Introduction

It is necessary when implementing an IBM FileNet P8 ("P8") application to analyse and design the content and process management behaviour of the solution to meet the desired business requirements. Two separate, but related, design activities are required; one focusing on the content used or generated within the application, and the other focusing on the processes that define the movement of work within the application.

This document is intended to assist readers in their understanding of the process and design methodology by providing the goals of the methodology along with the relevant rationales and discussion. A practical guide to the implementation of the methodology is provided in two separate documents: the P8 ECM Content Design Handbook and the P8 ECM Process Design Handbook. These handbooks describe the activities that need to be conducted and the inputs and outputs to be expected for each phase of design. Both handbooks follow the practices described in this design methodology.

Additional design is required when considering applications contributing to a global solution that is built upon content and process management. A global solution will encompass common infrastructure and many individual applications. A single application will represent a subset of the global requirements, and must fit within a common framework.

A major benefit of a global solution is reuse. Design activities performed by each individual application should contribute to the global solution, which will ensure that the processes and content behaviour of other, or future, applications can benefit from these activities. As a global infrastructure, each variation introduced into the solution will add to the management and maintenance effort of all solutions. By approaching solutions as contributors, rather than exceptions to the global solution, reuse is increased and variance is reduced. The result is a more manageable solution, which directly reduces the design effort required by each subsequent application added to the solution as well as the ongoing maintenance costs.

There will be exceptions to the design approach. In some cases, applications are radically different in their requirements or implementation techniques. These exceptions need to be handled correctly. The general approach should be to minimise variance, but only to the extent that it creates more value to others than it detracts from the individual application. Where there is greater benefit to others or little detriment to the current application, exceptions must be allowed, documented and managed.

The remainder of this document focuses on the activities required for the analysis phases of the design methodology. This is to support the definition of a global solution, and to assist in the decisions necessary to determine applications that fall outside this approach.

Content and Process Design Approach

It is often the case that resources that are proficient at content design are not the same as those that are proficient at process design, and vice-versa. It is prudent for all resources to establish some awareness in each discipline, though specific expertise is required in each. An overall view of both is needed, and dedicated activities in each are desired.

The design approach must start with an analysis and agreement on the elements and components of content and process management that will be utilised. Once agreed, each discipline must separate and perform the desired analysis of the relevant requirements and determine the most appropriate techniques to utilise. Defined milestones must be established where each discipline comes together and a joint content and process design is expanded. Again, the disciplines will separate and focus on their individual expertise, until finally they join and provide a combined content and process design that can be implemented.

These activities should be separate, as performing the activities within the same stream complicates each individually. Content and process management have different points of focus, and each has some overlap in the underlying system requirements. Performing the activities together complicates each design activity and will result in application specificity that isn't desired. A more reusable solution is likely result when designers separate and focus purely on one discipline each, as each individual designer considers the existing and desired components for the application within their context without being constrained by the overlapping elements of the design.

The phases should focus on the following activities. There will be interaction between the content and process activities during each phase, to ensure consistency of analysis, with a specific joint design effort at the end of each phase.

Phase	Content Activity	Process Activity
Phase 1	Content Management Style and Content Characteristics	Business Logic and Process Segmentation
Phase 2	Content format and classification	Data referencing and retrieval
Phase 3	Content auditing and reporting requirements	Audit and Reporting Requirements
Phase 4	Content creation, publishing and lifecycle	Integration, Event, Timer and Escalation Model
Phase 5	Content security and lifecycle	Process Security
Phase 6	Content storage, retrieval, delivery and access	Queuing and Work Delivery Model
Phase 7	User interface	User and System interfaces
Phase 8	Coding specification	Coding specification

Each of the phases may require that earlier phases are revisited. Changes that are required in one phase may impact the output from prior phases and these must be reviewed in their own context. The combination of content and process outputs from any phase may also cause earlier phases of either stream to be revisited. This should be expected, and is essential to ensure that a common reusable structure is obtained.

The decision to move from one phase to the next phase must be a deliberate choice. No later phase can be started until some element of each earlier phase has been considered. A phase can have an empty set of activities, but the inputs and outputs must have been considered and also be represented as an empty set.

The duration of any phase, and the assimilation of the output from the content and process activities, will vary depending upon the size of the process and content being analysed. In cases where only small, simple activities will be required, these tasks may be able to be delivered through a single resource. The activities define the states that should be analysed, they don't represent separate long term analysis activities. The path through the activities is defined to formalise the thought processes involved and doesn't necessarily reflect mandatory separation of project teams; though not separating project teams introduces a risk that the process design is driven by content requirements or vice versa, which will reduce the reusability of the design.

Content Analysis

As the design of any solutions is likely to be based on earlier solution elements, it is important to know the existing structures that will be reused within the solution. It is also necessary to identify those characteristics that will not be reused. As the goal of the design methodology is to support future change, decisions and analysis should be considered as input to future solutions as well as the current solution. An isolated, single-project based view of decisions will not support the modification the system at some future time. Decisions must be based on consistent documentation from earlier analysis sessions, and must be produced so that later designers can readily understand and synthesise these analyse. Clear and descriptive documentation of your analysis must be produced, and this documentation must be targeted at an audience that potentially knows nothing about the project being analysed.

The phases of analysis are designed to guide the gathering and deconstruction of information in order to assist in the design of the content management structures of the solution. The phases are not mandatory, and may not be sequential, but each phase should be at least considered when designing your content structures. The phases may also be iterative, meaning that information you gather in one phase may cause you to revisit the activities in an earlier phase.

Phase 1 – Identification of Content Management Style and Content Characteristics

This phase must identify the characteristics of the content and the style of content management that will be used by the solution.

Styles of Content Management

There are many sub-classifications of content management. Each has its own requirements, expertise and impact on any solution. The style of content management that is required must be determined to ensure that the correct resources and approaches are used during content analysis.

Content management covers a broad range of attributes, and each content style has a slightly different priority of the attributes that should be considered. This therefore drives the analysis activities in slightly different directions based on the purpose and style of the content involved in any business solution.

The styles of content management that should be considered are:

1. Document management. Document management focuses on the creation and management of business documents, such as those created using Microsoft Office applications. The documents usually involve the creation, publishing or capture of information. This information may form part of other applications where correspondence to customers is tracked, or may be reused on web sites in a publishing model. Versioning, publishing, searching and lifecycles of documents are areas of interest for this style.

2. Image Management. Image management focuses on the capture and retrieval of scanned documents. It is typically supportive of high volume transactions that result from the scanning of paper documents such as applications, forms or objects requiring signatures.

3. Email Management. Email management focuses on the capture and management of email messages, both to and from an organisation. Email traffic is used to support many business transactions, and this style is driven by the need to ensure compliance with retention and disposal policies and ensuring that complete records of business interactions are maintained.

4. Records Management. Records management deals with the management, retention and disposal of objects that are deemed records. Records are typically items that have significance to an organisation due to their purpose, their content or their context. Sometimes records must be kept in order to comply with legal requirements, though these requirements differ between legal jurisdictions. Records management solutions provide for the management of the access, storage and destruction of these important assets.

5. Case Management. Case management is a content-centric process solution, typically involving the initiation of knowledge worker business processes based on the receipt and collection of document based data. The content is application specific, but the processing focuses on the movement of the content through review, information gathering, processing and approval style process segments.

Characteristics of Content

Content characteristics will determine whether certain components of the FileNet P8 platform are utilised in a solution, and how the components that are used are assembled. Each characteristic of each type of content that is stored within the solution should be defined in order to enable the most effective solution to be selected. Whilst many individual content objects will not have been identified at this point, the business requirements for the solution should reference the major content objects that the solution will be concerned with, and it is the characteristics of these objects that will have the most influence of initial solution design.

The content characteristics that should be considered are:

1. Format. This refers to the format of the paper being scanned when the solution will incorporate an imaging functionality. The size of the paper, the scanning resolution and the colour depth will all affect the size of the resulting images files. The size of these files will determine such things as network bandwidth requirements, hardware configurations and caching models.

2. Type. Where a piece of content is created electronically, such as with a desktop application, its type refers to the file format of the piece of content. For example, a Microsoft Word document will require different storage requirements than video editing files. As with format, the type of content will affect its size, and this will again affect such things as network bandwidth requirements, hardware configurations and caching models.

3. Volume. The volume of content refers to how much content will be added to the system over a certain period of time. For example, the requirements of a solution may state that the system will need to ingest one million pieces of content per month. The volume of content being ingested will affect system performance, network bandwidth requirements, storage topology and the number of servers required.

4. Frequency. The frequency of content refers to peaks and troughs in the rate of content ingestion/retrieval. For example, the requirements may state that the solution must be able to handle an ingestion peak between 9am and 12am every day, with relatively minor ingestion required for the rest of the day. This is especially relevant information when multiple virtual servers are deployed on the same hardware. Matching the virtual servers of applications that have complementary frequency on the same hardware will yield the most effective use of resources.

5. Location of creation/Location of storage. If content is created in a location that is remote to where it needs to be stored then this will impact storage topology, caching models and network bandwidth requirements.

6. Required Metadata. A basic behaviour of any storage and retrieval solution is the ability for someone to locate specific content objects. For this to occur definition of the attributes that uniquely identify each object is needed and the data values need to be provided when the object is acquired.

   Data that is unique and as well defined as possible should be chosen to support the simplest and most efficient retrieval or access to any object. Values that identify related objects should be identified in order to provide context and identity. Many of these data items will also be used to identify work related to the content or interactions with other systems.

7. Security Requirements. The security and audit behaviour for stored content needs to be defined. There are varied techniques available for supporting a wide range of security models. The fundamental security requirements for a solution need to be known and considered when determining deployment configurations. This includes audit requirements, as they will also impact the techniques and configurations that are supported. If any lifecycle transitions are required to occur on content objects used by the solution then this implies that more than a basic storage and retrieval solution is required.

   Audit requirements imply that event and data related to events must be retained within the solution. These events and data must be defined. Differences in security and audit requirements can affect storage topologies and identifying data, which in turn can impact class structures.

8. Static or Dynamic. Content can be static or dynamic. Static content can be considered as unchanging over the life of the content, such as a scanned image. Dynamic content is likely to change during its life, such as a document that will be expanded and modified, or perhaps re-purposed.

   The process lifecycles of static content and dynamic content are different. Static content is likely to be stored only once, and may be referenced rarely over the life of the content. Dynamic content may be stored, modified, versioned, retrieved and continually referenced over a short period of time that may recur periodically. For exampe, a document that is created, reviewed, modified, approved and then revised each year is representative of this behaviour. Each year a period of activity occurs, which must be supported topology and components of the solution architecture. Within a case management model this is represented by periods of communication with customers as their requests and applications are processed. Periodically marketing activities occur and changes are made to the content objects that represent them to the solutions.

9. Simple or Intelligent. Content can also be simple or intelligent. This is not a representation of the amount of data in the content, but more the ability of the content to be extracted, utilised and possibly re-purposed during its life. Scanned data that is data entered into another system and remains solely for validating information such as signatures on the pages can be considered simple. The extracted data from the static image may be considered intelligent. Content that is inherently intelligent includes data such as Word documents, emails, electronic forms and XML data. These are examples of content that can be utilised and repurposed over their life, typically through automation, rather than statically extracted information that is used for longer term reference only.

In many solutions, a mix of content management styles will be required. In case management solutions, there are many examples where image data is delivered to initiate a case, whilst documents and forms are used to represent user interfaces and to communication with customers. Each of these has its own unique handling and implementation requirements, and each must be understood separately yet combined within the same solution.

One of the goals of this phase is to ensure that minimal duplication of definitions occurs and that reuse, if possible, is achieved. As a system grows, and content management styles are identified, there must be an evaluation to determine if new content structures are similar to previously defined content management styles. This phase deconstructs and reassembles styles of content; looking for similarities in the tools, components, techniques, attributes and identifying characteristics of content.

Different content management styles can affect the techniques, topology and behaviours that are apparent in a content management solution. Once a style is determined, the identifying characteristics of the content must be established and the resulting implementation techniques can be selected.

Imaging based solutions at least two specific components: the hardware and software required to scan the images into the system, and a viewer application that will display the images that are retrieved. Other components exist for different types of content such as word processing documents, spreadsheets or presentations. The tools used to create these types of contents are usually also used as the retrieval, modification or viewing component for the same content. An extension to this style can be shown when the viewing of the same content is performed with a different format of information. The publishing model is representative of this variation. The original document formats remain within the system, and are likely part of a dynamic style of content management, whilst a static published version of the dynamic document content is used by others within the same system. For example, a static PDF version of a document could be made available on a corporate intranet after the dynamic version of the document is created, reviewed, and approved. The original document is still a dynamic intelligent document, whilst the PDF rendition of the document becomes a static intelligent document.

The identifying characteristics of the content, such as its format, use, and unique data attributes that allow someone to know what it is and possibly for what purpose it is used, are all attributes that must be analysed. These gross characteristics will be used to identify the document class, whether pre-existing classes can be used to represent the style of any document, and to group together information that has similar characteristics.

A system becomes more complex as more classes are added to it. As the system grows, it becomes more difficult is to isolate, identify, repurpose and manage the classes within it. This first phase supports grouping information together by the grossest possible means, ensuring that where commonality exists it will be identified, and where gross differences exist they will be readily apparent. If too much detail is defined at the start, then the differences will outweigh the similarities and the opportunity for reuse and repurposing will be reduced.

Each unique class within the content engine is used to represent unique properties, security or behaviours. Classes should not be created simply because something has a name or characteristic that is different. Classes must be created to represent the widest possible characteristics for the widest possible list of objects.

The identifying data for any object will be used to initiate, identify and track the behaviours that are implemented within the lifecycle of the object. It is important that unique data is available to link transactions, events, processes, audit and retrieval activities within the entire solution. This identifying data is an element of both content and process design, and will likely be common where content and process systems interact. For example, a loan application will probably be identified by a Loan Number and/or a Customer Number and the processes that are initiated to deal with the content will utilise similar data for auditing and reporting functions. The same data may be used to assist call centres in locating the content and the processes that have used them.

This identifying data should be as general purpose as possible, without becoming vague. In many early imaging systems, unique tracking numbers were generated to match the paper to the transactions that they initiate. This data is very specific and has little relevance beyond the matching behaviour. It rarely translates to audit information within the process space as there are likely to be many different content and process interactions each with their own unique data tracking behaviour. To propagate this data for other purposes confuses the long term consistency of the data. Identifying data should be for long term analysis or retrieval, and transient data should be used for its specific purpose and not made into longer term identifying data.

Phase 2 – Content format and classification

Once the style and identifying characteristics of content have been determined, the next level of analysis of the content can be undertaken. This phase looks at the purpose and use of the information described by the content, rather than its physicality, variability or its internal representation.

The data that is present within the content will drive much of the use of the document and the actions that occur to the content. For example, imaged forms that are read by a user will usually represent business transactions that need to be entered and validated. Different data elements may be described by the image and used for different purposes. Each format of content will have similar requirements. It is the data that is contained in or on a page that is relevant for processing, classification, retrieval and for providing context. Images, email, word processing documents, XML documents, drawings and other content formats all contain data within the format of the document which must be identified and its use determined.

This data will be directed to several probable uses (which may overlap):

- the data used to identify this document in content behaviours and work processes (for example, Document Type, Application Number or Customer Name);
- the data used to retrieve and provide context to the document in the future (for example, Date Received, Business Address or Document Type);
- the data used to process the transaction or information within the document (for example, Account Number, or Loan Amount);
- the remaining data that is retrievable in the future, but not used to retrieve the information or represented in current business processes (for example, Signatories or Terms and Conditions).

This phase of the content analysis focuses on the information presented in or on the content, and determines the use of the data and the techniques that will be used to extract this information. Assumptions regarding the techniques used to store, retrieve and process content are isolated by separating this analysis phase from the identification and style of the content.

Imaging, email management, document management, records management and a number of other subsets of enterprise content management have their unique requirements. If phase 1 and 2 are executed together then data is not truly isolated in its purpose in the solution, and incorrect technology and tools may be used, resulting in overly complex or poorly performing systems.

By isolating the source, extraction technique, and usage of the data within the content from other analysis activities, more specific attributes of any object are made available to assist in later phases.

Whereas phase 1 dealt with the immediate identification of information, this phase deals with data that provides context over time, and data likely to be relevant to other systems and processes. The content engine must not be regarded as a database, but as a content management structure. The data that is analysed in this phase must be representative of content management requirements, and not process or data management needs. The layers of software that are required in content management constrain the ability of the system to be used as like a database, and attempting to do so will result in high maintenance and management issues, including performance issues, as the quantity of non-content related information grows.

The data that is identified should be classified as data related to:

1. Informational retrieval. Additional data elements that support more focused retrieval characteristics to a user or system in order to isolate and retrieve the content. For example, the date created or specific type of the content.

2. Contextual retrieval. Additional data elements that provide context about the content that has been retrieved. For example, the mime-type or business area that produced the content.

3. Records management context. Data elements that relate to the classification, retention or historical relevance of the content. For example, information that describes a board meeting, corporate reference or timeframe for destruction.

4. Processing context. Data elements that are provided to the process management system, which may not be required for content management reasons, but may be required for the initiation of business processes. For example, the value of a loan application where a business rule exists that loans over a certain value should trigger a specific approval process.

5. Embedded data. Embedded data refers to information that describes the content, remains within the content, is relevant to the content or process management of the content, but is not exposed within the content or process engine. For example, a signature on an application, or data that is contained within an electronic form or XML document which may be retrieved from the content when needed.

By the end of this phase there should be sufficient information available on the types and attributes of content in order for some analysis, deconstruction and reconstruction to occur.

The initial characteristics of the content can now be used to determine the initial class structures of the solution, the likely storage behaviours, and the tools and techniques that may be used in its construction.

The goals of this analysis should be to:

- Deconstruct the attributes and other information to their lowest sensible components.
- Identify similarities and differences in the information. Commonality of attributes should indicate similar class structures. Gross differences should indicate different class structures.
- Look for techniques for grouping similar (and dissimilar) information into a common structure.
- Create a prototype of the information gathered in order to validate the consistency of knowledge gathered, the assumptions made in class construction, and the ability of the solution to represent the information that has been gathered.

The outcomes of this analysis will be a starting point, and will not be the final solution. As later phases of analysis occur, this prototype is very likely to be modified. The ability for the design to change must be embedded within the design itself. If change becomes difficult in later phases, then the prototype is likely to be incorrect and this phase may need to be completely reworked. A prototype should never become a final solution. It is a throw away output that validates your framework and supports visualising and demonstrating your design.

Content management solutions impact system characteristics such as performance, network and bandwidth utilisation. The attributes gathered to this point in the analysis can be used to verify the topological requirements of the solution. Each content management style provides many options and techniques for supporting specific requirements. It is likely that you will have to modify and extend your topology as new business needs are identified. The storage, ingestion and retrieval characteristics of your content will be a major influence on the system topology.

The volumetric data that is gathered at this stage should be used to validate the transactional load, the performance characteristics, and the resulting topology required to meet that load. It may be necessary to either simulate or perform focused testing of the data in order to ensure the topology provides the desired performance. A performance modelling tool, known as SCouT, is provided by IBM in order to generate a suggested performance model based on the transactional data provided. This will not be an exact model, as there is likely to be variance in the transactional data, and assumptions and sources of error will also be introduced. It will, however, provide a broad estimation that can be used to validate topology requirements. Where more precise data is required, or where the modelling technique would be too complex, a targeted proof of concept may be conducted to provide specific topological performance.

Phase 3 – Content auditing and reporting requirements

Audit and reporting requirements will alter the classification design with the content solution. Where specific data is required for analysis, then this data must be present within the content management data. The audit information required to meet legislative or business requirements must be considered when designing classes and data that is stored within the system. For example, to be able to analyse the content stored within the system by a country or jurisdiction, then this data must be present within the system.

Audit data is collected as the system performs actions, so it is necessary to configure this design and implement this behaviour prior to the ingestion of any content. Failure to do this will result in missing information in the audit logs, which may invalidate any further data. It is extremely difficult to pre-load or modify audit information, so the validity and content of this data must also be understood as part of the solution design.

It is likely that the identification information determined in phase 1 will be the major focus for retrieval of audit information. Other data elements, such as the involved user, date and time of action, and the actual action performed are pre-defined elements that the system tracks. The events that must be audited and the data that is required for each must be determined and configured.

The audit attributes may cause changes in the class structures. These attributes will vary for different styles and types of content, and therefore will require deconstruction, analysis and reconstruction to minimise the configuration and maintenance effort within the system. It is likely that separate classes will be required where different audit requirements exist. Where these audit requirements can be grouped into similar behaviours then class amalgamation may be suitable. The goal is to minimise the number of unique behaviours in the system, whilst ensuring that any global behaviour is not too generic to be practical.

An awareness of the process audit requirements must also maintained whilst performing the analysis of content audit requirements. Much of the audit requirements of a solution are related to the actions taken whilst processing work, rather than the low level events that occur to content within the process. It is likely that high value or high security objects will have content-based audit requirements, while most other content will be audited through the processes that utilise the content. Neither content nor process auditing can be considered sufficient in isolation; both are nearly certain to be required in order to meet business requirements.

Where both content and process audit requirements are identified, then the identification and reconciliation requirements must be considered. Data elements that relate the content and process audit records must be defined so that the two separate audit environments can be amalgamated.

It is also likely that many of the reporting structures related to content, such as chargeback or utilisation management, can be handled through either systems management software (see FileNet System Manager) or through process based audits. The content engine and process engine have different data structures for auditing, and amalgamating these can be problematic. Ensuring that a specific audit purpose is implemented by each separately will minimise both effort and the requirement for amalgamation.

Phase 4 – Content Creation, publishing and lifecycle

Dynamic content has a much larger system requirement that static content. This phase deals with the analysis of the lifecycle of the content, identifying events that change the content, and techniques for managing the changes to, and availability of, dynamic content.

The attributes of dynamic content that should be considered are:

1. Content creation. The location, tools and interactions needed to ingest content within the system. Different styles of content may be ingested using different techniques such as scanning[1], Workplace, Office Integration, WebDAV, Records Crawler or customised applications. Creation characteristics impact the network requirements, user interface, auditing, security, deployment, process initiation and many other elements of the design. The earlier phases of the design should provide a constrained view of the creation events and some guidance of the areas to be analysed.

2. Content modification. Dynamic content can be modified. The behaviour undertaken to modify the content, and the results of the modification, must be understood. It is likely that different aspects of content can be modified and that many different results are possible. Each should be defined and understood.

   The areas to be considered are: the attributes of the content, the physical content of the content, and any annotations made upon the content. Each of these has impact on the class structure, the processes that use or are initiated by changes in the content, the audit requirements and the accessibility of the content over time.

[1] It is assumed that the reader is familiar with each of these applications or ingestion methods. Further information on each of these capabilities can be found within the ecm_help provided with the P8 platform

3. Content access. This portion of the analysis focuses on separating the creation and modification requirements from those purely needed for retrieval or access. This portion of the analysis is deliberately performed separately, as the publishing and caching models implicit in the platform can be used to meet different access requirements. The publishing model supports different renditions of content being made available to different audiences, in different formats, and with different security. The caching model supports minimising transmission times and will be dependent upon the topology of the users of the solution and the systems that are deployed.

4. Records management. The requirement for content to be managed as records should be considered at this point. The events in the lifecycle of a piece of content may be linked to declaration and disposal actions within the records management component of the software. It is preferable to identify the behavioural events early in the content design, but the content design should not specifically change to deal with records requirements unless the content design is specifically for a records management solution. Content design for document management and content design for records management are discrete activities. This is because document management focuses on managing the documents used in business processes, whilst records management focuses on the retention and disposal of large groups of objects. At most, the events that will cause records behaviours must be identified.

As with the earlier phases of the design process, the goal is to identify areas of commonality in the analysed data. It is likely that the events that occur for many types of content will be similar, leading to the amalgamation of class structures or a more generic implementation of what may have been specific in earlier phases of the analysis. This would require separation or subclassing of the behaviour where specific characteristics are obviously unique.

Typically, the major events and the data that should be considered in this area of lifecycle are:

1. Create. At this point the attributes that are necessary to identify and initiate any classification or processing activities should be understood. There are likely to be data attributes, content elements and security behaviours that are known at this point. The tools used, and the user or system interaction required, to create content will be defined. The analysis of some elements should be relegated to the following phase (security lifecycle) or to the process methodology (process initiation).

2. Checkout. The checkout and checkin behaviour are linked, but analysed separately. Checkout may be restricted to certain users or, in the case of static content, unavailable. Audit requirements and content retrieval performance should be reviewed. Checkout is only possible for document classes, which may impact class selection if checkout of data only attributes is needed.

3. Checkin. It is typically only possible for the checkin of a content object to be performed by the specific user that has checked out that object. The audit requirement is normally attached to this event rather than the checkout event. Notification and process initiation on this event should be documented. The versioning behaviour that is required (major versus minor) should be understood, and any related separation of processes that are initiated or security attributes that vary due to versioning need to be considered. In a publishing model, this event may initiate publishing processes (major or minor version may be separately relevant) that create alternate renditions of the content for distribution. The publishing model used may simplify any security requirements related to checkin, security changes, or alternative user groups.

4. Update. Update describes a change to the attributes of an object. The security model must support changes to attributes or they will not occur. The definition of what changes are permitted, rather than under what conditions changes are allowed, is the focus. The definition of under what conditions changes are allowed is the focus of a later stage.

5. Annotate. An annotation is an object that is attached to a content object in order to denote a change to the content object. There is no specific event tracked in the system for the annotated object. Each annotation is an object in its own right, meaning that it can have events, audit and security attributes that need to be understood. Annotations are possibly dependently secured based on the object it annotates, which may impact the security or audit requirements.

6. RM Audit. The records management model implements specific events related to auditing of records behaviour. These should be considered when designing the records behaviours and are not relevant for document management solutions. The RM Audit behaviour covers specific behaviour for Update and Delete events for record objects.

The analysis of this data set will provide information to support the selection of the correct storage and caching topology to ensure system performance. It will also indicate the tools, systems and integration approaches that should be used to support the lifecycle of the objects.

Phase 5 – Content security and lifecycle

The security of any content object is not static, and will change over the lifecycle of the object. It is important to analyse the events and the security behaviours they imply. Security lifecycles are analysed at this late stage in the design deliberately, as they complicate the content and process analysis if they are performed at the same time as other phases of analysis. Security requirements are separate to, but impact on, most other areas of content design. The security model of the P8 platform is complex enough to meet many different security requirements, and the selection of the correct techniques to use to implement your security model must wait until earlier phases are complete. This will simplify the framework for extracting the security attributes as much of the classification, identification and lifecycle of the objects will now be understood.

Security attributes are rarely considered globally, and are typically based on separate application requirements, or are bound by the current business practices for managing paper-based content. Most manual or paper-based content security is limited by the existence of a single non-distributed version of the content (ie the paper), whereas a content management solution makes the content available to all that have access to it. This phase starts with assuming that content is available to all, and only restricts security when specific requirements are known. In many cases, security restrictions that are described in business requirements are extreme, and are based on the concept of content being 'mine' rather than on content as a business resource that must be managed. Any approach that starts with constraining access based on ownership will result in a very complicated security model, and will likely create maintenance issues and result in a poorly performing solution.

A fine-grained security model is available within the content engine that provides many different techniques for implementing the total security model. Different aspects and techniques of the model will be used to meet each analysed security requirement. The capabilities of the P8 security model must be understood prior to this analysis[2].

This phase looks at the same events that were defined in phase 4, but focuses on the security attributes that are represented within each event.

[2] Refer to P8 Security Model within the ecm_help provided with the P8 platform software.

1. Creation. The ability for a user or system to create a type of content must be configured. If specificity is required, then new classes or subclasses are likely to be needed. Classes may be amalgamated where general behaviour is needed. The desire for audit and process initiation (as described in earlier phases) must be used as an input to determine the creation security characteristics of each class.

   Default Access Security is defined on each class within the system and is copied to any instance of that class when it is created. This configuration ability may cause classes to be separated, subclassed or amalgamated at this point in the analysis. The default security alone may be sufficient for objects that do not have security lifecycles, and where a known starting security state exists.

2. Checkout. Checkout requires specific security permissions and the source and management of the permissions should be defined. There are property dependencies that must be considered as checking in an object implies the ability to modify some attributes of the document. Checkout should be configured for objects where Checkin behaviour has also been defined.

3. Checkin. Major and minor version behaviour must be defined. Security can change when using security policies through the "reservation", "in process", "released" and "superseded" version states. As a document moves through these states the security of the document can be automatically changed the system. The versioning behaviour that is required (major versus minor) should be understood as specific security attributes are required to allow each separate style of checkin.

4. Update. Whether an objects properties or content can be modified must be incorporated into the security model. It is possible to separate the ability to modify the attributes of an object from the ability to modify the content of the object (documents only).

5. Annotate. An annotations is usually dependently secured based on the object it annotates, or it may be secured through its own access control list. Identification of these characteristics is necessary for each class of object that is permitted to be annotated.

6. Delete. The ability to delete objects must be considered. Some system objects (eg. process definitions and search templates) should not be able to be deleted by users of the systems, whilst other documents, such as those created by a user, may need to be delete-able. Deletion is not allowed within a records management context, and the transition when a content object is declared as a record should consider this resulting behaviour. Records management implements a specific disposal behaviour that is defined and managed separately.

   Deletion and availability of content can also be affected by the caching topology. Secure information should be reviewed for desired behaviour, as cached copies are not deleted when deletion of an object occurs.

7. Administration. Normal users are provided access as defined in the security model. Administrative rights are permitted at the object store level. Any requirement to restrict access to administrative users must be defined. The users that are permitted to administer content objects should be considered and documented.

8. Ownership. The creator of an object is typically allocated ownership rights of that object, which includes the ability to delete the object. This should be analysed where ownership is needed for security behaviour, notification, or lifecycle events (which may require return to phase 4).

Access Control Lists

Access control lists are applicable to almost every object stored within Content Manager. It is best practice to consider the security on the configuration objects within the system, not just the content being managed. Each class definition, for example, can be secured to ensure that only authorised users have the ability to make changes to the definitions within the system. This analysis should be considered as part of the deployment planning for the solution. It may be an output of some of the security analysis, but should not be pursued at the same time as this may further complicate the required analysis steps.

A technique to simplify the security model within any system is to provide a gross level of access control on a large portion of any individual application. This is possible through access control lists that are applicable on entire object stores[3]. For example, it is possible to restrict access to all human resources or board documents by placing them in a separate object store and constraining the users and groups that have access to the object store. The model inside the object store becomes simpler as only those that are granted "use" permissions on the object will be given access to any content within it. Inside the object store the special users "#Authenticated Users" can be used with less constraint as only those authenticated users that are given access to the object store will be considered. Care should be taken in this model. Even though this behaviour simplifies access control within the target object store, it is an exclusive behaviour. It doesn't promote sharing of information and should only be considered where this is the desired outcome.

Records management behaviour implements an additional security model that should be analysed in its own context, and then amalgamated with the content management security attributes of the solution. When a content object is declared as a record, it gives up all of its current security behaviour and takes on the security behaviour of the records management object it is associated with.

Security behaviour can be directly assigned to any individual object, assuming that there is not already security behaviour in place on that object prohibiting this. In this case, where specific security behaviour is needed, specific process based coding should be implemented. A complete description of the aberrant security behaviour must be performed and the reasons for the designed implementation must be documented. It will be critical to future users of the solution to understand the reasons for the approaches that have been undertaken in this respect.

The security attributes that have been determined will mostly separate classes or determine subclassing behaviour. The class and metadata structures created in earlier phases may be required to be restructured in order to enable them to be subject to the desired security behaviours.

Auditing requirements may also be impacted by security requirements, as in many cases it is sufficient to implement security through auditing rather than making restrictive rules in the security model. In this model, rather than stopping actions through the security model, the system tracks and notifies relevant users or processes of any attempted actions that may be suspect.

[3] It is assumed the reader is familiar with the P8 security model and with the structures that make up the content engine.

Phase 6 – Content storage, retrieval, delivery and access

A clear and defined content requirement should exist at this stage of the analysis. The form, style, classes, data, interaction, security, audit and process integration points within the model should be clear. If this is the case, then the model for storing, retrieving, delivering and accessing content should be simpler to define. Each exception to any existing model can be simply seen, and each requirement that may impact the model should be known.

The system behaviours that require configuration in this phase are: the storage areas, content cache areas, index areas, file stores, and storage policies. Several default and pre-built behaviours will exist and this phase will focus on identifying:

1. What are the quantitative characteristics of the content (performance requirements, creation frequency, volume, retrieval frequency and distribution, by geography and content type).

2. Where will content be stored (object stores and storage areas, fixed content devices).

3. From where will content be retrieved (object stores, storage areas, content cache areas, site topology).

4. Where will content be cached for each retrieval (content cache areas, site topology).

5. How will content be indexed (object stores, content based retrieval – CBR).

6. How will content be ingested (tools, storage, format, style).

Each of these areas should be identified based on the characteristics that have been analysed earlier. The physical topology of the solution can then be identified and implemented to meet the characteristics of each individual requirement. It is likely that different behaviours and configuration activities are needed to meet different requirements. There will be no single topology that meets all requirements, but rather many individual features of the P8 architecture will be involved for each individual requirement.

The physical topology may modify the class structures where specific data or class behaviours are needed to meet the topological requirements. Some attributes of the system are class specific and some are driven by attributes of objects. A change in class structure may be required to meet the topological or deployment characteristics of specific content objects. Similarly, a change in deployment may be needed to meet the characteristics of specific content. For example, a cache server may be required to cache imaging traffic in a specific location, or a subclass or class attributes may be required to redirect the storage requirements for a specific type of document object to a specific geographical location to meet performance requirements.

The volume characteristics must be understood before the system topology can be finalised. It is desired for this level of information to be gathered in the early phases of content analysis.

Phase 7 – User Interface

The user interface, or sequencing of actions on a users desktop, cannot drive the configuration of the system. The lifecycle of the entire content management system is far wider and longer than that of any application that uses the content. Multiple separate applications will utilise the content, and each may have its own build language and development characteristics. The user interface to the system must be separate to the content structures that support any application.

For this reason, the user interface is relegated to the latter phases of this methodology. Whatever user interface is required can be designed once the content (and process) interactions are defined. The content should be designed to remain in the system for many years and to support many different users and applications. The user interface of any single application cannot be used for any purpose other than as a data point that may provide some indication of the data that is required to be stored within the solution. The sequencing, completion and validation of any data is separate to the storage, processing and management of any content (or process).

The user interface design may impact upon the content and process design, but it should not proscribe the design.

Phase 8 – Coding Specification

Once all possible configuration behaviours have been defined, then specification of any coding requirements can be completed. Some coding specification may occur in parallel to the content design where specific requirements that do not change the design activity can be isolated. No database or user interface design should occur until at least some portions of phase 4 and 5 have been considered. Coding, database or user interface design that occurs prior to phase 2 will not be representative of any real content topology, and will likely generate siloed applications.

Best practice implementation of code for content management involves incorporating any code as discrete segments of functionality embedded within the content engine. The content engine is able to invoke code and processes as events occur within it, and this model should be used to ensure that no application specific code exists that is external to the content engine. In this way any application, regardless of its development model, will incur the same behaviour within the system for the same action taken. The system will respond to your action, by invoking the code and process segments that are defined regardless of the source, format, application, location or access technique that is used (unless specifically defined).

This approach promotes global content behaviour, ensures reuse through embedding, and simplifies ongoing maintenance and expansion of the system. It does, however, require a consistent and controlled design methodology that is well known and promulgated.

Coding itself may be multi-phased, and iterative, where specific requirements that come from any phase can be passed to development staff. Design and development can be multi-phased, where phase 1 through 8 are done once and an initial coding specification is created which occurs in parallel with the revisiting of each of the phases of this methodology. Coding should not commence until the outputs of each phase have been considered in some form.

No specific coding tool or behaviour is needed, other than the constraint that the content engine only supports Java based code for embedded functions, whilst access to the content engine from external sources can be via .Net, J2EE or other web services techniques as described in the content and process design documentations provided by IBM. There are no assumptions made within this design methodology based on code development, and coding specifications should follow standard organisational models.

Pre-existing and externally sourced code should also be considered within the content design activity. There will be libraries of code modules developed and available from third parties and within the organisation. These must be managed and publicised to ensure use and reuse.

Process Analysis

This methodology defines a process as the actions that occur to complete a piece of work, whereas many businesses define processes as what the work is, who does the work and how the work is allocated. When the latter approach is used, business processes tend to represent the organisational and operational behaviours of the existing systems, and rarely deal with the real activity that occurs.

When viewed at an abstract level, the actions taken in performing work rarely change, but the sequence and conditions under which they occur are the business value that organisations deliver to their customers as differentiators. By focusing on what is done, rather than who does it, the process designs are not dependant on the delivery method, the people, and the organisational behaviour of the business. This supports process change and is the goal of the methodology

Process Style

Different process characteristics require different techniques for implementation, but the activities involved are similar regardless of the process being designed.

Specifically, four different stages of process modelling should be considered. Each stage has a slightly different baseline of technical requirements and process complexity. Each may be impacted differently by content interaction.

These models can be considered stages in process delivery, as the subsequent model is an extension of the earlier model. In some cases, this is irrelevant, for example a storage and retrieval model may never need to grow into a larger process-centric application. In these cases only a single model will ever exist within a single application space, but when considered as part of an enterprise solution, each must fit into the consistent enterprise process architecture, as each element of any application may become re-used by other applications, or become a data source for subsequent applications.

The four stages of process modeling are:

1. Work routing and tracking. This process model involves delivering and tracking pieces of work, with or without associated content. The work items that are delivered to users interact with process and possibly content data. The goal of this model is to deliver work to users and to support tracking and reporting of work that has been processed.

   The process definitions in this model tend to be simplistic and rely on user decision for routing paths rather than embedding of business logic into the process models. Some integration with other systems may be performed.

Initial imaging and content management solutions that are used for storage and retrieval or process initiation fit within this category.

2. Case Management. This model involves delivering and tracking pieces of work, but typically is triggered by, or involves, content. It also involves a more complex process definition where some elements of business logic are represented within the process models. It mostly relies on user decision for routing paths, but also implements business logic and system integration within the process definitions.

   This model is typically implemented using the Business Process Framework ("BPF") product as it provides many of the services and configuration abilities necessary to deliver this model quickly.

   This model extends the work tracking and routing model by more complicated matching, collection and auditing functions.

3. Process Services. This model involves short term processes that deliver specific functionality to other systems. The process definitions tend to be non-interactive and perform integration functions on other application data. At most, approval or review functions may be included in this model.

   As the processes are short-lived and simple, it is likely that they will be widely applicable as they are discrete in their capabilities.

   This model is typically both a technique for implementing re-usable processes or process segments, as well as a specific style of process. The processes are likely to be specific to an integration requirement or a business activity. The processes are also likely to be widely implemented and re-used for other applications to use the functionality that is delivered.

   Many records management and content creation processes fit within this category as they rely on short-lived processes to perform specific actions. In records management the disposition processes are all short lived phases created as needed, driven by the changes in state of record content. Content creation uses short lived processes to manage the state changes that occur in the lifecycle of a document. Both of these models rely on the content engine for the storage and initiation of the process actions that are linked to specific processes.

4. Process Centric. The process centric model is the long term goal for any process management system. The processes are more complex, but implemented as discrete process segments. The processes interact with other systems and a wide range of other processes to deliver and manage work. This is a progressive model as there is a large amount of design, development and integration required to break existing business practices into re-usable process segments. As process services are implemented they can be joined into large process structures that implement wider business processes.

This model is the most dependent on re-use, and must be architected into the earlier models to ensure that process services and other simple work delivery models fit into the final process structure. Some strategic decisions taken in earlier models may require re-work as implementation teams learn more about the real process delivery and grow the base of process segments that are available for insertion into wider business processes.

Process centricity comes from a design based on processes, rather than from application functionality. Any individual business application comprises segments of process behaviour, and these segments must be identified and implemented so that they can be altered or supplanted without breaking the business functions that they implement.

These processes also implement structures for supporting business intelligence, real time monitoring, security and auditing in a much broader model than the earlier models. As the processes are an amalgamation of other processes the tracking and audit data that is generated by all processes must fit within a consistent data model. The final result from the tracking and audit data can then be viewed at any level of the process structure. Low level processes and complete business processes can be analysed, tracked and simulated to support process improvement.

Earlier models provide tactical benefits to the business, but this final model delivers process agility, the means of outperforming competitors by rapidly dealing with business change.

Process Commonality

One of the major benefits of process management is the consolidation and management of information related to disparate processes. Process re-use and process separation are goals that should be attained in order to maximise the effectiveness of a process-driven enterprise.

Process re-use allows portions of or entire processes to be used by other processes. The design criteria for the processes need to take this goal into consideration, because data and integration specificity will compromise the ability of the process to be used by others, in ways that may not have been apparent when they were originally designed. Process re-use doesn't solely mean re-using process as it is, but through extension and overloading of different segments of any process, a different implementation could be derived.

Process separation is a requirement for re-use to be possible. Discrete functions within a process must be identified and designed so that areas that are unnecessary for other business use can be modified, whilst maintaining the portions of the process that are relevant. For this to be possible, process segments must perform specific functions and must be considered as entire processes by themselves, invoked and used as needed.

These two elements of process design have other constraints and benefits to a process-based organisation. Commonality of data and process components is necessary so that a single analytics capability can be implemented. The techniques for locating, processing, auditing, securing and analysing processes are common for the entire system. This must be considered during the design phases so that extraneous solutions are not needed, and a holistic view of the process management system can be obtained.

Activities and Outputs

Phase 1 - Business Logic and Process Segmentation

The purpose of this phase is to extract the business information from the user community, validate it and to show an initial model of how process management works. The knowledge of the business processes is represented differently in the minds of any user.

The diagrams that are often produced to represent a business process are rarely focused on the requirements of implementing the process in a process management system. The process definitions tend to be high level, focused on who does the work and tend to be self-focused, dealing with the scope of the current process only.

This phase uses any diagram or business knowledge purely as knowledge. Different process management systems use different terminology and process modelling software rarely deals with the implementation requirements of any specific process engine. Even commonly used modelling notation such as Business Process Modelling Notation (BPMN) focuses on abstracting the process behaviour, and not the implementation structures. Even if it was extended to define the implementation technique, an implementation specific diagram would be created, which would likely require application specific technologists to work on the design.

The approach of this phase is to take whatever information is available in whatever format, and to extract the business logic only. Most modelling designs focus on user interactions rather than business logic, and rarely extend to define the large number of exceptions or time constraints that exist in any business process. In many cases, the process designs purely define the straight through process.

Once a process is implemented within a process management system, then there are no extraneous process paths and there are no options for doing something other than what is described in the business process. The process definition is the complete diagram of everything that is done, and there are no other paths. Designs that don't consider this invariably fail, as users are unable to do what is needed to complete their work. The work tracking and auditing that result from the system is unrepresentative of the actual work practices, and most benefits likely come from the removal of transition times due to content management, rather than from any process improvements.

This first phase extracts the major business paths that are possible within the target process. It doesn't deal with any data, user interface or integration requirements, and purely focused on extracting the rules and decisions for moving work from one activity to another. The derivation of queuing or work movement is not the focus of this phase, but purely the actions that are performed on the work and the sequencing of those actions.

It is imperative that the business process is understood by the person or team that is building the process design. The business process will be documented in a format that is required for business level understanding of the process, and will not be suitable to direct translation into the process design tool. The business analysis document represents the business view of the process, whereas the process design, in its final version, will be the implementation of this process in a much broader scheme. The process design will encompass sub-processes that are used by other processes as well as data and conditions that are more complex than the simple business oriented rules normally described in analysis documentation.

For this reason, it is important that both business analysts and process designers are involved in the process design. The process designer tool involves both process and programmer style constructs. Programming constructs should be encapsulated and avoided in main business logic processes. The business analyst must understand that these constructs will evolve to support a much broader context than that of the current business process. The process designers must understand that major portions of the process design must be targeted at business users, and not process designers. Users must be able to understand the broad behaviour of the process from the process maps that are designed.

A simple design goal at this stage is to ensure to align the business process knowledge with the process design requirements. Many business processes can be broken into 5 to 7 broad phases that occur in the process. For example, something is received, it is reviewed for correctness, it is processed, it may require approval, and it is completed. These five phases form the basis for many processes such as claims processing, loan approval and correspondence handling.

The intent of this phase is to translate the process definitions at a high level from whatever tool or technique has been used to capture the knowledge of the process. In most designs, a business process should be summarised so that there are approximately five steps on a process map. No complicated processing or detail is included in the process map as this initial step is to prove that the desired, missing and exceptional paths of the process can be validated. Five steps is a guideline that works purely as people tend to think, rationalise and remember in groups of five as a maximum. The more steps there are on a single map the harder it is for someone to understand its purpose quickly and concisely.

Too few steps on a map reduce the flexibility of the process for future change, as the process becomes very specific, due to its inherent lack of scope. For example, placing a single step on a map merely generates more effort, without any benefit to the process. Where this is documented and performed for future expansion then this approach, however, becomes valid.

Too many steps on a map reduce the flexibility of the process and impedes the understanding of someone determining what function the process delivers. Steps that have many branching paths, complex rules or complex dependencies should be isolated within their own submap.

There are no hard and fast rules for determining the granularity of any object, only guidelines that are adaptable as required, as long as all assumptions are documented to assist later users and designers of the process.

At this phase, the paths have been labelled with why they are chosen, but no implementation specific conditions or routing options have been used. This graphically represents the process paths and should allow someone to see that all relevant paths are described.

The following basic principles must be followed in order to maximise the re-use and clarity of the process extraction:

1. No more than five real activities should be present on any process segment. People can immediately identify up to five things of the same type without requiring a great deal of mental processing. Keeping the number of activities on a process segment to a small number allows people to readily understand what the process does, without requiring further analysis. There are cases where empty steps and start and end placeholder steps increase legibility. Steps of this type are not included within the "five". System steps, queued steps and other activities that represent what is done are included in the 'five".

   In this model, five is a guideline, that is not absolute. Seven steps can be acceptable where a complex process is being modelled. Nine steps would indicate poor deconstruction and granularity. An objective decision must be taken, but if the guideline is followed then it is far more likely that reusable and understandable process designs will result. Across a larger set of processes the average number of real activities present per process map should approximate five. The more this deviates in a solution, the less likely it will be able to be extended quickly in the future.

2. Each activity on the same process segment must of similar granularity. Activities that are present on the same process segment must be related within the same segment. They must be related by similar means and focus. Actions must take similar time; have the same complexity and likely deal with a similar level of detail. There must be no association or dependency between each action as no queuing or sequencing behaviour is considered at this point. The analysis and design must focus purely on what is done in any activity, and not deal with who or where the activity is done.

3. Each activity must be described concisely to the same level of detail using a common vocabulary. The purpose of process design is to support changes to the process. No ambiguity can exist in the definition of any activity as it is likely that other designers will be user the processes in ways that are outside the scope of the current design process. Terminology must be consistent, concise and from a common knowledge base within each process. Activities cannot be named "Approval" or other simplistic terms. This provides no information to anyone that may use this activity in the future, and leaves ambiguity for the current team that may be implementing the activity.

4. Naming conventions for content and process attributes must be adhered to. These naming conventions can be found in the SCB ECM Naming Conventions document.

5. Each process segment must be discrete and must be self-contained. The data attributes and rules used to determine the paths within the process segment must be of similar scope, in that they should be consistently referenced within the same level of the overall process structure. Where a data value is used to drive a decision in the business process it should be consistently used within the same level of the process structure. As all data fields are global to the process, they should not be used to control actions either up or down in the process stack. Recursion of process segments is a much more efficient and readily modifiable means of implementing this behaviour. Where a higher level process is needed to execute based on a lower level outcome then the higher level process should be invoked directly from the lower level process, encouraging discrete, reusable processes at all levels.

6. Process segments must be balanced in the overall structure. The process structure is a two dimensional view of the process segments and how they interrelate. As a process segment is invoked by a higher level segment, it is represented graphically as a lower level in the process structure.

If the rules for granularity and step counts are followed then a balanced process structure should result. Each lower level should implement steps of similar granularity and similar scope. As the number of steps is also similar then the overall view of the process structure should be a graded curve, expanding exponentially in a balanced and organised structure.

The top level should have around five activities, each of which should decompose into around five activities, where each of these should decompose further into around five activities. For the structure to be balanced, the granularity at any level in the model should be similar and the number of activities are each level should follow the equation "five to the power of level – 1". This can rapidly lead to a complex and large structure.

The scope of any business process will determine which nodes within the map hierarchy will be decomposed further. As more detail is required, then submaps are used to provide the detail of the activities are being decomposed. This implies that portions of the hierarchy will be deeper (ie more levels) than other areas. This is normal, but should not alter the high level "curve" structure of the process hierarchy. The steps at the high levels must be of larger granulary, and as lower levels are designed, they should be of smaller granularity, effectively taking up less space graphically, but expanding the base of the curve.

Typically, the process is stopped when the desired level of granularity is reached. The third level (125 activities) of granularity is a reasonable target for business processes. The fifth level of granularity approaches programming specifications, and is rarely needed. When this level seems to be desired, it may prove that more than one process is actually being decomposed and a separate decomposition exercise should be undertaken (i.e. you're actually deconstructing a much higher process that itself is made up of 5 major process activities).

When considered at a higher level in an organisation, there are many more than four levels of process deconstruction. Good process design will support processes from different areas of an organisation being reused in others, effectively placing another abstract process level above what was thought of as the top level of the two, now, sub-processes. A process centric organisation is a target for an enterprise process management system. Each new process must add to the process capability of an organisation, and it should be designed to do so.

> This phase of deconstruction does not proceed to the final level of granularity, but stops after each major portion of the process has been deconstructed. At this point, the process is executed moving a piece of work through the defined business logic ensuring that the design matches the expectations of the designer and the business users. Once each process segment is agreed to represent the process, then the following levels of granularity can be deconstructed, adding further detail to the process, all within the agreed framework created by the parent higher level process.

This approach provides a framework that promotes reusability as long as the general principles are followed. A common vocabulary ensures that process segments describe themselves in a similar way and that other designers and users will comprehend what the process implements. The artificial limit of five activities ensures that process segments are easily understood and that complex and likely clumsy logic is stopped before it gets into the model. Large numbers of steps on a single map promote sloppy thinking and invariably lead to "moments of magic". Moments of magic involve disconnected activities that have no real linkage or have no specific detail involved. They don't add value and confuse and complicate the process segment. Activities with names like "Approve" or "Print" are indicators of magic. They don't provide sufficient detail for people to understand what they do and they're likely a step of such scope that the really need to be represented by a sub process. Top level processes should start with this level of granularity with the implicit assumption that they will be further deconstructed. They should always, however, be named in a sensible fashion such as "Credit Application Approval" and not "Approve".

As each level is deconstructed, further detail is added to the model, and the model is executed, it is likely that the model itself will require change. Each change should be considered in the same framework and must represent the entire business logic at the same level of granularity within each process. By keeping the process segments discrete and self-contained they can be rapidly re-linked into different process configurations without issue. This is a goal of the process design so that future designers can do this to meet their business requirements.

If it is difficult to re-link the process segments into other business processes then the granularity or specificity of the process segment is likely to be incorrect. In these cases, the process segments may need to be named more specifically to ensure that future designers understand that this is your intent. If this is the desired behaviour then the process segments must be revised to ensure that they do have the same granularity, the same naming topology and do not rely on data passing for determining their own process paths.

This phase is the critical determination of the paths through the business process. It must represent all of the process segments before implementation, but may not necessarily represent the entire process before the next phases commence. As described earlier, the final process structure must be a balanced one, but during implementation the scope of any process can be defined to explicitly exclude detail in any process segment. For example, the "Update Stock System" activity may be part of a separate project at the time of process definition. This activity remains empty during process design, a placeholder for future development. The other four activities on the process segment may be completely defined and further decomposed but this single activity isn't complete. This is a valid process structure through defining the scope of the process, even though it isn't a complete structure in itself.

If process segmentation has been successfully performed, and process segments are of discrete and manageable size, then they will promote re-use across other applications. This isn't a mandatory goal of this first phase, but if as skills grow in process design it will be an outcome. It should be a desired outcome in the future.

Phase 1 will be revisited later in the process methodology, as each phase is recursive and iterative. As details are added in later phases, changes may be needed in the business logic. This implies that phase 1 will be revisited due to changes made in later phases. When this occurs, the following phases must be revisited. For example, if phase 4 of the analysis requires a change in the output of phase 1, then phase 2 and phase 3 must be performed (to some degree) before returning to phase 4.

During the execution of the business logic, changes will be required in the data and process definitions. These should focus on the requirements of the business logic only and not on any later phases. Where data references (Phase 2 analysis) are readily determined they can be added in phase 1, but it should not be a target of phase 1. Phase 1 is focused on proving the business logic and the data that is required to support the business logic. Introducing phase 2 data at this point can complicate the extraction of the business logic and misdirect the purpose of the execution of the process. Experienced analysts and designers can do this, after some level of business logic has been proven.

All paths in the process should be tested prior to moving to the phase 2 data referencing activities. The simplest methodology is to firstly test the straight through path, launching the process and bouncing the piece of work from step to step on a single workstation. This will allow the data that is used and the paths that are followed to be reviewed and changes to be made. At each point where paths branch into alternate paths, then the branch point must be validated. It isn't necessary to validate every convoluted path within the process, but merely to test the positive and negative options at each branch point. In most cases, it is better to fix the problems immediately as they're found rather than accumulate a list of problems. Errors in earlier steps that are not rectified may cause erroneous behaviours later in the process. By fixing them early and re-executing the process, the number of error paths will be significantly reduced.

Phase 2 – Data Referencing and Retrieval

Additional data can be added to the process model once the phase 1 business logic has been represented and proven to be indicative of the required business process. The output from phase 1 will be a process model that contains business logic and data values that control the business logic. This next phase looks at expanding the data that is contained within the process to support locating, identifying and referencing data within the process and in external data sources including the content engine.

The data that is added is used for several purposes. Firstly, data is added to identify the processes within the system. Data such as customer identifiers, names, policy numbers are used to allow people to see what a piece of work involves as well as support finding pieces of work. For example, when a call centre is involved in locating a piece of work for a customer, the customers name may be more relevant than an identifier automatically generated by any systems.

Secondly, data is added that permits the piece of work to reference other data sources. Customer identification numbers, account numbers and similar data are rarely used in logic for determining paths in the business logic, but are used to both identify pieces of work and to assist in updating or creating data in other line of business systems. The process engine should never be considered as a database system and is not architected at managing large quantities of business data. It is designed to manage process data which in turn will support creation, retrieval and update of data in other systems.

As each additional data value is added to the process, the same execution model as described for phase 1 is undertaken. The process is launched within the system and bounced through the steps that have been defined in the business logic. In this execution model, the focus isn't on the business logic, but on the data that is presented to any specific step in the process. The identification and reference data must be available at the relevant steps in the process to support what is needed to be done at the step.

As the process design focuses on "what is done to the object" and not "who does it", the name of the step will indicate what is meant to be done. The data values that are presented at each step must be sufficient to support what needs to be done. If data elements are missing they must be added. If too many data elements are provided, they should be removed from the step.

This execution process does not deal with any data validation or presentation techniques. It only deals with validating that the required information to support what needs to be done has been provided.

One consequence of the execution model is that data that is entered at one step is likely to be needed at other steps. The execution process must ensure that where data is needed, for example, at step 4 that somewhere prior to the execution of step 4, it has been entered. This sequencing of data is another required activity within the process validation that is occurring. It isn't sufficient to define the correct data at each step, but you must also ensure that the required data for subsequent steps is also provided.

A review of the behaviour of the process engine may assist. Within the process designer you define the data attributes of a piece of work. These data attributes are global within any single instance of this process in that they're available to the process engine as needed. Data isn't globally available at any step, however. The process designer must define the individual data attributes that are available to be read and modified at any step within the total process. This data isolation is important and thought must be applied to determine the correct data that is available at each step, for input, output and for both.

Within this phase there are also other data requirements that should be analysed and designed. As previously described, data elements that are within the process are also used identify and to reference the pieces of work in the system. Data elements are also used to inform the users of the characteristics of the work, and to support the ability to prioritise, filter and select individual pieces of work.

Where work items need to be sorted into a specific order for presentation to workers, then the data that represents the possible sorting behaviour needs to present within the work items. The data may not be used for identifying the work or for passing to other systems, but it must be present within the work items for them to be ordered or filtered.

Once the referencing data has been defined, and the process execution modelled to prove this, then the additional data elements that are needed for sorting and filtering of work items must be analysed. For example, where work may need to be prioritised by its financial value and filtered by country then these two data elements must be present within the work. These data elements must be added, and further execution tests conducted to ensure that data is available at the relevant steps when needed and that it is completed in the correct sequence within the process.

The definition of the work allocation, queuing and delivery model comes later in this methodology. Phase 6 deals with the requirements to get work out in a specific order or to a specific user. The collection of this filtering data may occur at different phases as it becomes apparent, but it will be analysed explicitly in phase 6.

During all of these process executions default user interfaces are used. No integration or constraining behaviour should be considered. Later phases in the design methodology will deal with these. It is critical that the process design is the focus at this point, otherwise the process design will be hijacked by extraneous issues and a specific process will be created, rather than a reusable one.

Two specific characteristics of the process engine must be considered at this point, Rosters and Event Logs. A roster provides an index to the work that is currently within the system, supporting users quickly locating pieces of work across a large system. Event logs provide a historical transaction log of all events that have occurred for any individual piece of work. Both of these components of the process engine require data that is used to identify the pieces of work and the actions that have occurred upon them.

As described, the roster is an index to work within the system. A roster is used to collect like pieces of work and to provide a common means of finding work across the system. For example, a Lending roster may contain references to Home Loan applications, Credit Card applications and Real Estate Valuation work items. Each has common data that can be used to locate it and each exists indexed in the Lending roster. The Lending roster has data elements defined on it such as Loan Number, Customer Identifier and Customer name which can be used to locate specific pieces of work when any or a combination of these data attributes is known.

The event log is an audit trail of what has happened to work within the system. An event log is used to collect the audit information for like pieces of work and to provide a common means of tracking what has occurred within the process across the system. For example, a Lending event log may contain audit events for the same Home Loan applications, Credit Card applications and Real Estate Valuation work items. The event log has data elements defined on it such as Loan Number, Customer Identifier and Customer name which can be used to identify the events that have occurred on specific piece of work. The Process Analyzer can use this data to extract, sum and correlate different actions producing reports across a wide range of work within the system.

The audit requirements for event logs are covered in phase 3. Some data will obviously occur in the event logs as identification and reference data will often be needed to determine what has occurred in the system. This initial behaviour rarely changes the business process or the data required, whereas specific audit requirements may impact business logic and data references as described in the next phase.

One of the outputs of this phase, in addition to expanded data within the business process, is a framework definition of data that may be required from other systems. Typically as data referencing is expanded, it may become obvious that business logic data that was used for routing purposes is actually represented by data from other systems. This interaction impacts the data that has been used to represent the business logic, the data that is carried and delivered at specific steps in the process, and the data that is audited within the system.

It is likely that phase 1 will require revisiting when this level of data referencing is reached. It is also likely that the outline definition of interfaces to other systems will be defined, where data that is known at one step must be passed to another system so that business logic data can be retrieved for decisions later in the process. For example, a step may require that a credit check is performed on a customer prior to a loan being approved. The credit check was originally a step that provided a binary outcome, either ok or not.

As the reference data is expanded, this step becomes more than a binary check, with different actions depending upon a variable outcome in the credit check. Certain variable conditions may affect the outcome of the credit check. These data values must be considered, passed to the credit check and the variable data that is returned will likely change the business process. The same data may change the audit requirements and may require that additional data is collected in earlier steps so that the credit check can be performed. This iterative and recursive design process is expected and should be encouraged.

In future process re-designs, this same model will be used, ensuring that the same methodology works for the initial process analysis and for ongoing process analysis.

In addition to traditional data types, such as strings and integers, the process engine has the ability to define and manage XML data fields, attachments and workflow groups. These three specific types of data should be analysed and reviewed at this stage of the process design.

The initial data, such as AccountNumber, will be used to provide identification and relational attributes of the work being modelled. Attachments such as a "Loan Application" or a Workflow Group such as a list of "suggested approvers" are also relevant data values that can be placed within the process engine. The extraction and analysis of this data should be performed once the business logic and 'normal' identification data types have been determined. Including these types of information in the analysis before the business rules and identification characteristics have been determined will pollute the process design.

Content analysis would be applied to this information prior to it being included in the process design. Phase 1 of content design will extract the relevant identifying data from the attachments, which may provide data to this phase of the process design. Changes in the business logic should not result from this content analysis, but changes to the data maintained within the process may result. In all cases, data separation should be a focus, ensuring that the process only contains relevant data to support business logic, data references for integration to other data sources, and data required for audit and reporting requirements (see phase 3).

It is common within a FileNet P8 environment for content and process to interact. The points of interaction must be determined and isolated as separate activities in the design process. Following this phase, content objects should be known as attachments within the process, and data that is provided to or from the content and process should be understood. The data contained on the content objects (eg Address on an application form) should remain isolated on the content and should not be reflected in the process (ie Address should not be within the process unless it is determined in a later iteration or phase that it must be present).

The identification and reference data for different processes will provide the first level of separation and amalgamation that will be performed when collections of processes are reviewed. The purpose of this review is to determine firstly the granularity of processes and secondly to determine if they can and should be either broken into smaller discrete processes or if they should be amalgamated with other processes. The simple attributes that identify the processes will rapidly provide separation of different processes, such that differences in the identifying characteristics indicate a different process. This must be done iteratively and small steps should be taken in the design process, testing if processes should be separated or amalgamated into other process. For reuse to be effective, data and process attributes should be named generally until specific requirements are determined. Each iteration through this phase will provide more characteristics that will support either the separation or amalgamation of processes, and will likely determine common characteristics that should be reused.

The reference data that is retained within the process models should provide sufficient data to interact with external systems. Some data will be relevant to the process, some to content and some to other systems. At this point in the process analysis it should be possible to identify the large scale points of interaction with external systems, and some level of function interface should be defined. This allows other resources to commence working on the test harness versions of the final interfaces, and for these to be prototyped and tested within the design process. For example, where it is determined that an account retrieval is needed to provide data for a specific step in the process, then the reference data, such as account number, will be present in the process. This data, when provided to a suitable web service, will retrieve the account name and financial details which are displayed to the user. Some of the data, such as the balance of the account, may be used to determine the routes that a process will follow. The creation of the interface and the code to support it can proceed (Phase 7) with the assumption that for this iteration phase 3 through 7 are empty. This phase can continue in its analysis, whilst other resources work on the interfaces.

As described earlier, it is not the goal of the process and content engine to be treated as a database. Both engines deliver specific capabilities for managing the data that is used in processes and for identifying content, but they do not (and should not) support complete database manipulation techniques. Data encapsulation should be a goal of the process and content design, ensuring that only relevant data is retained in the content and process engine and that all other data related to areas such as the user interfaces remain at that level in the system design. Forms that contain large numbers of data fields should maintain their data within the XML structures of the forms, and it is unnecessary for the entire data structure to be mapped into either the content or process design. XML data, and other user applications such as BPF should follow this same paradigm, ensuring that only data that is relevant to the business logic, the identification or auditing of the process is retained within the process.

Phase 3 – Audit and reporting requirements

Phase 3 focuses on the data that is used to track what has occurred within the work and to provide metrics related to the work. The goal of this phase is to support the functional requirements of using the Process Analyzer product and to meet the management requirements of the business.

Some of the identification and reference data that has already been analysed may support the audit and reporting requirements, but it is likely that other business metrics are required from the process management system. For the metrics to be available suitable data must have been captured in the process design. For example, if the business measures the performance of the staff based on the number of applications that are approved, then the granularity of the work must represent the count of applications, and the status of the work related to it being approved must be captured somewhere within the process.

There are also cases where the audit and reporting requirements require additional data to be captured during a step in a process. If it is necessary to analyse the work volume by postcode, for example, then the postcode data should exist within the work and its value must be provided prior to the point in time when the analysis data is recorded.

This phase expands upon the data that is carried within the processes and ensures that the data that is required for auditing and reporting each piece of work is provided and available when needed.

The prototype behaviour is identical, though the use of the Process Tracker, Process Analyzer and Process Administrator is recommended to follow the work processes, and validate the correct data is being recorded and available for analysis.

Within this phase, milestones are also introduced. The audit and reporting requirements include not just those that are related to the business implementing the processes, but also to the users or customers that are the reason for the process. It is important to contain data that assists in the tracking, tracing and auditing of the work, but it is also important to track data that is related to informing the users and participants of the process of its status. For example, it may be important for the customer that originates a request to know its status, but it is rarely sensible for the customer to be provided a graphical process map. The neither care about the internal processes, nor understand the design of a process map. Milestones allow business specific data to be encapsulated and provided to different consumers simply.

It is important to validate that the data that is being used in reporting activities is relevant and appropriate. Each event that occurs within the process engine will generate a row in the event log and this table can grow rapidly if inappropriate data is exposed to the event system. A balance must be achieved to ensure that data that is used to identify, relate and contextualise the audit data is kept, but that this doesn't become an overhead in the system performance. Anecdotally, the event management system consumes less than 7% of the CPU load that is generated by the entire process engine. The amount of data, however, can be significant.

The audit and reporting phase must focus on the data that should be tracked for audit and reporting, and not deal with any security or work delivery issues related to access. These attributes will be analysed later in the methodology.

It is important that phase 3 is fully tested to meet the business requirements before phases 4, 5 and 6 are completed (except where phase 4 through 6 are explicitly empty and interface behaviour is being developed separately). Phase 1 through 3 are typically the 'minimal' phases to establish process management. Most straight through process rules will have been defined, and the data required to identify, process and track work will be completed. Once established these phases provide a solid framework for the remaining phases, and a core behavioural system to extend with additional iterations.

Phase 4 – Integration, Event, Timer and Escalation Model

In many process designs, the asynchronous behaviour that occurs can complicate the business process. This phase specifically isolates all asynchronous process decisions from the synchronous rules simplifying the earlier phases and providing a foundation for determining the real asynchronous process behaviours.

This phase, deals with the asynchronous behaviour, events that occur at any time during a business process. It is important to deal with this separately as each individual complication at any step in the process design complicates the alternate paths in the process exponentially. For example, if there are three alternate paths at the second step in a five step process then there an exponentially complex series of paths that need to be tested following the second step, due to the three alternatives at that step. There are likely to be a similar number of variations in the data that may be necessary either to represent the conditions or to pass to the alternatives which complicates the thought processes in the analysis and will result in a non-modifiable process definition. As the event style of process actions are asynchronous (ie they can occur at any time) then this behaviour is far more difficult to analyse. The analysis phases are simplified, by removing these from the thought processes until the basic behaviour of the process is defined, simulated and agreed.

It is likely that the event behaviours that are required may be based on other objects, specifically content objects, within the system. Phase 4 of the content design methodology deals with many of the creation and update events that occur within a system and ties into this phase of the process design. Both behaviours must be analysed separately and the points of interaction modelled. There will be process behaviours that impact on content, and content behaviours that impact upon the processes. To ensure they are managed in the correct environment they must be dealt with separately and only brought together at the end of this phase. When these two phases are joined in this manner, either phase may require additional iterations to deal with the data that has been determined in the other system. The analysis must be isolated in each, but triggered and joined together as appropriate.

The areas that must be analysed within this phase are the exceptions in the business processes. In Phase 1 there will be known paths in the business process that are understood by all. This phase attempts to determine paths that have not been considered or are less frequently traversed within the business process. During phase 1 specific choices will be made to move some process behaviours to this phase, rather than complicate the earlier phases of the design process. This phase, therefore, can become significantly larger than phase 1, as most business processes benefit comes from dealing with the exceptions rather than the straight forward processes.

In addition to the defined exceptions, which are likely to be synchronous, there are many asynchronous events that must be included within the process design. Deadlines, timers, reminders are behaviours that can complicate the basic process analysis unless they are moved to this phase, where a baseline has already been established.

It is important to define the time based behaviours within the process, as this is a major benefit derived from a process management system. If this is not performed, then it is likely that many time based behaviours are implemented within code and in databases. They rarely deal with the attributes of the process and create static dependencies that undermine the usefulness of the process management system. Most systems that follow this incorrect path become database / application centric solutions that rarely benefit from the ability to change or grow. The more dependencies built into the code, the less ability the process design has to deal with change.

Typical time based behaviours are deadlines and reminders for completion of an entire piece of work, for a sub process within a piece of work, and for a single step within a piece of work. This separation is important as it simplifies the time management behaviour, whilst isolating the behaviour from other areas of the process. The boundaries of process interaction are either the entire process, a sub process or a single step, which allows others to make use of the process and know the time based behaviour that they implement. A simple rule should be followed that timers and deadlines are set/reset and stopped on the same submap. This ensures isolation and increases the likelihood of reuse by others.

The process engine deals with deadlines and reminders natively, but imposes some design constraints for how these are implemented. The F_Trackers workflow group should be utilised to implement deadlines and reminders across the entire process and on individual steps within the process. General timer behaviour must be used for sub process based time constraints. Where time based behaviours (including deadlines and reminders) exceed the facilities or configurability of the native techniques then more complicated timer based sub process actions should be used. These should be used minimally and must consider not just the business requirement, but the impact on the process maintenance, data that is carried within the process and the definition of the business logic to support them. It is common for the analysis from this phase to cause changes in Phase 1, 2 and 3 which requires revisiting each of these.

As with all phases of this design, this phase is performed iteratively. As an exception or deadline behaviour is determined, it should be modelled and its impact on the process design should be validated. Exception handling should be isolated into sub-processes and should make use of the complete features of the process engine. In many cases, exception handling is not attachable to the process design at a step, and the user interface may be required to cause process change, without having a specific definition in the process map. The analysis of the processes required to deal with exceptions, deadlines, reminders and escalations should focus purely on the actions that are needed to deal with the event. The linkage into the remainder of the process occurs after the analysis of the actions has occurred. The linkage can be statically from within a calling process (deadlines, timers) or dynamically from with the user interface code (development). The actions taken in the sub process are not dependent upon the linkage technique.

Deadlines and reminders and the notification behaviour that occurs from them must be analysed, and the native ability of the process engine to deal with these should be understood. The system has the ability to notify users that are allocated to steps, or who are part of the trackers (F_Trackers) group to be automatically advised of actions taken or steps reached within the process definitions. Specifically, the use of milestones and the notification techniques that these support should be examined to ensure that out of the box capabilities are used as much as possible. Where specific notification behaviour is required which cannot be implemented natively, then this must be defined within sub-processes and will likely require coding to implement any notification such as email notification. To be specific, notifications can only be provided when work is delivered to users, not generally to queues. If work is pooled to a queue then the system is unaware of any user that may be notified. The milestone capabilities, however, do support notification to the nominated members of the tracker group when milestones are reached, which can be used to indicate that work is ready to be processed. Where other notification techniques are used then these must be isolated in sub-processes and utilised in the process as needed. It is likely that an email notification component may be needed where specific behaviour that is not handled natively is required. This should however only be considered once the complete native behaviour of the process engine is proven to be unsuitable.

Deadlines, reminders, escalation and exception processes are considered part of the business process and not something that the user interface or integration components should influence. It is important that the process is complete and that elements of the process represent these sub-processes. It is typical for statistical and audit data to be heavily dependent on these characteristics. If these are dealt with by the user interface or integration components then the process engine is being underutilised, and any changes which are very likely in these types of processes, will be statically defined outside of the process engine, within the user interface. It is important that these areas of process design are completed before any user interface or integration components are developed. The prototyping approach for handling time and event based processes remains.

The focus of these process steps is the actions that are taken, and should not depend on the people involved in the process execution. The process design must consider what is needed to be done to the work to deal with the reminder, deadline, exception or escalation and not any consideration of to whom the work may be delivered. For example, during escalation the work requires managerial approval rather than the work is delivered to a manager. The same approach applies to notifications. The work requires processing, the notification to a user that the work exists does not change this fact and the process must represent that actions that must be taken on it.

At this point the business process should be defined completely, to a level where the actions that are needed to be done for the process are known and are represented within the business process. The basic process flow (Phase 1), the data that is used (Phase 2), the audit and reporting data (Phase 3) and the exceptions and alternate paths through the process are known. Interfaces to external systems should have been defined in terms of the actions that need to be performed (eg Update Customer Balance) and development and extension of these interfaces should be possible. The process will grow from the baseline completed to this point.

Phase 5 – Process Security

Phase 5 deals with the security attributes of work to determine its impact on the process design and the access methods needed to process work.

The process security model focuses on two dimensions, rosters and queues. Roster security controls whether a type of process can be created and / or processed. Queue security controls whether a process can be retrieved and / or processed. The intersection of these two behaviours provides mid-grained security to the process instances within the system. There is no fine grained security behaviour within the process engine and this must be understood and managed in any solution. It is not possible to isolate a single piece of work and alter it's security, the structures simply don't exist within the process engine.

As the granularity of access is defined by the roster and queues that contain work, this phase can impact the topology of a process environment. Where security attributes must be represented, the queues and rosters of the system must be altered to effect the desired change. Care must be taken as any change to existing queues or rosters will affect all objects within these structures.

Due to this constraint, it is important to ensure that security requirements are definitely security implementation related, as many seeming security behaviours are in reality process and queuing behaviours. For example, ensuring that the same person doesn't both process and approve work is a queuing and task behaviour where the second task in the process can't be performed by the same user as the first task. This is a filter on retrieving the work at that step, and not a security implementation.

Security behaviours should be reserved for explicit security, such as confidential work that can only be seen or processed by a select group of people. Security should not be used as a filtering mechanism, such as restricting access to work items based on geography. For example, securing work so that only someone in Singapore can process this type of work, rather than allowing anyone to process it, implies a heavy load on the process engine to evaluate security. A large number of work items, where many of them aren't for Singapore, would cause significant load on the system. In this case, the work should be filtered based on an attribute, such as processingCountry, rather than a security model behaviour that is implicitly much more complex and not intended for this behaviour.

Security, as in most systems, should be specified at the group level, in that only Groups of users from the directory service should be applied to any object, if security is required.

Once the 'true' security behaviour of a process is determined, then like processes can be determined and any aggregation or separation of processes can be completed. Where complex security requirements are apparent, then further analysis and probable process separation will be required. In most systems, security is a simple behaviour that can be readily managed. It is unlikely that the system will not deal with the business requirements, but it is important that they are represented through the correct techniques.

A common poor behaviour is to model the security that may exist for the content and to extrapolate this to the work. There is no need for the security of the content and process engines to be similar. In most cases, the security on the content will be tighter than the security on the work item, which may imply an additional filtering behaviour to ensure that only authorised users are delivered the work for processing. In these cases, the work item security and content security must be evaluated together, both at the end of phase 4, so that the issues can be identified and a solution determined. It is important to separate the content and process security behaviours as they will rapidly cause confusion and rework if they are analysed at the same time. The models in the two engines are different, and the behaviours must be defined as discretely as possible and in small increments during the prototype behaviour for this phase.

As with all other phases, the security phase is an iterative model. Incremental changes are made to the security attributes of the system and these are tested. It is likely that additional steps may be added to the process behaviour as specific security behaviours are identified. It is also likely that system integration work may be required as no security attributes are natively modifiable from the provided CE_Operations component. It is common for specific content engine behaviour to be developed, whilst it is also common for the process engine behaviour to be represented as filtering rather than specific security attribute allocation.

It is common for phase 5 and phase 6 to be closely tied. Security attributes will be determined to be filtering behaviours rather than security, and queuing behaviours may alter security behaviours of the system. In these cases, phase 5 and 6 may be iterative and recursive whilst the iterations occur.

There are other security attributes of the P8 platform which are outside of the scope of this design model viz. The system security, service security and directory service implementation. Whilst these characteristics are important to the overall behaviour and management of the system, they should not be represented in the process design behaviour. The organisational, political, geographical and topological aspects of the system wil change over time, and including them within the process behaviours at this point will compromise the ability of the processes to deal with changes in these areas and in the business model.

Phase 6 – Queuing and Work Delivery Model

All activities to this point in the design process, have been describing what occurs to pieces of work and the reasons for the actions that have been taken. This phase deals with the mechanisms to retrieve and filter work within the system, in its simplest terms "work allocation".

Work allocation within the process engine should be considered an exception time behaviour and not a design behaviour. Processes should not explicitly allocate work to people, but at runtime, the process engine should support the filtering of work to appear as if the work was allocated to the people. By following this approach, queuing theory shows that the most work will be processed in the most efficient manner. If the queuing behaviour implies the delivery then the two behaviours must both be met for work to be processed. For example, the queuing behaviour that occurs when someone enters Singapore or most other countries, is not pre-allocated. Travellers enter a queue and await the next available immigration official for processing. Each traveller isn't assigned to a particular immigration official at any time, allowing spikes and delays in the volume and arrival of travellers to be handled. Two models tend to occur in this particular example. Either multiple queues exist, separating perhaps local residents from foreigners, or two single queues exist where a continuous queue of travellers are presented performing the same function. Either model still employs the same underlying ensuring maximum throughput whilst separating high value requirements. This behaviour and this approach should be applied also to business process queuing behaviour, where queues are used to separate and prioritise high value or high volume behaviours, and not to define work allocation.

This phase deals with more data than earlier phases, yet relies on the validity of the earlier phases. Changes to the business rules should not result as part of the analyis at this phase, though changes to the process itself may occur. For example, where high volume or geographic separation of work item data is needed, then this will require additional steps in the process definitions to represent the additional queuing model. Volumetric and geographic data will influence the queuing model as queues will be used to separate high volume objects from others, and geography implies local queues of work. This in turn will affect the system topology, which may require changes to the security model and database configuration. All of these attributes must be analysed after the business process has been shown to represent the business activities that are performed (ie after phase 4 at least).

A characteristic that results from the earlier phases is that work will be grouped in individual queues that represent a range of different activities and states. Earlier phases focus on what is done and not who does it, which will mean multiple types of work will enter the same queue. This phase must now ensure that the right people do the work, without altering the business rules and other analysis activities that have occurred earlier. Queue and Roster filtering provides this capability and supports the requirement to filter out work from a queue so that only the relevant pieces of work are shown to a user, effectively allocating the work at runtime to the user.

Filters are similar in functionality to "where" clauses in SQL. They must be based on data that is determined to be important to the filtering behaviour and this will affect the data that is carried in the process. One consequence of the methodology is that this phase will typically cause phases 1 through 5 to be re-evaluated as the requirements to implement queuing behaviours are determined. For example, if work within a particular state should only be seen by users in Singapore whilst the queue contains work from multiple countries, then a filter must be applied over the queue to show only work that is relevant to Singapore. This implies that the process must have this data within it, which means that phase 2 of the methodology must be revisited to ensure it is captured at the correct time in the process. It is likely that this data is also relevant to the auditing behaviour of the system, and phase 3 should be revisited.

Any data that is required to filter or sort the queuing behaviour to mimic runtime work allocation must be present within the process definition. For example, if work should be filtered by country as in the previous example, then the country should be retained. If within each country then only specific types of work or values of work should be shown then this filter must be extended to include these values and the values should be within the process definition. Sorting behaviours are similar. Extending the earlier example, if the value of the individual items is important so that the highest value work is process first then a sorting rule must be defined that sorts the queue in this fashion when it is presented to the user in addition to the filter related to the country.

Filter and sorting data items must be exposed, so that the data is available on the tables within the process engine, rather than existing solely within the process object (ie Blob). Filter data can be exposed on either the roster or the queue based on the use of the data element. If the data is to be used to prioritise or filter work for delivery to a user or system then this data should be exposed on the queue, which also implies it must be available within the process by the time it is exposed. If the data is to be used to identify or locate the work, then this data should be exposed on the roster. Data that isn't exposed and is used in this fashion will incur an overhead within the process engine that can cause serious performance issues.

Exposing data on queues and rosters creates columns in the tables that represent these structures. It is important that only those data elements that are needed for filtering or sorting are exposed as poor database design will result. It is also important to define the indexing behaviour on these data fields so that relevant and rapid retrieval of information is possible.

As a consequence of performing this analysis of the 'filtering' behaviour of the system, separate queues and rosters may be created. This must be a conscious choice that is used to meet business requirements, and is not something that flows from the earlier phase 1 business rule activities. Process definitions should not arbitrarily assign work to queues just because the designer thinks they might be separate queues. The simplest queing response will always be with everything in a single queue and even though this is hardly practical, it should be a goal. Keeping the minimum number of queues and rosters in the system will simplify maintenance, and will promote good process design. Designs that arbitrarily create queues before the business process is understood will result in poorly performing and difficult to maintain systems. Queues can only be created once the business process are complete (Phase 4) and can only be truly determined once the volume, security and geographical attributes of the system have been completed. If these are done at the same time, then a convoluted system will result. If no analysis of the queing behaviour is performed, then a poorly performing system will result.

Good practice focuses on making available only the data that is required, and when the amount of data exceeds good practice then decisions must be taken to deal with the amount of data. All business process modelling in earlier phases must be done using simple queues, and when requirements need additional queues, then these can be added.

One consequence of using the process design tool, is that once a process is associated with a roster and eventlog it cannot be changed. This creates a minor issue that may impact your prototyping behaviour. Prior to commencing phase 1, a decision on whether a new roster and event log should be taken. If necessary, create a single roster and an event log prior to starting any process design. If an already existing process is being extended, or an existing roster and event log are available then this can be used. Queues and their creation and use, however, will only come from the output of this phase of the analysis.

One queuing behaviour that is common and is representative of the purpose of this phase, is when work is allocated to one user for processing and another user for approval. The focus in the business rule (Phase 1) is what is performed to the work and not who does the work. The focus in this phase is how to ensure that the correct user does the work. The implementation of this model is simple. The filter applied to the second task (the approval) must be where I am not the user that did the first task. Each of these data values can be represented using standard data elements of the process engine (refer to F_ and system fields in the ecm_help) and the filter that is applied is done at execution time, rather than through any system or component steps that may pollute the process design. In the future, there is no change to the process definition if some other rule is applied, such as changes to approval limits. The work still requires approval, who performs the step is a filter that is applied at run time.

Volume, geographic and work distribution metrics will affect the queuing model. Normal database design and queuing theory applies when large volumes of data are present within queues. Assume, for example, that a single queue contains three pieces of work each worth 10 million dollars and fifty thousand pieces of work worth each worth ten cents. The work processes may be identical for these items of work, but it would be prudent to process the three higher value pieces of work first, implying a filter and sort behaviour. If in the same scenario, all work was worth the same amount, but there are three staff in Singapore who process the high value work, and there are fifty staff in Sydney that process the lower value work, then this may imply different queuing behaviour rather than a single queue with a filter. It makes little sense for the three pieces of work to appear for the Sydney people, yet it may make sense for the Sydney work to be available to the Singapore people.

Queues are used to represent any sensible separation of data to meet processing requirements. Queues can be located in different geographical locations and can be filtered and sorted separately. When these requirements make sense from the business requirements, then separate queues should be used. Queues should not be created to represent different steps in a process, as it is good practice to have many different types of work at many different steps in the same queue.

Queuing behaviour is used to model workload, filtering and security. As mentioned in earlier phases, security can be applied to rosters and queues. This should not be confused with filtering and security should not be used as a filtering mechanism. Filtering should be used for work allocation, and security should be used to ensure secured work items or secured processing activities are only available to authorised users.

Phase 7 – Interfaces

Earlier phases of the methodology have dealt with the analysis practices and approaches to define the business processes. The final two phases have little direct input from the process and content design activities. They have interactions with the process, but are not the purpose of them.

This phase describes the areas of interaction between the user interface and the process behaviours that have been designed, and the interactions between the process and external systems.

In all earlier phases, the out of the box user interfaces have been used, and an explicit choice to remove any user interface dependency from the process has been defined. It is poor practice for the user interface to be determined prior to the definition of the process and content design. In much the same way as user interfaces shouldn't define database design, they shouldn't drive process design. There will definitely be areas where the user interface must interact with the process and content areas of the system, but the processes must be agnostic of how they are represented to a user. There several constraints and behaviours that must be considered in the user interface, and this will be used as input to whatever technique or tool is chosen to deliver any user interface functionality.

Within the process and content design space, the user interface must perform several functions to support the solution.

Two models exist for interaction of the user interface with the process engine. The first is where the process engine drives the user interface that is appropriate to the current task as determined by the process engine. The second is where the user interface makes reference to data that is within the process engine, and provides input to the process engine at the end of any process step. Both of these are implementations of the same basic behaviour: the process engine is the arbiter of what must be done and the user interface must work within the bounds of what it is told to do, to deliver the final functionality. For example, if the current piece of work is at the "Select Approvers" step, then the user interface must at least perform the actions that related to "Select Approvers", taking whatever data is provided by the process engine, and providing back whatever data is relevant to completing the step that has been indicated. No other data need be provided and no other action need to performed for the process to be satisfied. The user interface can perform whatever other functions it needs, as long as the data being passed in and the data being passed out meets the requirements of the step that is being processed. In this example the user interface is doing as it is told. The two variants that were described earlier are whether the user interface is invoked directly by the process engine, or whether the user interface interrogates the process engine for a piece of work and then presents the desired interface. In both cases, the end result must be that the user interface advises the process engine that the task has been completed in some fashion.

All the activities that occur within the user interface are at the discretion of whatever code is used to deliver the user interface. Some standard tools existing with the process engine are described below:

Step Processors	A step processor is a user interface that is invoked from the inbox provided by the process engine. It allows the entry and modification of data and the indication of status changes that will determine the next phases of the process. The out of the box step processors perform minimal data validation and provide the means for prototyping process designs (phase 1 through 6). There are several different out of the box step processors, which can be used. They all have similar characteristics and differ mainly in presentation style.
Forms	Electronic forms can be used to implement the data entry and validation that is required for a step in the process. A form processor is an extension of a step processor in that it is still invoked from an inbox, but it utilises an electronic form that is separately defined and maintained to implement complex data entry, validation, and guidance.

BPF	the Business Process Framework has the ability to generate form-like behaviour through configuration, whilst supporting inbox and other process centric behaviour. It is the main user interface used for case management processes. BPF also supports the use of electronic forms which can be used in conjunction with native BPF user interface components.
Inbox	The inbox is a native function within Workplace, that supports listing work that is assigned to a user or to a queue. Simple work listing, filtering and sorting behaviours are supported and this is typically used to model the business processes (Phase 1 through 6)
Custom	Custom applications make direct use of the process engine APIs and retrieve work, process data and dispatch work under its own rules. BPF is a generic implementation of such an application. The content and behaviour of the user interfaces for these applications is out of scope for this design methodology.

The major functions performed by all of these user interface components are:

Data validation	ensuring that correct data is entered, and that validation with external systems is performed at the time the user is working with a piece of work
Work delivery	allowing a user to select which piece of work they wish to process, typically supporting filtering and sorting criteria.
Work processing	the finalisation of work where data is completed and the work is dispatched to wherever it's next step may be.

This phase should not impact significantly upon the earlier phases, and should be performed only when the earlier phases have all been completed to some degree.

There are typically many data elements and validation requirements within the user interface. These should not be represented within the business process, as the process doesn't need or want to carry large amounts of data around for processing. The process should contain data that is used to reference data in external systems and the user interface should perform any lookup, retrieval or validation that is needed without reference to the process behaviour. Minor changes to identification or reference data may be required to support user interface requirements, but major changes to the process design should not occur due to any user interface requirements.

System interfaces form another element of the process engines interaction with the external world, and in the same manner, they should not drive the process design.

The data that is passed to the external interfaces should not corrupt the data that is being carried within the process design, and every effort should be made to ensure that only process related data is within the process engine. This can mean that external data sources are used to retain transient data, or that XML data structures are used to retain and pass encapsulated within the process engine. Business users are the focus of the process design behaviour and large quantities of system related data must not pollute the design behaviour within the system.

The integration techniques and the approaches used to encapsulate or isolate the data from the business process isn't discussed further, but every effort should be taken to ensure that only data related to routing, referencing or reporting should be retained within the process engine.

Phase 8 – Coding specification

Coding is outside the scope of the process and content design methodology, and relies on good practice techniques that are relevant to the developers of the code. There are some dependencies that must be understood and good practices to follow related to interfacing of the process engine to coded modules, but there is no specific activity that occurs in coding that is proscribed by the content and process design itself.

This phase should not be started prior to the completion of at least phase 3, preferably phase 4. The coding that can occur at this phase, should be related to any services or components that are determined based on the process interactions with other systems. Where the process must interact, such as update, another system the code must be created to perform this task, and it must be represented within the process definition.

Initial versions of coded modules should meet simple data passing parameter rules and should comply with common naming conventions. As the process engine is version specific, interface changes that occur cannot be propagated into existing processes. This is a common requirement in any modern development approach and interface definitions should follow current best practice. As system interfaces are further qualified then they should be extended and included into the modelling behaviour of earlier phases.

It is important that any code that is developed is as general as possible whilst still meeting the business requirements of the components. General purpose functions will not be used successfully by users that are designing the business processes, and specific functions should be the design target of this phase. For example, a component that updates a line of business system should not expect the process to provide complex data streams that must be assembled by the process, but should supply suitable legible and understandable interfaces for access. At the same time, the interfaces should not be so specific and convoluted (eg tens of parameters) that their purpose is unclear.

The process engine supports JMS and Java classes that can be invoked, as well as web services. Good practice suggests that web services are the preferred integration technique. Java classes may be used in cases where web services are not possible. JMS should be avoided as it is an outgoing message function only and no reply is possible.

Parameters passed to any interface that is being implemented must conform to the data passing that is possible from the process engine. Methods on java classes are invoked passing data values to the mapped method calls of java classes, and a single result is return. The process engine provides mapping of the data types from standard java data types to process engine data types. Simple data types (string, integer, Boolean, Double, Date) are supported. No object types are supported other than specific process engine types of Workflow Group (participants at a step), and Attachments (reference to content stored in the system). Vectors (single dimensional arrays) of each of these types are supported.

It is good practice for the methods to perform discrete actions based solely on the parameters being passed to the method. It is bad practice, and discouraged, for adaptors to be built that create arbitrary dependencies between the types of work that can be delivered to them. For example, the records management components have specific adaptors to allow manipulation of work item data directly, without reference from the method signature. This approach reduces flexibility and enforces a dependency which in turn compromises reuse.

Summary

This methodology describes a framework that supports the outcomes that are necessary to implement an IBM FileNet P8 solution. It delivers an indication of the phases that will assist in this delivery, but requires experience and an ability to take a pragmatic approach to any decisions that are taken. The methodology provides information to assist in the decisions, and to ensure that future designers and builders of the system can work from your decisions.

Each phase describes the areas that should be considered both from the requirement and the systems views, and attempts to describe the behaviours and pitfalls that are relevant to any decisions that are taken. It is about good practice and the activities needed to support that for each phase.

The content design handbook and the process design handbook take the information in this methodology and provide a further level of detail to each phase, with actions to take and tools to use to assist in completing each phase.

Always do what is needed, but remember, it's someone else who'll be changing the design in the future, and process design is intended to deal with and promote change, not inhibit it.