Design, Development and Deployment Considerations when Applying Native XML Database Technology to the Programme Management Function of an SME

Samuel Sambasivam
Azusa Pacific University
Azusa, CA, USA

ssambasivam@apu.edu

Paul Storey
Fujitsu Telecommunications Europe, UK

ps@spiess.co.uk

Abstract

This paper explores the design and development considerations inherent in the production and deployment of a web-based programme management coordination tool (PMCT) that has been built on a native XML database (NXD). In selecting a small to medium sized enterprise (SME) as a target, specific problems had to be addressed. These practical issues are highlighted along with assessments and analysis of the suitability of the NXD chosen for the function. The paper offers suggestions relating to the cause and effect of underlying trends currently affecting SME's within the telecommunications sector, and it shows how the management tools and the technologies that they might employ to derive benefit, need careful alignment with the purpose to be effective.

Keywords: Cocoon, Database, eXist, Internet, Management, NXD, XML

Background of the Problem

Readers who have worked for multiple software development organisations are likely to have observed that few use exactly the same mechanism for communicating or controlling information flows between their constituent parts; as such the coherency and effectiveness of the processes as a whole can vary. In some cases these variations can be stark and differences even occur between teams in the same organisation. Unfortunately the precise reasons are beyond the scope of this paper, but in general such fundamental differences as the maturity, size, cultural make-up, and geographical location of the organisation all influence or accentuate the differences. At the nub, is the human factor, where individuals attach varying degrees of usefulness to the processes. Unlike more mature industries such as manufacturing that have managed to converge and streamline many practices, the software industry with its spread across all the other industry verticals, gets pulled in all manner of directions. This is probably because of the different pressures each sector applies and the huge variety of platforms each sector uses to develop their software on (or for). Due to this wide range of variables it is perhaps quite understandable that anomalies exist and persist in the software development arena. However, it would be naive to think that there is one single development process or management mechanism, such as the programme management coor-

dination tool (PMCT) considered here, that would solve the problems.

There are already many established and accredited methods for the management of workflows and business process engineering that would have an impact on the shape of a PMCT, and essentially this goes to the heart of what the investigation was addressing. Using different business processes, widely dispersed and/or ad-hoc tools in any management function immediately opens up the possibility of project failure as information gets lost in translation, or, essential activities disappear down the gaps between. Where materials or products are the end result, the absence of an element can usually be traced by auditing and inventory controls. By the same token the management of software needs to be controlled in the same way, if not tighter in many respects as the potential for a software component to be badly specified or to be implemented incorrectly is huge.

Total Quality Management (TQM) systems and software process improvement (SPI) frameworks such as TickIT (TickIT, 2000), SPICE (Software Process Improvement and Capability Determination) (Software Quality Institute, 1999), CMM (Capability Maturity Model) (Software Engineering Institute, n.d.), etc. are a few of the business processes that have allowed significant strides to be taken by many organisations in pursuit of quality assurance and all of these are already a widely recognised benchmarks for development activities. These and other accredited methods provide adopters with a sound business process or framework in which to develop their own approach towards meeting best practice so individual components do not weaken the quality chain.

Taking into consideration of the above mentioned, the issue of organisational maturity, the flexibility in the interpretation of such a process and often the pressure to short-circuit some step or other to meet a particular software delivery deadline, usually allows enough scope for the framework to be bent out of shape and become ineffective, with project developments lapsing back into a failing spiral. In the authors experience resistance to adoption and where accepted practices exist, such deviations occur frequently. These can and sometimes do lead to the processes only being followed in spirit and not adhered too as they should to be a real benefit. Recognition of this fact is not new, as Humphrey (1998) stated that “a careful examination of failed projects shows that they often fail for non-technical reasons. The most common problems concern poor scheduling and planning or uncontrolled requirements. Poorly run projects also often lose control of changes or fail to use even rudimentary quality practices”. Intrinsically linked to these failures due to the ‘scheduling and planning’ problems suggested, is the activity of gaining control over these using simple information management techniques. In capturing information elements in XML the PMCT attempts to address that tenet.

History

While the software development activity in an organisation can be kept under one roof and the size of the said organisation is sufficiently large to cope with the adoption and adherence to proven processes, the approaches discussed previously will undoubtedly be very attractive. Indeed, one of the major goals of the International Standards Organisation (ISO) is to promote “the use of generic quality management principles by organisations and enhancement of their compatibility” (International Standards Organisation [ISO], n.d.).

Notwithstanding, SME’s have historically always required very flexible approaches to structuring their business arrangements and often use the latitude expressed within a framework of processes to their maximum. This attitude has also been seen to bleed into much larger mobile telecommunications software development organisations and the approach is not one that any seasoned practitioner would normally expect to see. It is also contrary to what most onlookers might have anticipated given the variety of commercial challenges that face any technological leader. Speculation as to the underlying cause can probably attribute this to them having to behave in very much

the same ways as SME's, as the general down turn in the mobile telecommunications sector took hold and the financial pressures of developing third generation mobile technology filtered through to the responsible department. Consequently, what once might have been a well funded and well resourced department, in recent years OEM's have increasingly had to rely on small and very dynamic teams corralled within a special competence centers that have many more freedoms afforded than in the wider quality controlled production environment; in a word 'lean'. It is in these cases where the high degree of correlation with SME's can be observed and in those very environments that the author has gained experience. This exposure gave rise to questions like:

- What happens to the information management process if the team trying to develop software has to resort to using the short-circuit routes mentioned above or can't maintain the proper processes?
- What happens to the information management process if the same body needs to use offshore development staff to keep development costs to a minimum?
- What happens to the information management process if the organisation wants to migrate some of its developers to using Tele-working or home-working?
- How well do organisations cope with multi-site development because of differing specialisations in certain locations?
- How well do organisations handle developments that have part of the work done in parallel by an industry partner?

In the authors own development and management experience and in an increasing number of cases, the above questions seem to be being raised in organisations of all sizes. Hence the PMCT investigation was identified and undertaken in an attempt to understand how it might make management of information in such circumstances easier and still allow enough flexibility for each organisation to apply a suitable process.

Trends

Even without the earlier questions to challenge the way the accepted process described may (or may not) fit the changing landscape of software development enterprises, a detailed review of related literature provided evidence to suggesting that a tool to help manage information flows might be of some use for SME's below CMM level 3. Estimates vary in relation to how many software development organisations there are that have achieved what is considered as acceptable quality levels in their software production processes but CMM level 3 has been suggested by Vu (2001), "Overall, the capability maturity has increased. In 1987-1991, 80% were at the 'chaotic' Level 1. Results through 1999-2000 indicate this had reduced to around 45%, with 31% at the 'Repeatable' Level 2 and 16% at the 'Defined' Level 3" (Software Measurement Services, 2000). Given the variance in the data and the lack of current feedback on adoption rates, an in-fill estimate was created in an attempt to spot a trend. Taking the known starting values and the years in which data existed (marked 'R' and in bold in Table 1) the significant figures (Level 1 to 3) in the middle term (1996) were estimated (marked 'Est.') and then extrapolated ('Ex.') to create the plot (Figure 1).

CMM % Level	Year				
against Time	1991	1996	2001	2006	2011
Level 1	**80**	62	**45**	37	30
Level 2	**15**	23	**31**	36	40
Level 3	**3**	12	**16**	19	22
Level 4	1	2	5	5	5
Level 5	1	1	3	3	3
	R	Est.	R	Ex.	Ex.

R = Result, Est. = Estimated & Ex. = Extrapolated

Table 1 – Estimated CMM % Level of Attainment over Time

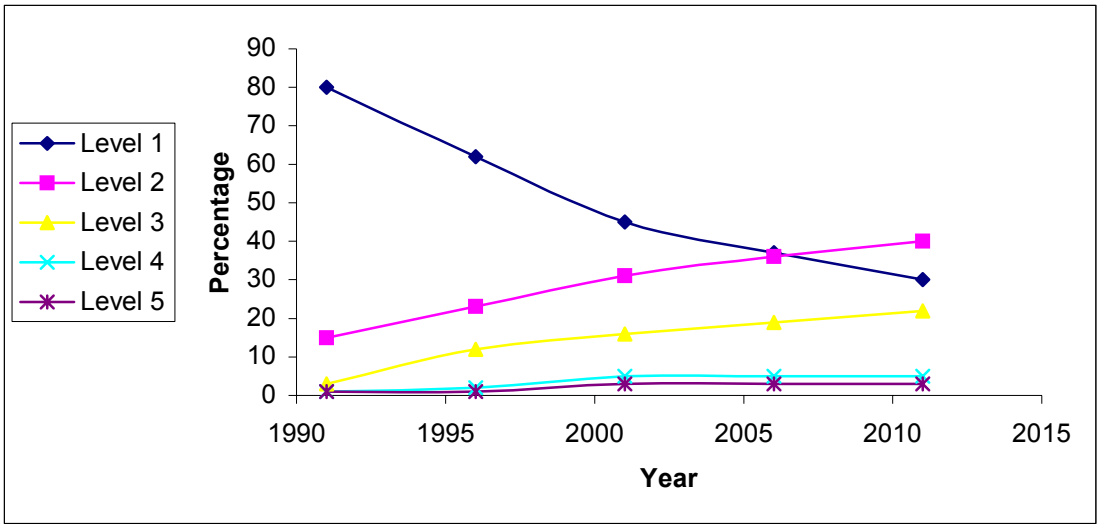

Figure 1 – Estimated CMM % Level of Attainment over Time

From the above known results values gathered in 1991 & 2000 it was observed that although there was a significant reduction in organisations labeled as level 1, the apportionment of this re-distribution amongst the higher levels was greatest at levels 2 and 3, with only small changes in level 4 and 5 attainment. Given the incompleteness of the information assumptions of this kind will be highly speculative. However, it was thought that even with a more aggressive adoption rate of TQM and SPI, the percentage that would remain in the lower brackets (<CMM Level 3) in the near future would still be 40-50% (as opposed to the 70% derived by extrapolation). Although big generalisations were being made, the above indicators and given the experience within some companies of CMM type 1, 2&3 it was thought that such a phenomenon would not be extraordinary. In embracing change and adopting structured processes for managing software developments various degrees of resistance are met with each organisation. This resistance acts as the damping effect and on that basis it was hoped that the PMCT would be adopted more readily in an attempt to tighten up on information control if the deployment and resistance was low. For

SME's the basic assurances needed would be low cost and low time overhead leading to some basic boundaries being set for the prototype:

- The requirement for the tool to be a lightweight offering (accredited methods could have been an inhibiting factor already).
- The requirement that deployment costs would be minimal and need limited effort to manage (dedicated equipment and personnel may not be available).
- A directional requirement that without adapting their ways users could derive benefit immediately without having to learn new systems or interfaces.

These factors then channel the research into the sphere of web-based deployment as it has been widely recognised for many years that this is one of the ways the above needs can be met. Following on from the research performed and the visible trend of businesses becoming more geographically spread through alliances, mergers, outsourcing, etc., this has the effect of channeling the thinking down the web-based route in the search for an appropriate mechanism on which to roll out the PMCT over such a dispersed user base. The underlying issues surrounding the need for companies to spread across the globe such as cost sharing, cost cutting, knowledge migration are all relevant in relation to the PMCT but are not discussed further as they were considered beyond the scope of practical implementation considerations. Instead the focus for the PMCT was maintained on the technical hurdles to be overcome in building the PMCT if it hadn't used the browser for the delivery.

Following slightly behind the trend in browser based deployments for mass market applications and services like banking, travel, government, general support functions, etc. are the browser based extensions to niche applications within individual industry sectors. Although there was no getting round the fact that in any event the content being deployed would vary, but the method of delivery to the user via the browser has matured to such a point as the users now expect it as a natural extension of other access routes. In this respect "web services are self-contained, modular business applications that have open, Internet-oriented, standards-based interfaces. They allow businesses to connect together applications either behind or outside of firewalls, independent of hardware, operating systems, or programming environments" (Hewlett Packard, n.d.), and hence resulted in the same chosen line for the PMCT.

After the above, the backend database application and choice of language became paramount. The trend towards the use of XML as the standard for data exchange is not new, but evidence of the use of NXD's was found to be relatively rare.

Potential Benefits

The main benefits of using "what was called TQM in the 1980's, Continuous Improvement in the early 90's, and Six Sigma Quality in the late 90's and 2000" (Sky Mark, 2000) are well known, and for ease of reference some are highlighted below (TickIt, 2000):

- Improved product quality and repeatability
- Increased process efficiencies
- Reductions in failure costs
- Increased employee satisfaction
- Reduced staff turnover

The PMCT was not conceived as a professional tool, neither was it conceived as an 'out of the box' or 'plug-and-play' ready business solution for SME's to be able to magic away all software quality issues. Instead it was initiated to understand how a tool might support two of the columns of TQM (Increased process efficiencies & increased employee satisfaction) and might allow a step to be taken towards SPI.

TQM is defined as "a strategy for improving business performance through the commitment and involvement of all employees to fully satisfying agreed customer requirements, at the optimum overall costs, through the continuous improvement of the products and services, business processes and people involved" (Institute of Management Services, n.d. cited by Jones et.al., 2003). The key components in the above statement with respect to the PMCT undertaking and its attempts to have an impact are the 'improving business performance' and 'the commitment and involvement of all employees'. In the first statement block the PMCT was attempting to allow managers to control the information flows more effectively by providing a structure to place them on. In the same way the PMCT attempted to provide a framework under which the whole organisation could gain access through their existing desktop browser facilities and goes a long way in meeting the goal of it being a tool that opens information to all and so achieving improvements in employee satisfaction.

Standardisation & the Programme Management Coordination Tool

It cannot have gone unnoticed by many industry observers that for many years now XML has been an area receiving increased exposure and use. In following this trend, it was decided that movement in this direction would offer an excellent opportunity to understand how a software development environment might be affected if critical information were documented in XML, stored in a database and delivered back through a web-based user interface.

The importance of using standards in relation to creating web services or publishing material cannot be overstated, simply because of the need to make any such offering visible and usable to the widest possible audience. This positive endorsement of the role of the standards bodies (e.g. World Wide Web Consortium (W3C)) is matched equally by the need to have emerging ideas, languages or frameworks feeding up from the bottom that get encompassed as they mature. In this way, draft standards offer the next best thing for developers that are at the leading edge and are pushing the direction without having to wait for full approval by a large committee or controlling body. Most of the implementation choices made therein used languages already approved by the W3C (XML, XHTML, CSS & XSLT), but never the less, outside of those the remainder were in draft form (XPath & XQuery) and under the W3C umbrella. This was the case for all except XUpdate, which although still has a reference authority (XML:DB) publishing and controlling the developing work, the XML:DB initiative "does not at this time consider itself a standards body" (XML:DB, n.d.).

In the same way as software languages have standards, the standards or benchmarks for TQM and SPI also need ratification. As mentioned, this convergence on recognised best practice is not only confined to the software development arena, but has also been around for many years prior in the wider body of business that recognises the benefit of structured processes and their application to workflows.

Existing Tools

During the dissertation study several comparable PMC type tools were found to be available against which conceptual comparisons were drawn. Many large scale process and/or combined

document management tools were uncovered (Dale, n.d.), but on inspection these types of applications were subtly different from the premise of the PMCT. Essentially the 'process' management tools found provided their users with the ability to model a workflow and/or monitor the activity of the business process. This might be construed as being very similar to what the PMCT attempted to offer, as it also used a structured approach to the capture of the organisational interfaces and allowed monitoring of the status of the interfaces across the enterprise. Where it was seen to differ, was in the target audience (SME vs. large OEM) and the user community it was trying to serve. Traditionally BPM tools are about process definition and monitoring from above, where the PMCT was viewed as a ground up enabling tool that every level of user could derive benefit from, not just those that define and police a high level process. One partially matching tool that aims at the SME was found to be Web Sphere Portal Express (IBM, n.d.). However, this tool did not appear to cover the areas the PMCT was aimed at even though the cost (£1,125.00) was at a price point most all SME's could meet.

Another closely matching product was found: Process Central 2003 (iGrafx, 2003). This was observed to be a very full and complete tool that covered and extend the capability defined for the PMCT. Costs for the above started at $7,000.00, and it was anticipated that that should be within reach of all but micro enterprises. In assessing the crossover between the commercial products available and what the PMCT was attempting to offer, obvious differences were present: the ERC tool was web based where the PC2003 was a custom client-server, and it was noted that neither offering relied on NXD technology.

Environment

As discussed earlier the authors experience indicated that there was no common shape or indeed universally accepted process found that could be applied to a team tasked with developing software in the telecommunications sector. The background for some of these differences has already been covered but in all cases a layered approach to management in any vertical is always found and is captured by Wright (n.d.) in his wide ranging observations. He describes the common thread running through industries such as aerospace, automotive, construction, defence, and telecommunications, as the reliance upon three layers of management at the level of project delivery. Overall there may be many more, but ongoing studies show management 'delayering' being whole heartedly embraced and in the telecommunications software development arena this was observed first hand. Many different factors exist to drive these trends, but objectives such as the need to have "participative design processes and structures, high performance work systems and or internal consulting teams" (Solutions Consulting, n.d.) are a few. These kinds of drivers are more readily needed in today's software collaborations as software components that execute as peer entities are now more likely to be developed by two distinct teams of engineers. These teams that historically would have come under one organisations remit and hence a single management structure above them, now rely increasingly on COTS components or draw on the services of outsource operations to provide the peer component. Immediately there are bridges to be built between the two enterprises and these have to be supported on team foundations that can be cemented together with frequent and detailed exchanges of information to ensure adhesion. If a software development organisation were to have a tall management structure with coordination points at only the higher levels, then this clearly would not foster an environment of cooperation.

With the target enterprise of the tool being the SME it was unlikely that the PMCT would require the ability to overlay a tall management structure so the ideal of a flattened or lightly layered organisation was kept in mind throughout. One consideration not picked up by Wright (n.d.) in his observations was the complex and free-form nature of software development activities. Application of management layers to appropriately segregate any large development is fine up to a point; however, even quite small software applications can quickly spread beyond what most members

of the participating team can comprehend. Adding management layers when software projects grow is unlikely to add any value. For this reason, in software developments the team concept provides the answer, allowing the project manager to take a higher level view and maintain control without having to know the internal component detail. Knowledge of the interfaces and the higher functions is then sufficient to allow the elements to be unified. As such the conceptual structure of three higher layer management functions and one low overhead team management function would seem to be sufficient for any software oriented SME.

Within many organisations new methods of software development are continually being introduced. In recent years the extreme programming (XP) methodology (Beck & Fowler, 2000) has started to replace the traditional waterfall approach, and SME's are well placed to gain from the benefits on offer. With the seemingly constant change in the baseline of the methods and processes extolled by the developer community, then there is also a move forward in the tools that teams use. Unless there is sufficient flexibility and willingness of management to implement these changes then gaps can open between collaborating teams as their streams of work progress at different speeds. In a worst case the goals that teams are working towards could also diverge as the application of a technique like XP asks different questions of the test strategy and of the component stakeholders. With that in mind, the management of the software development process has to be just as flexible and must not constrain any one team from applying techniques or team structures they desire. With the PMCT approach, the encapsulation of the interfaces, actions and documents in XML, leaves open the possibility to easily export information to a totally new format if it were required. As a more immediate benefit in the case of XP it was also thought that it would allow the automatic creation of XML based documentation directly from design artifacts, source code and test results that could then be imported into the management domain without interim stages.

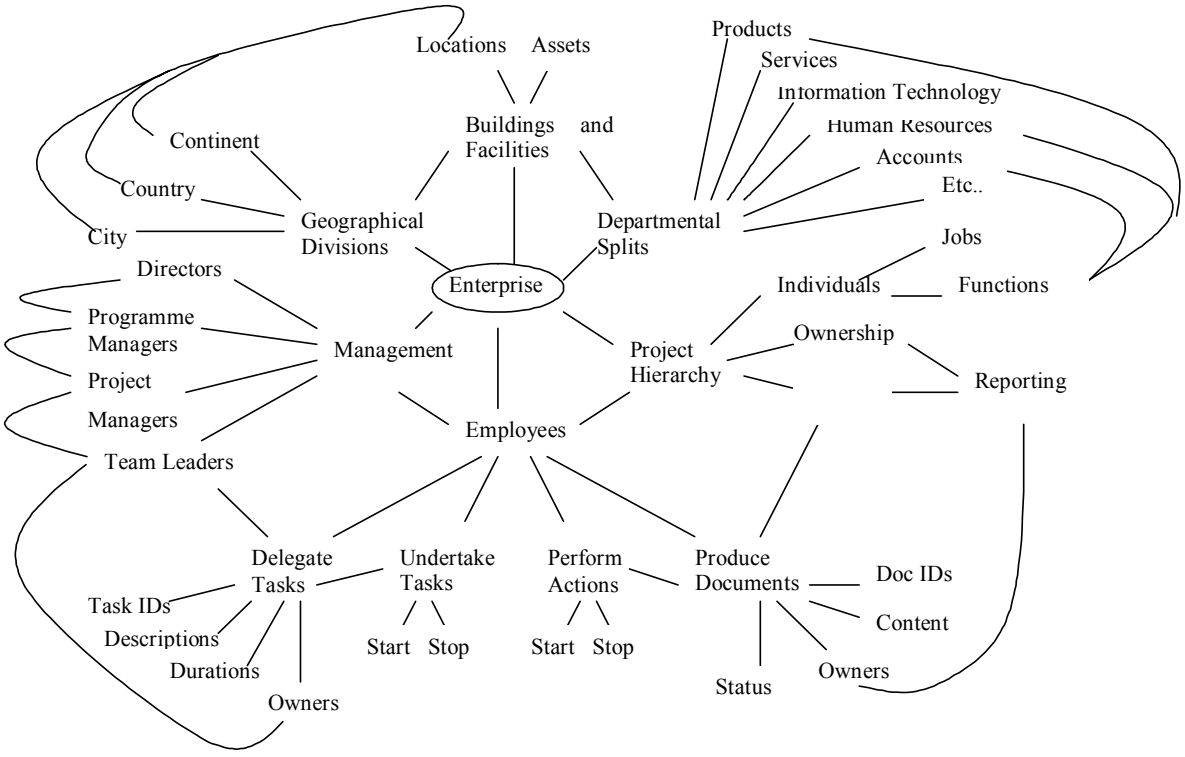

Figure 2 –Organisational Spider Diagram

Target Enterprise

In an attempt to provide a template on which to create PMCT structures the business level functions of an SME were defined and a spider diagram capturing most of these is shown in Figure 2. The 'Enterprise' is at the centre with the 'Management Hierarchy' and 'Project Hierarchy' nodes fanning out it in an attempt to model the specific software producing entities and their relationships.

Because of the general trend towards a client interface deployment favouring the use of a web-services approach and because the information to be contained within the PMCT was textual, it was thought that the best approach would be to employ the use of logically structured forms for text entry that had 'clickable' lists of links to XML documents. The main presentation structure was identified to be frame or table based with a consistent look and feel (layout and colour palate) along with a persistent set of shortcut links in a separate section or 'navigation bar' that allowed users to see where they were in relation to the data they might be viewing.

Where queries needed to be executed to build up specific information it was envisaged that pull-down or scrolling list boxes were to be used in an attempt to provide a better mechanism of building the desired database queries over user entry to minimise user error. Obviously provision needed to be made for customisations to queries to take place to cater for non-standard queries, but on the whole text entry also needed to be restricted to fields that could be parsed before running against or applying to the NXD.

Because the PMC tool was defined for use by all user types within an organisation it was envisaged that no matter what the user level, they should have the ability to create new interfaces and use the tool to capture their own self generated actions. To prevent corruption of the database structure it was thought that these abilities should be restricted to project level activity. It was anticipated that the hierarchical management structure would in large parts have had an equivalent user level structure and that mechanism should have been used to preserve the shape of the database tree and the path referencing into the database branches.

Tool management capability was also captured as a requirement and that it would follow the same style as described above. An administrator facility was also deemed to be essential and that it was to have the ability to create, delete and update user information.

Design

In commencing the design of the PMCT, it was assessed that the need to develop the eventual structure of the database backend would be fundamental to getting the overriding task or business application objectives met. In creating the database structure the design of the framework for implementation could follow that would eventually lead on to the presentational designs being firmed up.

Working from the spider diagram, the process of normalisation was undertaken spanning the 'unnormalised' form (UNF) through to First, Second and Third Normal Form (FNF, SNF & TNF respectively).

As the transition into the project design stage was made and technology selections were being firmed up, the suitability of a NXD was brought into question. Thought was given over to how the model of the stored data for this application would map on to the internal structure used in most NXD's. It was found that the way in which they store information and the types of application they are suited to (Content Management Systems - CMS) suggest that information held within them is usually constructed as a document. In CMS these documents typically contain some marked up metadata within the document and the stored item will typically be under control

and have a specific place in a document hierarchy. In this way the set of information is said to be ‘document centric’ and a document tree is built out of the objects as they are stored making it a strong candidate for use.

One of the preliminary steps in the design was the first pass attempt at creating data structures for the PMCT. The resulting database designs showed that set captured largely ‘data centric’ information with documents persisting at the lowest level. On first inspection it was thought that the information did not fit the natural NXD document centric framework, but it was also observed the data was still well structured and could quite easily be translated from a traditional RDBMS entity relationship diagram (Figure 3) into a hierarchical design of the document objects (Figure 4).

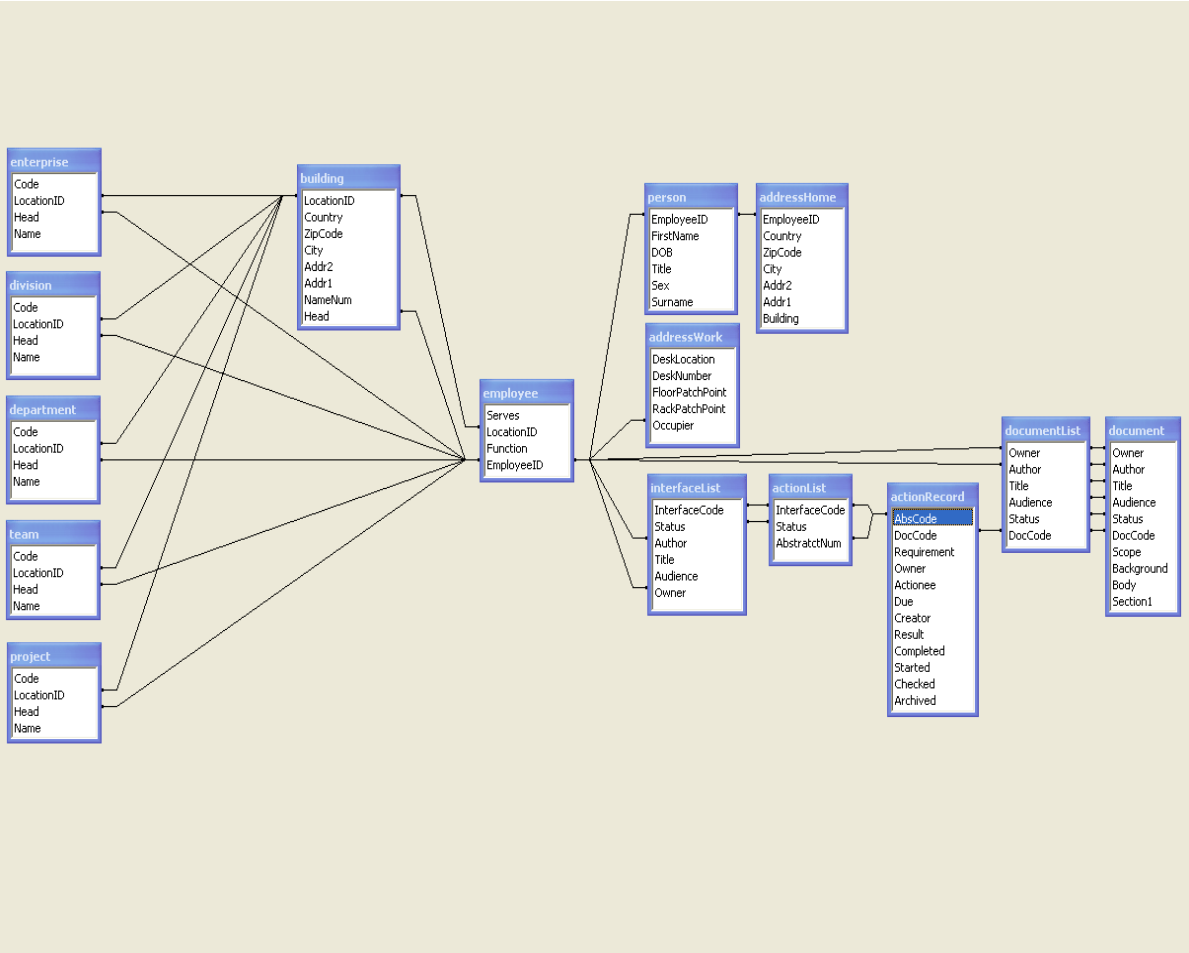

Figure 3 – Preliminary RDBMS Entity Relationship Diagram

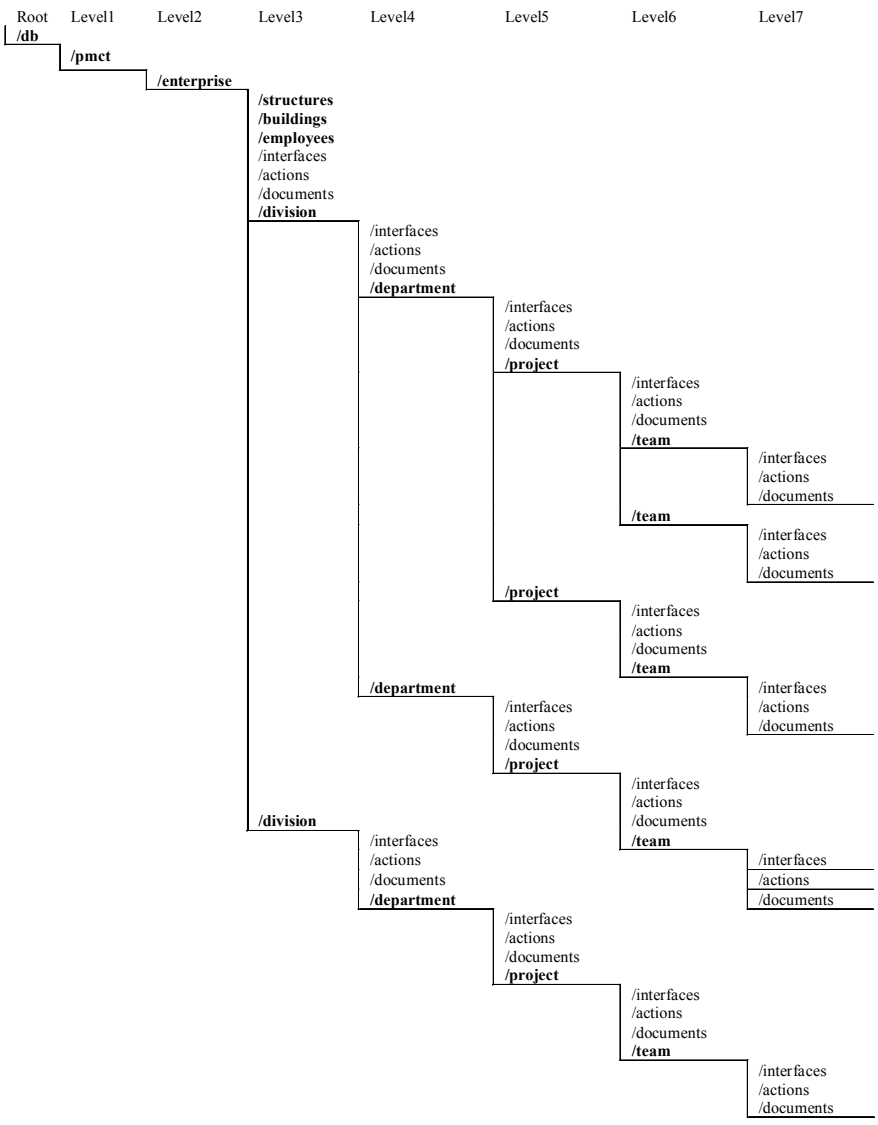

Figure 4 – NXD Collection Herirachy

In the normal course of events the core design steps are then largely complete, an appropriate RDBMS could be employed and the forms and queries built. However, after taking the above design considerations into account, the ERD was revisited and modified before moving to the hierarchical structure and derivation of appropriate reference 'paths' for the storage and retrieval of the information in an NXD.

Like RDBMS's, NXD's data sizes and quantities will vary with every organisation and it was anticipated that the database size and node calculations would be a simple function of the organisational size, the users of the system, the degree of data entry, or resolution of entries made by the users. When these were applied the PMCT system basic performance assessments in terms of size and scalability could be made. The detailed information is not reproduced here however the estimate for the total number of XML documents required to be stored in the hypothetical organi-

sation resulted in just over 50,000 artifacts in a rolling three year cycle. The overall database size was found to be running at a shade over 40GB that would be well within the capabilities of an entry level PC that an SME might use to host the PMCT and easily within the scope of the NXD eventually selected.

On going literature searches into NXD capabilities in this particular area suggested that database collections over several thousand documents can cause a drop off in the database engines performance and sub-division of the collections should be considered (Meier, n.d.). But for the depth and quantities expected this was not expected to breach any of the database choices. Initially the approach being taken was to keep the collection hierarchy as simple as possible at the expense of performance, but with collections sizes of >2000 causing unacceptable delays re-partitioning was conducted. In splitting the larger database elements into sub-divided collections and storing these under the structural hierarchy chosen to facilitate simpler XPath querying it was expected that the performance of the query when run should then be within acceptable response times (<500ms).

Since the concept of the PMCT was raised, a central theme persisted in making use of a NXD at the core. Preliminary research into tools and technologies in this area uncovered a specific drawback to this approach, specifically the decision to use either a NXD backend or a more traditional solution because of the security aspect. At the specification stage the known facts led to the decision to continue down the NXD route. This was decided because security of the information on such a non-sensitive project did not warrant a directional change. This important requirement would undoubtedly carry significantly more weight in any real world application and as such would likely influence the ultimate choice of technology.

Quality of a database application was found to be a very difficult characteristic to measure, not least because of the many perspectives from which individual user requirements needed to be viewed. As with the security, quality did not get a very high profile at the requirements capture phase, however the significance couldn't be ignored. In selecting a quality database backend, only basic performance questions were asked: How many records can it hold? How large can each record be? What is the response time in a given circumstance? What platforms can it run on? The answers for each of these ultimately affected the decision taken based on the unique requirements weightings in this circumstance, often referred to as the "value of a requirement" (Hull et.al, 2002). In the case of the PMCT, the requirements and hence constraints were minimal and the overriding concern was not so much a quality issue, more an issue of low cost of deployment so as not to preclude SME adoption on grounds of price.

As with performance of any software system, crucial to the success of an application is the way in which it allows users to interact. In designing the GUI care needed to be taken to ensure the user was provided with the set of tools they need to do their job, without adding unnecessary features or steps in order to achieve the desired result. If the PMCT were to be designed as a custom client application that received its data from a database server then many templates could have been followed. One such style guide could have been the Open Software Foundations MOTIF (release 1.2) (Open Software Foundation, 1997). This and other such guides layout the boundaries within which a GUI is considered to be conformant and is likely to receive wider acceptance through adherence to the recognised template. However, for web-based developments the boundaries are much looser and most all sites take their own direction.

Implementation

Considering a model management structure and looking to the '– Preliminary RDBMS Entity Relationship Diagram' (Figure 3) it was evident that the hierarchy needed for efficient storage of 'Employee' information in an NXD would have to come from the 'Enterprise', 'Division', 'Department', 'Team', 'Project' and finally the 'Employee' path. Given this assumption, the choice

was then made for the ‘Enterprise’ node to be the root element in the paired tree in relation to the ‘InterfaceList’.

During the analysis phase it was also thought that in an attempt to make cross referencing between the two logical structures as simple as possible, the primary key for the ‘Enterprise’ table (‘Code’) would need to take on the same path information as used in the ‘InterfaceList’ path referencing, or employ an abstract reference for each linkage as a pointer. Even though the NXD selected (eXist) supports such cross-referencing (XPointer using XPath) it was decided that referencing each of the two trees with the same path information would provide a simpler mechanism in the prototype phase. In completing the ERD restructuring for an NXD storage model, the first pass ERD showed ‘Employee’ with the related tables ‘Person’, ‘AddressHome’ and ‘AddressWork’ separately but linked through the ‘EmployeeID’ key. As the design to implementation mapping activity continued the decision was taken to migrate these three elements to a suitable XML document for storage as metadata under the ‘Employee’ XML tag. The idea of a placing XML files under collections within the database can be thought of as analogous to the storage of files in a normal file system. In the case of the PMCT, the collection hierarchy (Figure 4) was arrived at after considering the RDBMS structures and its application to the prototype database.

The collections above are shown in bold to indicate the mapping of the relationship diagram elements (Figure 5) on to the document collections that were to be created in the NXD. All the XML files stored under the collections were also derived straight from the relationship diagram shown and a partial example is shown below for the XML file stored under the structures collection. In this particular case, the first four tagged entries in the XML element ‘enterprise’ that is analogous to the equivalent RDBMS table ‘enterprise’, are the ‘Code’, ‘LocationID’, ‘Head’ and ‘Name’ fields also captured in the enterprise table:

```
<?xml version="1.0" encoding="UTF-8" ?>
<enterprise>
 <code>spiess/</code>
 <locID>uk/ch41qw/mac</locID>
 <head>000-000-001</head>
 <name>Spiess Ltd.</name>

 <division>
  <code>spiess/uk/</code>
  <locID>uk/ch41qw/mac</locID>
  <head>000-000-001</head>
  <name>UK</name>

  <department>
   <code>spiess/uk/rd/</code>
   <locID>uk/ab51df/def</locID>
   <head>000-000-009</head>
   <name>Research &&amp; Development</name>

   <project>
    <code>spiess/uk/rd/a01/</code>
    <locID>uk/ab51df/def</locID>
    <head>000-000-008</head>
    <name>Project Red</name>

    <team>
     <code>spiess/uk/rd/a01/man/</code>
     <locID>uk/ab51df/def</locID>
     <head>000-000-007</head>
```

```
    <name>Management Team</name>
   </team>
  </project>
 </department>   etc…….
```

Figure 5 – XML Document for the 'enterprise' structure

This process was repeated for all the table data captured in the project design stage with some very minor alterations to make the data more consistent.

In commencing the search for an appropriate database it quickly became apparent that the number of NXD's that had potential to be used in this application was huge. The types of database spanned the whole spectrum and ranged from the large commercial variety, through to those produced by individual developers and ones that were born out of other research projects. Many commercial database offerings were initially considered, but because the project was geared round an SME's needs, selection of databases starting at $15,000 and rising to $250,000.00 were ruled out on cost grounds. In eliminating a solution based on large commercial tools the scope then narrowed to more mainstream/mass-market database and open source solutions. The initial project proposal put forward stated that the application should be platform independent leading to a sidelining of the more modestly priced mass-market or free NXD's. After extensive reading and aggressive chopping two candidates remained; consequently both xIndice and eXist were downloaded, built, trailed and sample applications experimented with and which eventually leading to the selection of eXist.

In selecting a platform independent backend it was possible to decouple the selection of operating system and machine from the application part. This separation was seen as essential for 'best fit' into target SME's and for opening up the project to further research. The three mainstream platform choices available for the project were Sun Ultra 5, Solaris 8; Intel PIII, RedHat 7.2 and AMD Athlon 1.4, WinXP.

Because all three provide support for the Java Runtime Environment on which the selected NXD sat, the choice of platform was made solely on the grounds that would suit the largest community of individual developers. The single most prevalent operating system that is available to other students and was likely to be available to a large majority of SME's is Windows.

With eXist and the Windows platform selected the only decision left was the language choice. As with traditional databases that employ core Structured Query Languages (SQL) and variations, in choosing an NXD the language supported had to be XPath for referencing elements within the database, XQuery for querying the information stored and XUpdate for effecting changes to the data.

Considering again some of the central themes of the PMCT, the requirement to make the application available to all within an SME, the need for the user interface to be familiar and easy to use and the future possibility that it may need to be deployed across an organisation with geographically spread and architecturally different client machines; the natural choice was to make use of internet browser functionality rather than deploy a client distribution with a custom user interface. The majority of user equipment provides support for one browser or another, and if the user changes location or the workstation hosting the client application needs replacing there is no impact on replicating an environment or build state to follow the change. Similarly the look, feel and operation of the tool suite could be manipulated again without the hindrance of having to synchronise client updates and was an additional benefit of having a browser-based application. It was all these factors combined that pointed directly to the use of HTML, XML and XSL to shape the interface.

Although the initial research and design work undertaken had resulted in Windows XP and eXist, the method of implementation still needed resolving. The language choices, HTML, XML and XSL to shape the browser-based interface were already defined but the decision on the serving method persisted. Whilst evaluating eXist it was found that it provided several interfaces allowing interrogation of the XML data through the database engine. These were: SOAP, XML:DB API and the XML-RPC interface. Having the flexibility to connect the user through to the backend database in a variety of ways left sufficient scope at the design stage for the implementation to be steered and a limited investigation into the methods revealed many benefits and demerits in each. One such feature of the publishing framework eventually chosen, centered around its ability to extend the basic prototype HTML output specified to cover a wider variety of generated outputs including PDF, SVG and WAP formats. The ability on all these fronts made Cocoon the ideal choice for the PMCT to fulfill the needs of the work in connecting the NXD to the HTML output format. The way in which Cocoon leverages this flexibility is by inserting itself between the backend systems and the client elements in what is described as a three-tier architecture (Langham & Ziegeler, 2002):

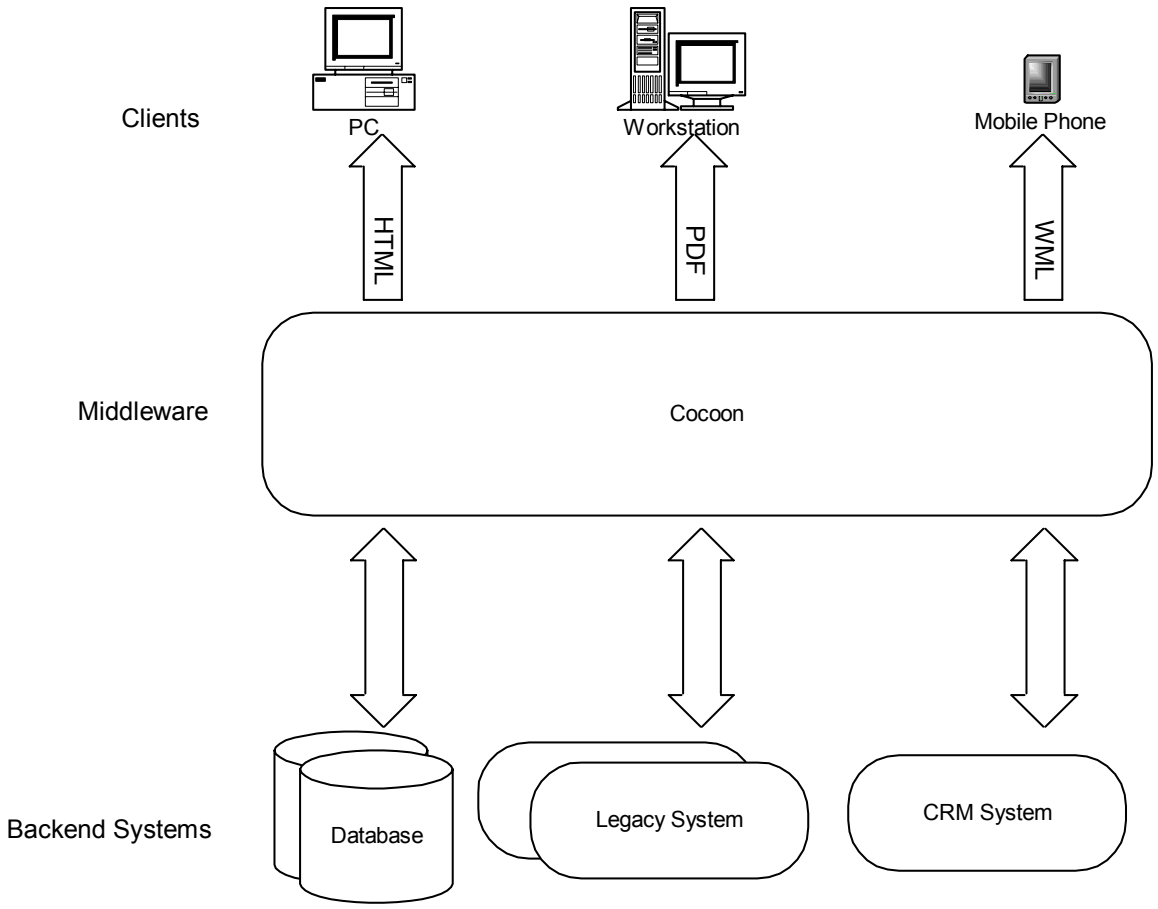

Figure 6 – A Cocoon-based, three-tier architecture (Langham et.al., 2002).

However, regardless of the future possibilities afforded, because Cocoon already provided direct support of XML:DB enabled databases the basic ability to bridge the gap between the direct provision of the interface to the XML:DB API supported by eXist and the browser based user interface defined previously made it the natural framework for selection.

Putting the PMCT application together from what appeared to be disassociated and distinct parts then followed. Steps were taken to first build the development environment comprising of a Java runtime environment on which to run the simple text editing tool Jext and the Jetty web-server. On the Jetty web-server and servlet container, the Cocoon servlet ran communicating through the Cocoon framework to the eXist backend also running as a servlet.

The intial steps taken were to install and prove the Jetty webserver on the WindowsXP platform. After that the eXist servlet was installed and started, test XML files added to the database and then the process of implementing the basic PMCT structure began. This process was then repeated, refined and the full implementation tested on an ongoing RAD basis.

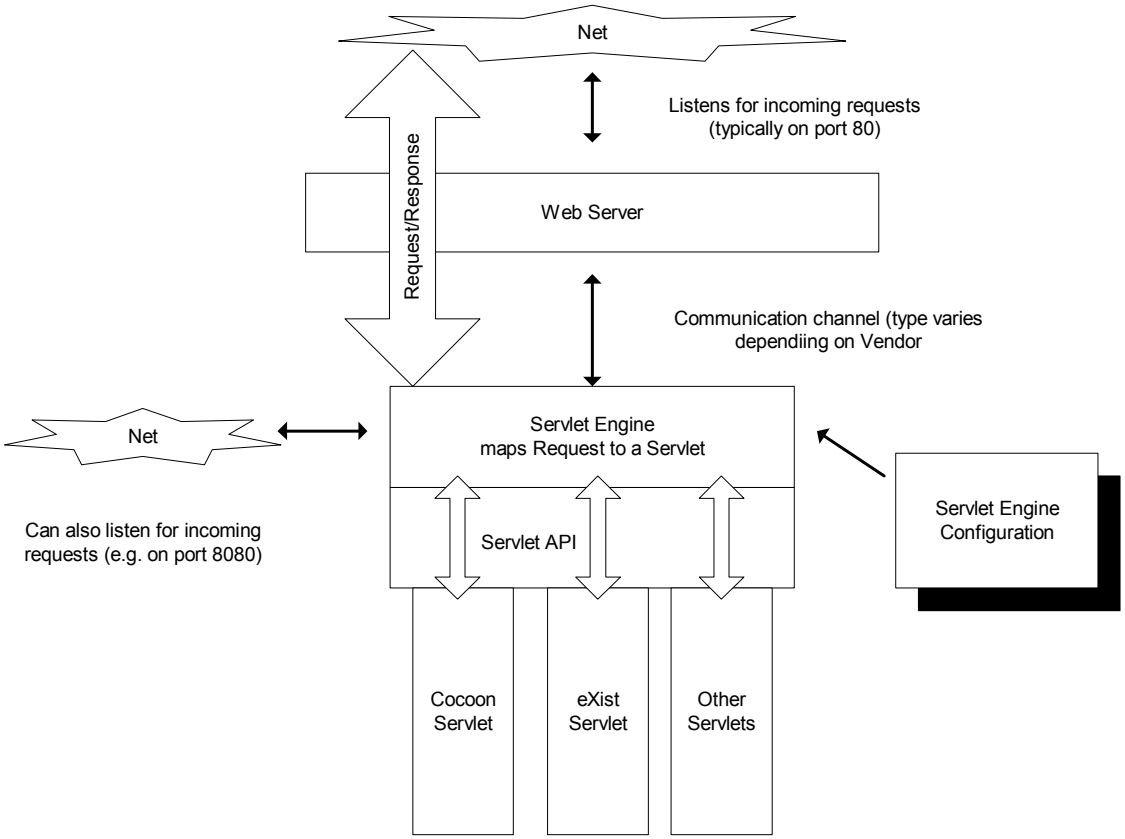

Figure 7 – Cocoon: The big picture (Langham et.al., 2002)

Conclusions

From the outset the PMCT project objectives were quite specific yet provided a reasonably flexible model through which they might be achieved. In creating an integrated suite of tools that allowed programme managers, middle managers, team leaders and staff co-ordinate reporting and daily activities within a software project, the PMCT development has achieved all but one of the aims (XUpdate). The main intention was to create the tool is such a way as to promote effective co-ordination and clarity of responsibility and in meeting the fundamental web-based delivery requirement this was seen as a success.

By building the PMCT on a middleware framework that operated on a JVM, the chosen core technologies met the requirement for the suite to be available on a cross-section of platforms allowing it to be opened up to differing industry environments. Towards the projects completion, an

assessment of the way in which users at each end of the knowledge spectrum with respect to the PMCT could extract and interpret information to gauge whether it met the basic requirement of clarity and logically presented data. Following this the development lifecycle was revisited and evaluated with a focus on the processes followed, selections made, features available and gave rise to the subsequent observations. In developing the PMCT on a NXD and using browser technology to deliver the information to the user, it proved that the information flow fundamental to any enterprises programme management activity can in fact be captured and controlled simply, efficiently and cheaply. It was observed that the three attributes mentioned were primarily inherited from the Cocoon publishing framework and not as a result of direct benefits derived through selection of the NXD, but the clarity of end-to-end operation was significantly enhanced as transparent storage of the original XML in the backend database took place. By selecting an NXD for such an application the ability to exchange internally captured management information between other tools is strengthened and supports the spirit of maturity and continual processes development within an accredited software process improvement framework.

One particular feature of the Cocoon publishing framework eventually chosen, centred around its ability to extend the basic prototype HTML output specified to cover a wider variety of generated outputs including PDF, SVG and WAP formats. From the tests done on the PMCT using a 3G based Motorola A920 supporting a WAP2.0 browser using a 64Kb PDP context it is thought that future work could be easily extended into the arena of mobile access of the PMCT so users can interrogate the stored information without major rework normally associated with these shifts. Variations on the presentation of published information in natural paper based and documented reports using the PDF capability could be investigated. Also with the support for SVG output, reports could be made more meaningful and their effectiveness compared against table based mechanisms when organisational structures can be expressed graphically, and trends & summary reports can be expressed as charts of various types. Encompassed in the full study was the analysis of other PMCT like products on the market that already supported GUIs with drag and drop facilities to enhance productivity. Applying this capability to complex query statements would be seen as the ultimate goal in data retrieval and productivity and one method of achieving such an offering was researched. The JGo libraries offered by Northwoods Software (n.d.) could possibly extend the PMCT to allow pictorial representation of the enterprise structure, manipulation of it and the general process of modelling management information flows.

A more modest step to apply to the PMCT would be application of alternative language resource files within the Cocoon processing catering for language transformations to be applied and resulting in a multi-language offering opening up the research into how the PMCT would be rolled out and accepted by other nationalities.

One final and very important new topic area not covered by the summary, would be in assessing the usefulness of a tool such as the PMCT and its impact on information security; how by concentrating the information flow between teams through a tool that acts as a central repository, the mesh of inter- or intra-organisational chatter might be minimised reducing the risk to breaches in IPR or misinterpretation of critical internal communications.

Overall it is believed that NXD's do offer significant benefits to more than just SME's and it's clear they can be put to use in a wide range of ways, but it is also observed that until NXD's become more widespread and available to developers in supported and mature packages at affordable prices, their core user base will be confined to the high end developer providing global enterprise class applications and that of investigative and speculative developers in the academic community.

This paper is a continuation of work previously carried out for the authors M.Sc dissertation submitted to the University of Liverpool (2004). Authors would like to thank the Laureate Online Higher Education and the University of Liverpool for their support and guidance.

References

Beck, K. & Fowler, M. (2000). *Planning eXtreme programming*. Addison Wesley.

Dale, R. (n.d.). Process and enterprise modeling tools. *ESRC Business Processes Resource Centre*. Accessed August 2003 at http://bprc.warwick.ac.uk/bp-tool.html#SEC2d

Hewlett Packard. (n.d.). Web service platform. Accessed September 2003 at http://www.hpmiddleware.com/SaISAPI.dll/SaServletEngine.class/products/hp_web_services/default.jsp

Hull, E., Jackson, K. & Dick, J. (2002). *Requirements engineering: A Structured Project Information Approach*. Springer.

Humphrey, Watts S. (1998). Three dimensions of process improvement Part I: Process maturity. Accessed August 2003 at http://www.stsc.hill.af.mil/crosstalk/1998/02/processimp.asp

IBM. (n.d.). Web sphere portal express. Accessed August 2003 at http://www-306.ibm.com/software/genservers/portalexpress/

iGrafx. (2003). Process central 2003. Accessed August 2003 at http://www.igrafx.com/products/processcentral/

Institute of Management Services. (n.d.). Introduction to the IMS. Accessed August 2003 at http://www.lmu.ac.uk/lis/imgtserv/prodweb/institute/intro.htm

International Standards Organisation. (n.d.). '.ISO FAQ'. Accessed August 2003 at http://www.iso.org/iso/en/iso9000-14000/iso9000/faqs.html

Jones, G. & Edgell, R. (2003). Total Quality Management – TQM. Accessed August 2003 at http://www.managers-net.com/total_quality_management.html

Langham, M. & Ziegeler, C. (2002). *Cocoon. Building XML applications*. New Riders.

Meier, W. (n.d.). eXist: An open source native XML database. Accessed August 2003 at http://exist-db.org/webdb.pdf

Northwoods Software. (n.d.). Java Diagram Graphics Libraries. Accessed September 2003 at http://www.nwoods.com/go/jgo.htm

Open Software Foundation. (1997). Open *Software Foundations MOTIF User guide (release 2.1)*

Sky Mark (2000). ISO9000:2000. Accessed August 2003 at http://www.skymark.com/pathmaker/uses/iso9000_2000.asp

Software Engineering Institute. (n.d.). CMM – Capability Maturity Models. Accessed August 2003 at http://www.sei.cmu.edu/cmm/cmms/cmms.html

Software Measurement Services. (2000). Software Measurements and Process - CMM Feedback Issue 7 (Winter 2000). Accessed August 2003 at http://www.gifpa.co.uk/news/7/p2_cmm.html.

Software Quality Institute. (1999). SQI. Accessed August 2003 at http://www.sqi.gu.edu.au/indexFrameset.html

Solutions Consulting. (n.d.). Creating High Performance Organisations. Accessed September 2003 at http://www.solutionsconsulting.org/services.html

TickIT. (2000). Costs and Benefits for Software Suppliers. Accessed August 2003 at http://www.tickit.org/quality.htm

Vu, John.D. (2001). Process Improvement Journey (From Level 1 to 5). Accessed August 2003 at http://www.processgroup.com/john-vu-keynote2001.pdf

Wright, David.T. (n.d.). Thirty Years of Project Management What Have We Learned? Accessed August 2003 at http://bprc.warwick.ac.uk/repwinch.html

XML:DB. (n.d.). XML Update Language. Accessed December 2003 at http://www.xmldb.org/xupdate/xupdate-wd.html

Biographies

Dr. Samuel Sambasivam is the chairman of the Department of Computer Science of Azusa Pacific University. Professor Sambasivam has done extensive research, publications, and presentations in both computer science and mathematics. His research interests include optimization methods, expert systems, Fuzzy Logic, client/server, Databases, and genetic algorithms. He has taught computer science and mathematics courses for over 24 years. Professor Sambasivam has run the regional Association for Computing Machinery (ACM) Programming Contest for six years. He has developed and introduced several new courses for computer science majors. Professor Sambasivam teaches Database Management Systems, Information Structures and Algorithm Design, Microcomputer Programming with C++, Discrete Structures, Client/Server Applications, Advanced Database Applications, Applied Artificial Intelligence, JAVA and others courses. Professor Sambasivam coordinates the Client/Server Technology emphasis for the Department of Computer Science at Azusa Pacific University.

Mr. Paul Storey is an accredited representative of the Association of Project Management Group (APMG) practitioners currently working for Fujitsu Telecommunications Europe Limited. His professional background is centered on the management of software development departments for an array of international companies within the telecommunications domain such as Lucent, Siemens and O2. He has had practical exposure to many methodologies, development processes, tooling and languages such as PRINCE2, UML, SSADM, RUP, C++, C, JAVA, PERL, etc. Having observed a wide variety of programme and project coordination techniques during his 12 year career in the industry, he has been able to extract many practical lessons and consider how the ongoing changes in his discipline might be served by Internet based tooling. His academic pursuits whilst studying for an M.Sc. at Liverpool University covered interrelated areas such as CORBA (Common Object Request Broker Architecture) and web base developments.