrics practitioners', consultants', Project Managers' and researchers' "bible".
Estimating, Benchmarking & Research Suite (incorporating the data disk) A CD with Repository data and tools for estimation & analysis.	CD	Provide standardised data to allow I.T. practitioners and researchers to do benchmarking, estimating and research.	1. Ability to benchmark 2. Ability to estimate against history 3. Ability to research areas of interest 4. Ability to create simple "ball park" estimates
Organisational Benchmarking Service.	Service	Provide reliable data that can be used for organisation to organisation comparisons.	The ability to gauge a company's current practices compared to similar organisations. The ability to set goals based on perceived "best practice".
Sector Reports	Report	Provide information of the current state and best practices in software management in specific industry sectors.	An opportunity to better understand factors that influence software development management in specific industry sectors and apply best practice.
Data Subscription Service	Service	An annual subscription service that allows the licensee to offer access to the ISBSG data to a nominated number of users and incorporate the ISBSG data into its own Benchmarking database if applicable. Data updates every six months.	Flexibility to have the ISBSG data easily accessed and used by corporate staff world wide. Regular updates provide current data instead of having to wait for a new CD.