Institute of Parallel and Distributed Systems
University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3078

Maximization of resource utilization through dynamic provisioning and deprovisioning in the cloud

Florian Fritz

Course of Study:	Software Engineering
Examiner:	Prof. Dr. Bernhard Mitschang
Supervisor:	Dipl.-Inf. Frank Wagner
Commenced:	September 01, 2010
Completed:	March 17, 2011
CR-Classification:	C.2.4, H.3.2, H.3.4

Abstract

The amount of data companies have to manage daily increases drastically from year to year. Whenever work is done information gets created related to a business process. Companies are legally bound to keep record of all their business activities to be able to provide evidence in case of lawsuits. In case of a lawsuit all data relevant to the case has to be collected, put on hold, be reviewed and produced. Lawsuits that require data to be discovered are not rare. According to survey of [Ful10] in 2010 16% of the consulted companies had over 50 lawsuites commenced against them.

To reduce the amount of documents that have to be reviewed, documents should not be kept longer than the company is legally bound to. But this management is not an easy task. Companies need to introduce big content management systems that need to cope with great amounts of data. To do so huge expenses into IT infrastructure have to be made and complex systems need to be managed. With the advent of the cloud computing paradigm the costs of these software systems could be decreased drastically.

IBM is working on a project to provide Electronic Archiving Management as a service using existing Electronic Content Management systems. This service provides means to reliably store documents that are not used in active business processes anymore. Documents are stored until the customer is not legally bound to keep the documents anymore. They are being full text indexed to be easily discovered for litigation cases.

This thesis gives an introduction into Electronic Content Management, Cloud Computing and Multitenancy to provide the basics for this project. After introducing a modeling concept based on Tivoli Service Automation Manager's Service Definitions. The existing software components are examined on their capabilities on multitenancy and segregated into manageable resources. The dependencies of the software components are then being examined and an automatic deployment process is developed. With this deployment process it is possible to reduce the time needed to deploy a new installation from weeks to less than two hours.

Contents

List of Figures

1 Introduction

The amount of data companies have to manage daily increases drastically from year to year. Whenever work is done information gets created related to a business process. Companies are legally bound to keep record of all their business activities to be able to provide evidence in case of lawsuits. In case of a lawsuit all data relevant to the case has to be collected, put on hold, be reviewed and produced. According to [Dir07] the costs to review one gigabyte of data can run up to $2000.

The legal department of the company DuPont assessed their three-year process to respond to one single discovery request. For this request 75 million pages of text had to be reviewed. It was found out that 50% of the documents were kept beyond its retention period and therefore should not have been reviewed. The costs of reviewing these outdated documents summed up to $12 million. [Foc07] states about the discovery process that:

> *"According to multiple studies, this legal process of exchanging and reviewing information represents approximately 75% to 90% of all litigation costs."*

Lawsuits that require data to be discovered are not rare. According to survey of [Ful10] in 2010 16% of the consulted companies had over 50 lawsuites commenced against them.

It can easily be seen that a better management of the companies information can reduce the expenses on litigation cases tremendously. Documents should not be kept longer than the company is legally bound to. But this management is not an easy task. Companies need to introduce big content management systems that need to cope with great amounts of data. To do so, huge expenses into IT infrastructure have to be made and complex systems need to be managed.

With the advent of the cloud computing paradigm the costs of these software systems could be decreased. By hosting the applications for multiple customers on one system the hardware resource utilization can be increased. Usually servers in data centers have an utilization of 5% to 20% to be able to cope with peak loads. Having multiple customers with different workload distribution can lead to a more stable resource consumption and the difference between average and peak load can be reduced. This will save hardware costs and enable an application provider to reduce its charges.

This thesis is part of a project that will provide an Electronic Archiving Management system as a cloud service. The rest of this chapter presents the technical fields, the project is based

on. At first the basics of cloud computing are described. After that an introduction into enterprise content management will be given. Chapter 2 provides an insight in the concepts of multitenancy. After that chapter 3 introduces the Archive Cloud Service project of IBM®. Chapter 4 presents a service automation software and will define a modeling language for deployment automation. Chapter 5 decomposes the Archive Cloud Service into manageable components and describes the deployment process of these components.

1.1 Cloud computing

According to Google Trends[1] the term cloud computing has become popular in the computer science area within the last three years. Being used in marketing campaigns by several companies however has created various understandings of what cloud computing actually is. Cloud computing is tried to be sold as a brand new technology, but in fact it is just a new paradigm including many old concepts of providing software to a customer. The following subsections will present a commonly accepted definition of cloud computing by the National Institute of Standards and Technology (NIST). It is separated in three parts. The first part describes basic characteristics of cloud computing. The second part presents different service models of cloud computing and the third part how they can be deployed.

1.1.1 Cloud Characteristics

The Cloud model of the National Institute of Standards and Technology [MG09] describes five essential characteristics:

On-demand self-service A customer is able to request a services without any human interaction needed on the side of the provider.

Broad network access The services are available over network and can be consumed with *"heterogeneous thin or thick client platforms"*.

Resource pooling The provider manages a pool of *"physical and virtual resources, that are dynamically assigned and reassigned according to consumer demand"*. There is a *"sense of location independence"*, the customer has no control over how the resources are assigned and where they come from. The customer might however be able to specify the country, state or data center in which his service should be provided.

Rapid elasticity Depending on the demands of the customer, additional resources can be provisioned and deprovisioned to quickly scale in and out. Resources appear to be unlimited and can be *purchased at any time or quantity*.

[1] `http://trends.google.com`

Measured Service To optimize the utilization of resources in a cloud system, providers use metering capabilities to control the resource allocation. Metering is also used to determine the amount of resources used by a tenant for billing.

1.1.2 Service Models

Cloud computing services can be delivered on different layers. The National Institute of Standards and Technology [MG09] defines three service models. These are Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS).

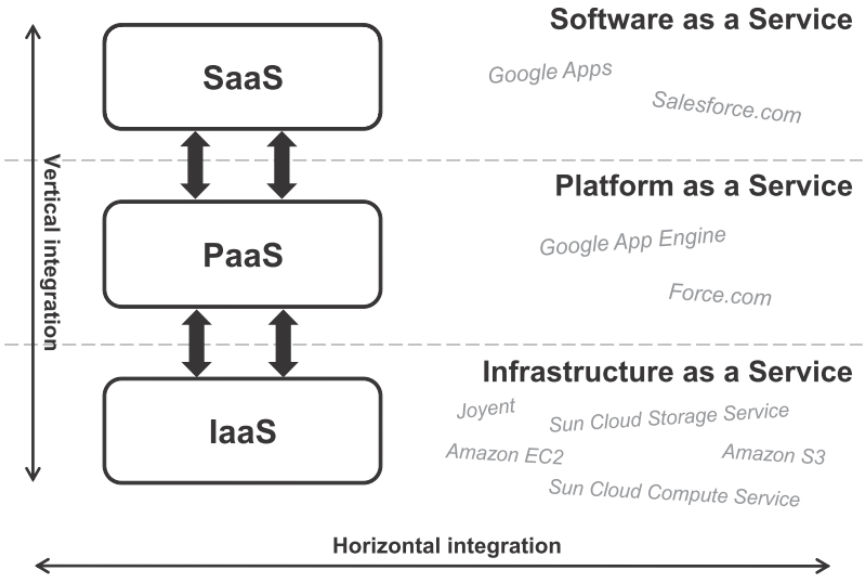

Figure 1.1: Different cloud service models [SSW10]

IaaS is defined as providing *"processing, storage, networks and other fundamental computing resources"* to the customer. The customer himself is responsible for deploying operating systems and applications. The underlying hardware can not be controlled by him. He only requests the desired amount of resources, how they are provided is transparently managed by the cloud provider.

In the PaaS model the customer is provided with an infrastructure in which applications can be deployed. These applications are created provider specific with certain programming languages and tools. The customer has knowledge or control over the underlying servers, operating systems or storage. The control is limited to the *"deployed applications and possibly application hosting environment configuration"*.

SaaS is a model in which the customer does not deploy any software in the cloud infrastructure. He is provided with a functionality that is *"accessible from various client devices through a*

thin client interface". Access to the cloud infrastructure is limited to *"user specific application and configuration settings"*.

1.1.3 Deployment models

Cloud infrastructures can be used by different groups of users. The cloud definition of NIST names 4 different deployment models, namely Private Cloud, Community Cloud, Public Cloud and Hybrid Cloud.

Private Clouds are used by one single company. These clouds can be operated by the company itself or by a third party. They can be located on or off premise.

Community Clouds are shared between multiple organizations with *"shared concerns (e.g., mission, security requirements, policy, and compliance considerations)"*. Like Private Clouds, Community Clouds may be managed by a third party or the organizations themselves and it may be on or off premise.

Public Clouds are available for the *"general public or a large industry group"* and is owned by an organization that is selling cloud services.

Hybrid Clouds describe the composition of multiple clouds that are bound together by data or application. E.g for means of load balancing.

1.1.4 Examples of Cloud Computing Services

As described in 1.1 there are different models of service. For each of these models there are several providers. Cloud offerings can be divided into two categories, computing clouds, that focus on running applications and storage clouds that focus on storing high amounts of data. This section provides a brief overview of the most popular cloud service providers.

Computing Cloud Services

Computing Clouds Services are services that provide reliable and scalable computing power on a pay per use basis. This section will describe four different service offerings.

Amazon EC2 is the most popular Infrastructure as a Service provider. With this service the customer can dynamically provision EC2 Instances. An EC2 Instance is a set of virtualized hardware. On this virtual hardware the customer deploys virtual machines with an operating system and the desired software stack. There are no limitations on the software that can be used.

This is a very low level approach cloud offering. Because the EC2 Instances are very application specific it is not possible to offer scalability and failover at the service layer. The customer himself has to build a high available software system, that tolerates the failure of single EC2 Instances.

The pricing model of EC2 is based on EC2 Instances per hour. There are different virtual hardware configurations with respect to memory size, CPUs, storage size and I/O performance. Each configuration has a cost per hour, regardless of their utilization. This way running one EC2 Instance for 1000 hours costs the same as running 1000 EC2 Instances for 1 hour.

To efficiently use this cloud service the used software has to be able to quickly adapt to changes of the resource demand by rapidly scaling both up and down.[AFG+09]

Microsoft Azure is a Platform as a Service solution. The customer can deploy applications into an application framework, that supports scalability and failovers. But the applications have to be specifically developed for this framework.

This approach allows the customer to develop scalable applications without having to cope with low level problems like data base availability and performance. Used machines are defined with declarative descriptions. They contain what kind of machines are needed, how they are connected and how they can be replicated. The platform is then able to scale these resources up and down as needed.

The pricing is based on an instance per hour concept.[AFG+09]

Google App Engine is also a Platform as a Service system. It is however more restrictive in its application than Microsoft Azure. Google App Engine is a framework exclusively for web applications. Applications have to be developed specifically for this framework, which enforces a strict separation of a stateless computation tier and a stateful storage tier. It also requires the application to be request-reply based.

These enforcements lead to a highly scalable and high available application that manages its resources automatically. The application developer does not have to consider any of these issues. [AFG+09]

The pricing of Google App Engine is very flexible. The customer only has to pay for the CPU hours, bandwidth, storage and sent emails that are actually used. [Goo]

Google Apps and Salesforce are popular providers of Software as a Service solutions. They both provide many business management applications integrated into each other. Typical examples are email, Content Management, Customer Relationship Management or Human Resource Management.

Pricing is usually done on an application per user per month basis.

Storage Cloud Services

Storage Cloud Services provide a secure, reliable and scalable data storage. Customers do not have to cope with data replication, backups or performance of the storage. As the data is mostly needed within Computing Cloud Services their providers usually offer storage solutions as well. The storage solutions of the providers mentioned above are: Amazon S3, Microsoft Azure Storage and Google Storage.

The pricing of all providers are based on storage in gigabyte per month, bandwidth usage and number of requests.

1.2 Enterprise Content Management

This section provides an introduction in the field of Enterprise Content Management (ECM). Companies produce a lot of content every day. [Pro10a] describes content as information about a document that is provided in electronic systems. This information contains the content of a document, its layout and metadata about it. [Pro10a] describes three kinds of content:

structured content Data that is provided in a known layout, like formated datasets from a database. Data, layout and metadata can be completely separated.

weakly structured content Data that contains layout or metadata in a not standard form within itself. Examples for this content are text documents and presentations.

unstructured content Data that contains content, layout and metadata within itself. The information within this data can not be accessed directly. For example images, videos or sound files.

Independent of the form the content is provided it contains knowledge. This knowledge has to be carefully managed. The existence of the knowledge within the company alone is not enough. It has to be made available when and where needed. ECM describes the technologies used to manage this knowledge. The Association for Information and Image Management (AIIM) defines ECM as:

> *"Enterprise Content Management (ECM) is the strategies, methods and tools used to capture, manage, store, preserve, and deliver content and documents related to organizational processes. ECM tools and strategies allow the management of an organization's unstructured information, wherever that information exists."* [AII]

Figure 1.2 gives an insight about all the tasks related to capture, manage, store, preserve and deliver content.

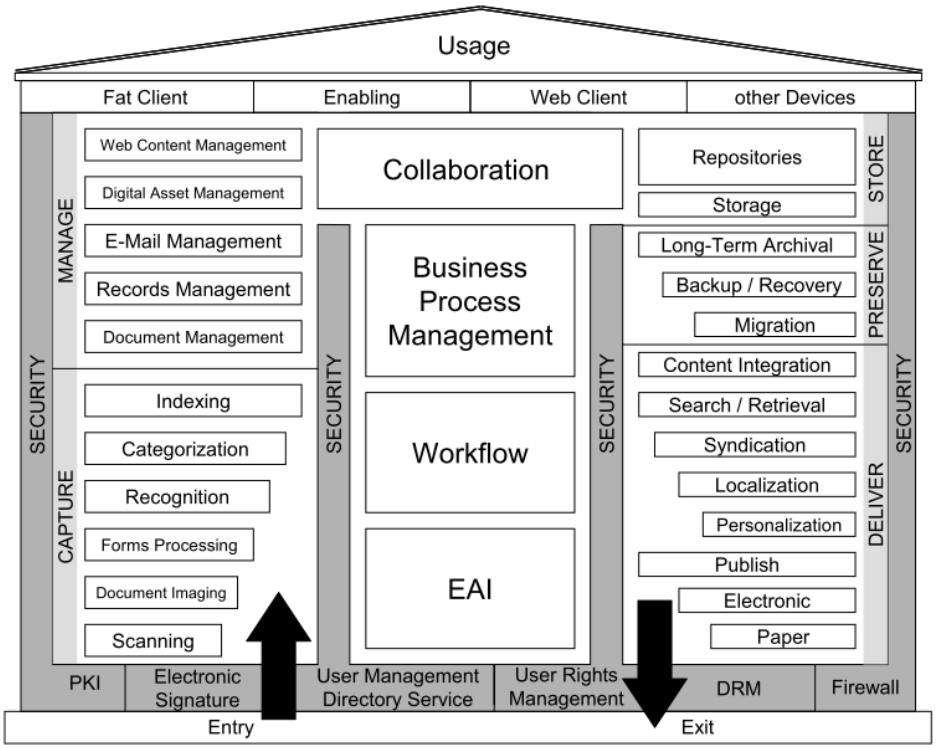

Figure 1.2: Activities in ECM [Kamo6]

The capture component contains all tasks required to aquire content ranging from scanning paper documents to categorizing and indexing them. The manage component deals with the management, processing and using of content. The task of the store component is to temporarily store, search and retrieve content. Unlike the store component the preserve component deals with the longtime archiving of content and its backups using various storage solution. The deliver component has to ensure security for the content and also transform and distribute it.

The focus of this thesis is on the manage and the preserve component. The manage components Document Management and Records Management which are most relevant in this project as well as the preserve component will be described in detail later in this chapter.

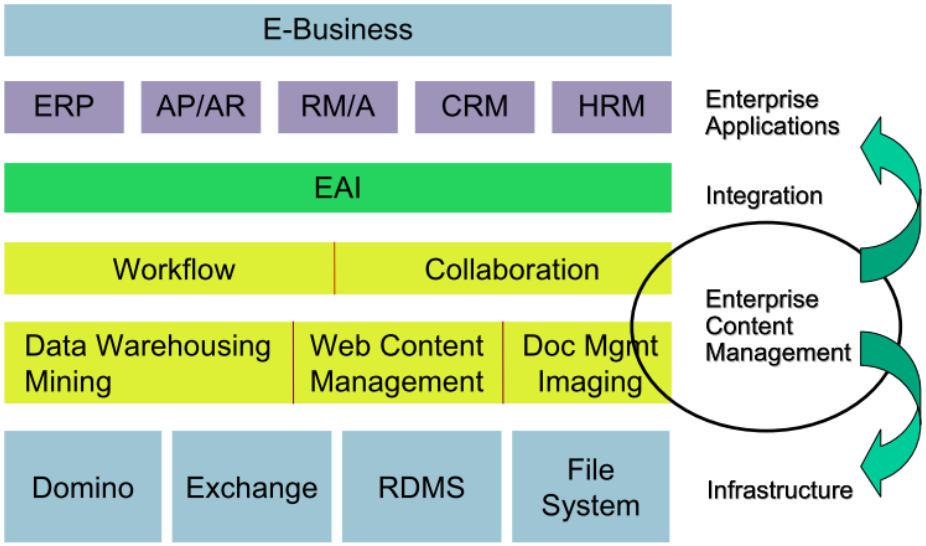

Figure 1.3: Role of ECM in an Enterprise System [Kam06]

Figure 1.3 displays the role of ECM within a modern company. Together with Enterprise Application Integration (EAI) it builds the middle ware for all areas of management like Enterprise Resource Planing (ERP), Accounts Payable and Accounts Receivable (AP/AR), Records Management/Archiving (RM/A), Customer Relationship Management (CRM) and Human Resources Management (HRM).

1.2.1 Document Management

The application area of Document Management describes the management of documents and their life cycle. A document management system has to manage data in different formats from different sources that are stored into a storage hierarchy with version and history management. In contrast to an archiving solution documents can be modified, combined with other documents and be attached with informations about their status. The application area of Document Management describes the management of documents and their life cycle. A document management system has to manage data in different formats from different sources that are stored into a storage hierarchy with version and history management. In contrast to an archiving solution documents can be modified, combined with other documents and be attached with informations about their status.

A Document Management System manages the creation, modification and archiving of a document by adding meta information about the business processes it is being used in.

Compound documents, that consist of different information sources like text, data, images, audio, videos or links are getting more and more relevant. Examples for this kind of documents are encyclopedias, training materials or manuals. They allow keeping relevant information gathered together in the context they belong to.

Dealing with big amounts of data an intelligent storage hierarchy has to be used, that stores the document depending on their access speed and frequency as well as the cost of storage. This storage hierarchy may contain memory storing, hard drives or tapes, where memory is the fastest and most expensive and tapes are the slowest but cheapest storage.[Pro10b]

1.2.2 Records Management

While the purpose of Document Management Systems is to provide easy access to documents throughout a company to collaborate, the purpose of a Records Management System is to document *"statutory, regulatory fiscal or operational activities within the organization"* [Roe10]. The central element of Records Management is the record. The International Organization for Standardization (ISO) defines records in the ISO 15489 standard:

> *"Record: Information created, received, and maintained as evidence and information by an organization or person, in pursuance of legal obligations or in the transaction of business"*

Records Management enables a company to control *"the creation, receipt, maintenance, use and disposition of records"*. This way records can be used as reliable evidence for all business activities. In case of legal investigations a company is legally bound to provide all evidence of a legal case, this is hardly possible without systematic records management.

According to [ISO01] records management within a company includes:

- setting policies and standards
- assigning responsibilities and authorities
- establishing and promulgating procedures and guidelines
- providing a range of services relating to the management and use of records
- designing, implementing and administering specialized systems for managing records
- integrating records management into business systems and processes

Introducing records management into a company is not an easy task. All policies and procedures have to be analyzed about their *authenticity*, *reliability*, *integrity* and *usability*.

For a record to be *authentic* it has to be provable what the purpose of the record was as well as when and by whom it was created. To be *reliable* it has to fully and accurately describe the related business activity. For the *integrity* of a record policies and procedures have to

specify what additions or annotations can be made to a record as well as who is authorized to do so. Every change made to a record has to be explicitly indicated and be traceable. A *usable* record can easily be located, retrieved, presented and interpreted. A record should therefore contain all information needed to understand the business activity as well as links to all related information to it.

1.2.3 Electronic Archiving Management

Electronic Archive Management (EAM) is a part of Records Management, that only deals with archiving records that are not part of active processes anymore. As mentioned before records have to be stored for a certain amount of time depending on their type. In the United States these retention periods are regulated by the sarbanes oxley act. Some examples for these retention periods are [Balo7]:

3 Years Employment Applications

5 Years Invoices to Customers, Invoices from Vendors, Purchase Orders

7 Years AP/AR Ledgers, Payroll Records & Tax Returns, Inventories of Products

Permanent Bank Statements, Contracts & leases, Legal correspondence

For these time periods the documents have to be stored in a way that they can quickly be discovered and retrieved for litigation cases.

Compared to a regular Records Management system an EAM system has different workload characteristics. While a Records Management system has to deal with a lot of living documents, that are retrieved and change their contents and statuses, an EAM system deals with documents that usually don't change. An EAM systems task is to store and index all records to make them discoverable in case of an investigation.

According to [MKW+09] the goals of an EAM system are:

> *"The designing of an EAM system must put major focus on the performance and scale aspect such to efficiently handle an unknown and variable number of documents, ingested at highly variable rates by a possibly very large population of end users or principals."*

That means an EAM system has to cope with highly irregular ingestion processes and still provide an acceptable performance. Ingesting a record requires several steps with different resource requirements:

Extract The first step of archiving data is identifying the data source and extract the relevant data according to the archiving rules. This data source could be a Records Management System.

Identify After extracting the data from the data source unique identifiers for each document has to be generated by using hashing algorithms.

De-duplicate Using the identifiers from before duplicates are being searched and eliminated. Only documents that are not already in the archive are being stored.

Classify In this step documents are being classified by their content or metadata. Document classes specify access rights and retention policies for the documents.

Decompose This step decomposes documents that consist of multiple data parts like headers, bodies and attachments.

Transcode In this step components of the previous step are transcoded into plain text representations of its content.

Annotate Using linguistic methods the document is enriched with annotations describing the document to prepare it for full text indexing.

Transform & Archive After these steps the document the document is enriched with all needed information and can be stored into the archive. In some cases the document might be transformed into other formats like HTML or PDF for convenience.

2 Motivation

ECM systems as described in 1.2 are complex software systems, that have to cope with huge amounts of content being created every day and ingested into the system. However the usage of the system varies a lot. In business hours a lot of documents will be created, retrieved or changed. There might also be peak hours with workload multiple times higher than the average. Having to be able to provide a good service even in these peak times, makes it necessary to provide huge amounts of processing power. This processing power however is mostly unused.

Being a critical part in organizations the ECM system has strict requirements concerning availability and disaster recovery. This means that the system has to be available all the time providing a good service and even in case of a destroyed data center no data loss can be tolerated.

To build a system that fits these requirements, high amounts of know how and infrastructure are needed. Both know how and infrastructure are expensive, especially for a company that does not focus on IT. This company therefore has to buy know how and infrastructure, that does not provide a direct business value and is not needed elsewhere.

This thesis is part of a project of the IBM, which wants to provide an ECM solution in a SaaS service model (see chapter 1.1). The customer does not have to deal with any hardware or software anymore he only orders a service that he can integrate into his business.

The next sections will describe the concept of multitenancy and how it is enabling this project.

2.1 Multitenancy

An important concept for Software as a Service is multitenancy. In a multitenant environment multiple customers or tenants share parts of the software they consume. This is done in order to reduce the cost of operation per tenant compared to hosting a software environment for each tenant.

Multitenancy has no direct benefit for customers. It has however the indirect benefit for the customer that it reduces the cost for the service provider, who is then able to provide

a service at a lower price due to economy of scale. Multitenancy is accepted by customers because the cost reduction outweighs the disadvantages of shared resources.

The next sections describe how resources can be shared between tenants and what special concerns have to be addressed doing so.

2.1.1 Layers of separation

In multitenancy the *"software must appear to each tenant as if he was the sole tenant of the application"* [MMLP09]. Therefore the tenants have to be segregated at some part of the application providing each tenant with his own data. This separation can be performed at many different levels of the application. [MKW+09] differentiates between two tiers of tenant separation, execution tier and data tier.

Execution tier

In the execution tier there are four different levels of isolation (Figure 2.1) between the tenants.

Application level There is only one application deployed. All software components are shared between the tenants. The isolation logic is implemented within the application. There is only one login mechanism that determines which tenant the user is from and provides the respective data.

Middleware level Tenants get their own deployment of the application. All applications are deployed into a shared middleware. This middleware might consist of application servers, database servers or directory servers. Each application connects to tenant specific data sources.

Operating System level In this level of isolation tenants only share operation systems. The used middleware and applications are deployed per tenant.

Virtual machine level In an isolation on virtual machine level tenants don't share any software at all. Every tenant gets separate virtual machines including the whole software stack. Tenants only share the hardware running the virtual machines.

From the first to the last layer the isolation level increases. Higher isolation leads to higher data security and less performance interdependence between the tenants. It also reduces the complexity of customizing the service for the tenant.

On the downside the increased isolation level also increases resource requirements by adding more and more overhead to the system, compared to the actual application of the tenant. Every piece of software that is additionally deployed needs resources to be deployed and

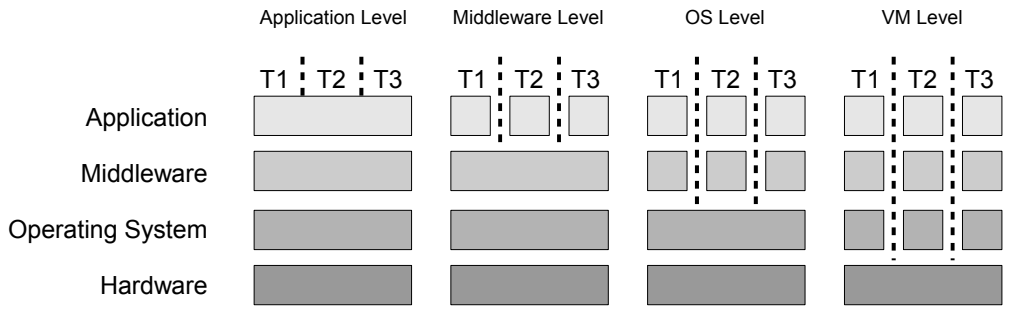

Figure 2.1: Execution tier isolation levels

executed as well as operational resources for maintenance like deploying, configuring, updating and undeploying.

In many cases an isolation on application level is not possible to implement due to legacy software that is being used, which is not capable of supporting multitenancy.

Data tier

The data tier defines the isolation on the data storage. There are also four levels of isolation:

Shared database tables Tenants share the same database tables. Tenant specific data is isolated by a tenant id. To prevent one tenant to be able to access the data of other tenants views on database tables and data encryption can be used.

Shared database Tenants share the same database but do not use shared database tables. Every tenant gets an own set of tables grouped together in a database schema. Data security can be provided by Access Control Lists(ACL) and data encryption.

Shared database server Tenants have individual databases, that are hosted on one database server. All database tables of a tenant run within his own database. ACLs and data encryption can be used for data protection.

Separate database servers On this level of isolation each tenant gets an individual installation of a database server.

These isolation tiers increase in isolation from top to bottom, but they decrease their performance. According to [CCW06] an approach with shared database tables is able to support more tenants with less servers. The shared database approach is appropriate for applications with only few databases per tenant. It allows to host more tenants than a shared database server approach. A disadvantage of both shared database approaches is, that backup and

restore mechanisms for the database are relatively complicated. It is not possible to simply restore the whole database backup into the running system. It has to be restored into a separate database so that the tenant specific data can be isolated and restored into the running system. This includes identifying the tenant specific data in both versions of the database, deleting all data from the running system and replacing them with the backed up version.

The shared database server approach makes it easier to backup and restore data and offer a good data isolation. But on the downside this approach has relatively high hardware costs because databases have a certain overhead. Every database for example has own buffer pools that need memory. There are are also limitations on how many databases a server does support.

In the separate database server setup no resources are shared. It offers the maximum level of data separation but also the highest resource cost.

In the ECM environment data is not necessarily stored only into databases, but also in file systems. The tier level for data isolation can also be adapted to mass storage devices:

Shared folders Tenants have files in shared folders.

Shared file system Tenants have their own folders but within a shared file system.

Shared hard drives Tenants have their own file systems but on shared hard drives.

Separate hard drives Tenants have their own dedicated hard drives.

These tiers increase their isolation level from top to bottom but it increases the overhead in managing them. A higher isolation leads to easier backup and recovery as well as destruction of data. When a tenant decides to quit a service all of his data has to be removed reliably without a possibility of recovery.

2.1.2 Concerns of Multitenancy

Even though multitenancy has a lot of benefits, there are also some issues that need to be addressed.

Isolation

When software and hardware instances are shared between multiple tenants, it is possible to save hardware costs by balancing the load of all tenants. But due to limited resources it is possible that one tenant consumes so much resources, that other tenants Quality of Service (QoS) is reduced. In some cases it might even be possible to bring down the whole service.

Security

Security is a very big issue in multitenancy environments. Special effort has to be put into protecting the data of one tenant from the other tenants. For example how can you make sure that a user of one tenant cannot access the portal of a second tenant.

Customizability

Sharing applications reduces the effort to maintain a service, but it also reduces its customizability. Many tenants may want to adapt the service to their own needs. This might start with simple graphical changes to the user interface for corporate identity reasons. But also changes to business processes and database schemas could be desired. [MMLP09] describes three basic patterns to implement customizability in multitenant applications.

Single instance One instance of the application is shared between all tenants. That means that the same workflow using the same code on the same infrastructure is used by several tenants [MMLP09]. Therefore no tenant specific configuration is possible.

Single configurable instance All tenants share one instance of the application that is configurable. Whenever the application gets invoked, it is customized at run-time for the invoking tenant. This customization data is stored per tenant in *configuration files or configuration entries in a database*.

Multiple instances In this pattern for each tenant an instance of the application is deployed. This pattern allows best customization per tenant, but also the most operational effort. For every tenant new code has to be deployed, configured and maintained.

The first two patterns with the single instance require early consideration of configurability in the development process. The application has to be developed as multitenant applications. The third pattern allows to provide non multitenant application as a multitenant service.

Maintainability

Running only one instance of an application reduces the effort to update its version a lot. But in some cases it might not be desirable to update the instance for all tenants at the same time. Upgrades of software always bear a risk of introducing new errors to a system. If only few tenants suffer from an error in the actual version of the application and most of the tenants don't, applying a patch for this problem to all tenants could lead to more problems.

Recovery

The more multitenant applications share the harder it will get to backup and restore tenant specific data. As mentioned before, it is easier to backup and restore a whole database than tables or even rows. It is also necessary for the system to support online backup and restore for a tenant. This means that the software has not to be halted to perform maintenance. Halting an application for a planned maintenance task in a single tenant environment might be tolerable, but in a system that is used by hundreds or thousands of tenants a downtime to perform maintenance for one tenant is not acceptable.

2.2 Electronic Archiving Management as a Service

Having examined the idiosyncrasies of Electronic Archiving Management systems and the capabilities of Cloud Computing and Multitenancy, it can be said that it seems to be possible and feasible to provide an Electronic Archiving Management solution as a service.

A limitation for such a service could be the network bandwidth that has to be provided to service all customers. Since this is only an archiving solution that deals with records that are not part of active business processes any more, retrieval operations are not very likely. According to [MKW+09] 80% of the workload are produced by the ingestion process and most of the documents will never be retreived. Compared to a common Enterprise Content Management System the bandwidth requirements are relatively low.

It is being assumed that the workload of multiple customers is relatively random and that peak loads of multiple customers are unlikely to appear at the same time. According to the law of large numbers the utilization should get smoothened and the effect of load peaks be reduced.

Having a relatively stable workload, a provider for an Electronic Archiving Service could reduce the expenses on hardware to provide a service to customers at a cost that could overcome the concerns of multitenancy mentioned in chapter 2.1.

3 Archive Cloud Service

The Archive Cloud Service is a project of the IBM which aims to provide a Compliance Archive Management System as a cloud service. The novelty of this approach is not the service that is provided, but the way it is provided. Existing Enterprise Content Management applications are used to provide a pay-per-use archiving solution.

In traditional ECM solutions, the customer buys hardware and software and has to install, configure and maintain the system himself. This leads to very high initial hardware costs as well costs for software licenses. Installing systems will either cost a lot of work or a lot of money when a third party is commissioned to do it.

In the Archive Cloud Service Model the customer does not have to cope with hardware or software acquisition or their sizing. The customer just selects his service by providing several Service Level Agreements like documents ingested per day and a maximum time to retrieve a document. These agreements are then signed together with a contract. After that the service will be immediately provisioned for the customer. When the provisioning is completed the customer will be notified and given access to the system.

The Archive Cloud Service does not try to provide a complete ECM solution to the customer. It focuses on the archiving of records (see chapter 1.2.3). The customers are assumed to already have a working Content Management System that is integrated into their business processes. The Archive Cloud Service provides a complementary service to that system, that deals with the long term archiving of documents that are not longer active.

The customer loads documents that are not used in business processes anymore into the Archive Cloud Service to have them reliably stored and indexed in case of litigation cases. In such cases it is possible to easily discover and retrieve all documents required. When documents reach the end of their retention period they are automatically deleted.

Since IBM already has a wide variety of ECM products no new applications have to be developed for the archiving itself. To ease the use of the system however a web portal will be developed, that reduces the complexity of the system configuration. The following sections will describe the software products that provide the Archive Cloud Service (see Figure 3.1). The infrastructure components for Directory Server (Tivoli® Directory Server), Database Management System (DB2®), Storage (GPFS™) and Application Server (WebSphere® Application Server, IBM HTTP Server) are described in more detail in chapter 5.2. All applications described below run within a WebSphere Application Server environment.

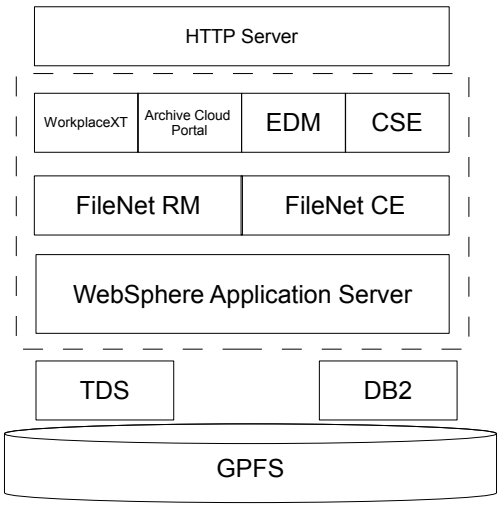

Figure 3.1: Archive Cloud Service software components

3.1 FileNet Content Engine

FileNet Content Engine (FileNet CE) is an extensible ECM system (see 1.2) by IBM. It provides the basic component of the Archive Cloud Service. It uses an object oriented repository that supports management of customer-defined business objects. It allows to define relationships between objects and manage their life cycle.

There are several addons for FileNet CE to extend its capabilities to support Records Management or integrate it with office solutions like Microsoft Office or Lotus® Quickr®. In the Archive Cloud Service the extensions Workplace XT, FileNet Records Manager and eDiscovery Manager will be used.

3.2 FileNet Records Manager

FileNet Records Manager (FileNet RM) enhances the FileNet Content Engine with capabilities of Records Management (see chapter 1.2.2). This contains automating the lifecycle process of records and enforcing compliance policies on them. It prevents records from being accidentally destroyed or altered while they are still needed for business purposes. [ZAB+09]

3.3 Workplace XT

Workplace XT is a user frontend for the FileNet Content Engine. It is a web based application that exposes content related functionality to the user. He can ingest documents into the FileNet Content Engine, declare documents as records and retrieve documents.

3.4 Content Search Engine

The Content Search Engine (CSE) creates full text indexes of ingested documents for eDiscovery.

3.5 eDiscovery Manager

The eDiscovery Manager (EDM) is a tool for electronic discovery. It allows *"authorized IT and legal users [...] to search, cull, hold and export case-relevant documents"* [ZCF+09].

3.6 Archive Cloud Portal

The Archive Cloud Portal provides an easier to use interface to the archiving solution. It combines all required administration functions of the included applications. This way there is one central point to manage the Archive Cloud Service. It provides the functionality to manage access control, document classification and archiving processes. The portal centralizes only administrative tasks to work with the archived documents and to discover documents for litigation cases Workplace XT and eDiscovery Manager are used.

4 Service Modeling

To be able to provide a service as described in chapter 3 operational efforts have to be reduced to a minimum. Every human interaction leads to higher costs, higher probability of an error and slower reaction times. IBM is working on a reference architecture that is able to manage complex data center automation tasks. This reference architecture is called Common Cloud Management Platform (CCMP).

Figure 4.1 shows a cloud management middleware stack based on the CCMP reference architecture. The Tivoli Service Automation layer deploys and manages Services into the Virtualized Infrastructure Layer that can serve different workloads. [BSWS+10]

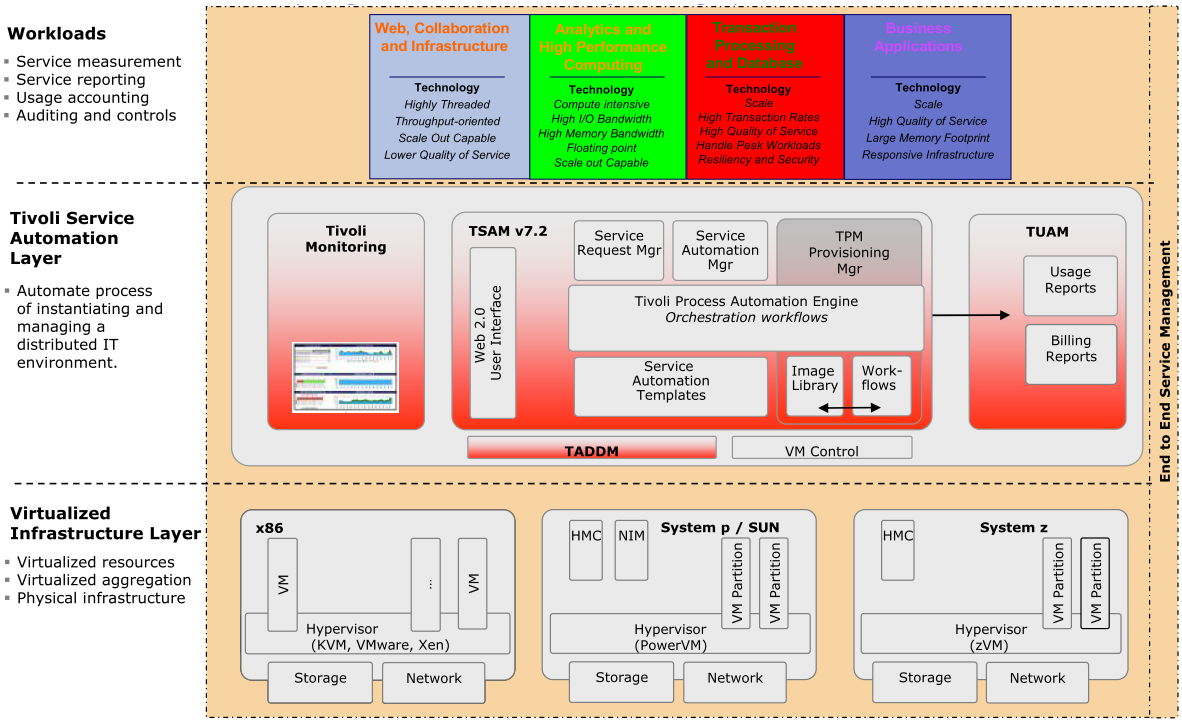

Figure 4.1: CCMP RA Stack [BSWS+10]

The scope of this thesis is the deployment of services and will therefore concentrate on the Tivoli Service Automation Manager (TSAM or TivSAM) part of that Service Automation Layer. Monitoring and metering will not be investigated here.

The following sections describe the service model of TSAM. Based on that a modeling language is created to model the Archive Cloud Service described in chapter 3.

4.1 Introduction in Tivoli Service Automation Manager

To manage big landscapes of IT infrastructure IBM developed the Tivoli Service Automation Manager. It implements the whole service life cycles with a service oriented approach. All assets within a deployment of a service are modeled within its Service Definition. This Service Definition contains information about the hardware and software requirements as well as how they need to be installed.

TSAM is a component for the Tivoli Process Automation Engine implementing a data model, workflows and applications to manage IT services. TSAM itself manages all processes of the service life cycle. These processes contain approval workflows for deployment, modification or undeployment of services. It also manages the reservation of resources and the steps needed to deploy a service. To execute provisioning operations it uses the Tivoli Provisioning Manager (TPM).

All hardware and software assets are managed within the Tivoli Change and Configuration Data Base (CCMDB). It contains the state of the IT landscape with all deployed services and their components as well as the hardware resources that are unused so far.

There are four graphical interfaces to TSAM for different user groups. An Administration UI, Offering Catalog UI, Tivoli Service Request Manager (SRM) Admin UI and TPM Admin UI. With its Administration UI it is possible to access all the TSAM specific functions. This UI is meant to be used by administrators, operators and designers.

To enable end users to control TSAM the Tivoli Service Request Manager can be put in front of TSAM. SRM allows exposing a certain set of functionality to the end user in form of offerings in the SRM offering catalog. In this Offering Catalog UI end users can request and manage their services. These service offerings can be administrated in the SRM Admin UI.

All of the aforementioned products can be integrated into existing applications via the Maximo Enterprise Adapter (MEA) or an REST interface.

The following sections describe how services are modeled within TSAM using Service Definitions and how they are offered to the customer using Service Offerings. This model will be used in the next chapter to break the Archive Cloud Service down into manageable resources.

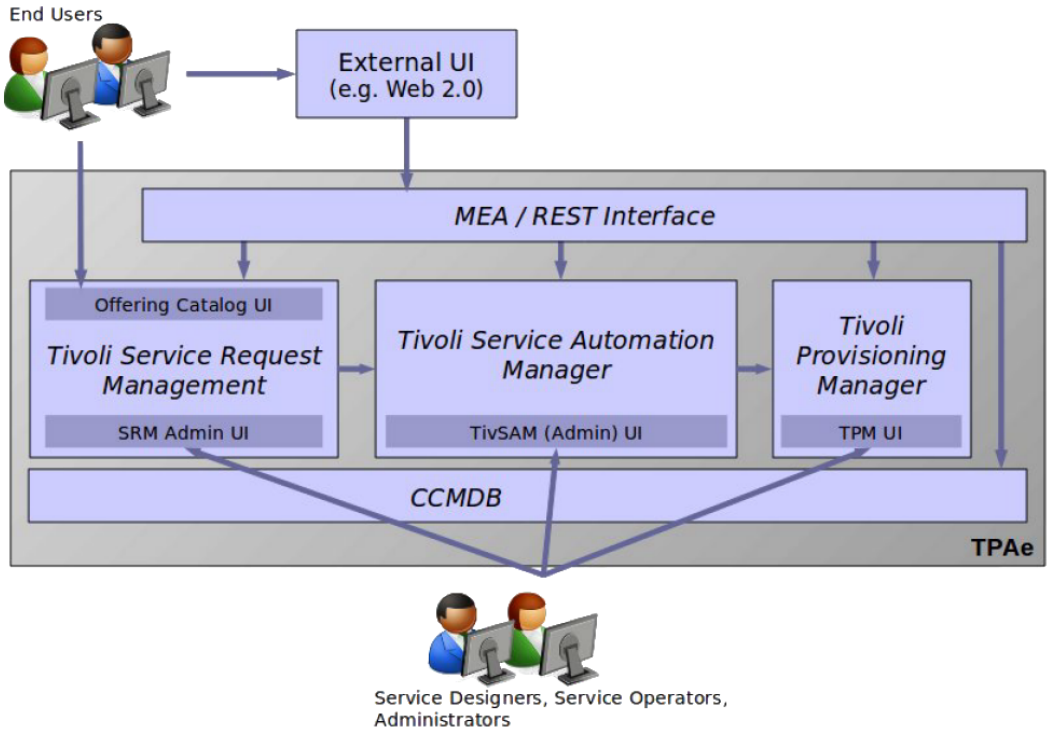

Figure 4.2: TSAM User Interfaces [SIKD+10]

4.2 Service Definition

A service in TSAM is modeled with Service Definitions. A Service Definition is an abstract model of a provided service. It describes what hardware and software requirements the service has and how they can be installed and configured. A Service Definition is separated into two parts, a structural and an operational model (see Figure 4.3).

The structural model describes what pieces of software are needed for the service and what attributes and operations they support. The operational model describes in what order the operations have to be executed to fulfill a management goal like deploying the service.

How these two models work in detail is explained in the next sections.

4.2.1 Structural Service Model

The structural service model describes components and the relationship between the components of a service. Each component is modeled with a Service Topology Node. A Topology node is a generic object that does not contain information about the type of component that it represents. Types of components are modeled with Maximo Classifications. These

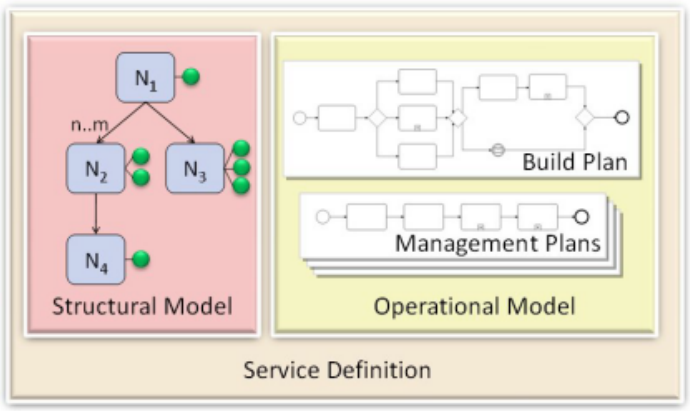

Figure 4.3: Service Definition - Overview [BSWS+10]

classifications are shared system wide and can be reused in all Service Definitions. Topology Nodes on the other hand are defined explicitly for a Service Definition.

The level of granularity a service is modeled in depends on the scope of the service. For example, a model for automating the management of WebSphere clusters will contain elements such as WebSphere Nodes and Application Server Instance, or logical entities like a Cell and a Cluster, but it will in most cases not handle elements at or below the operating system layer, these are more or less taken as a prerequisite.

The structural Service Definition may contain both topology nodes that are managed by TSAM processes as well as static nodes, that are not managed by TSAM but are used by the service. For example a WebSphere cluster could use a DBMS that is managed by TSAM or one that is set up manually.

Maximo Classification

A classification defines the attributes a Topology Node has. For instance a classification could be *HTTP Server* with attributes like *Installation Directory*, *Administrator user name* or *HTTP port*. When a topology node is classified as an HTTP Server it inherits all attribute definitions from its classification.

Relations between Topology Nodes

The structure of a service is defined with a tree of topology nodes. A node can have multiple child nodes but only one parent node. This relationship corresponds to a containment relationship. For instance *"A WebSphere Cell, for example, consists of (it contains) an HTTP server, a Deployment Manager, one or more Managed Nodes, and it can define clusters"* [SIKD+10]

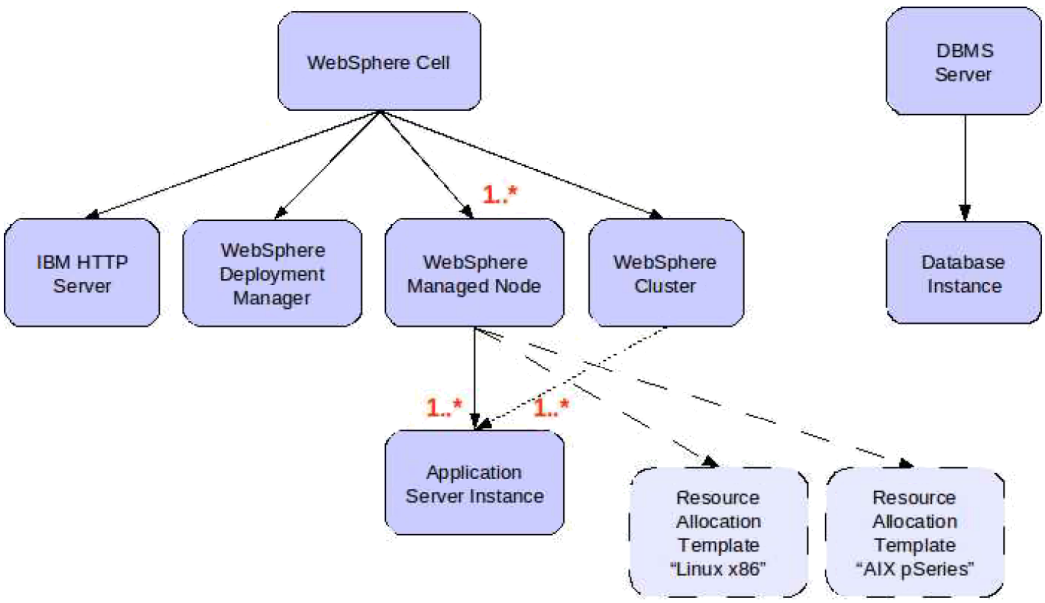

Figure 4.4: Structural Service Model [SIKD+10]

According to [SIKD+10] almost all environments can be structured in a hierarchical way. This dependency is modeled with a solid arrow in Figure 4.4.

Additionally to this strictly hierarchical containment relationship custom relationships can be defined. These relationships can model service specific semantics. For instance a WebSphere Cluster has a relationship to all its Application Server Instances. But are modeled in the parent-child relationship as children of the WebSphere Managed Node. The additional relationships can be used to easier navigate through the topology model within the operational model.

Topology Node Cardinalities

In a Service Definition it is possible to define the cardinality of Topology Nodes. This means that a Topology Node can be deployed multiple times within an instance. For every Topology Node a minimum and maximum cardinality is configured. The default of these values is one. In some cases it might be desirable to only define a Topology node once with all its attributes, operations and dependencies and increase or decrease its cardinality on demand. For example the WebSphere Cell in Figure 4.4 can contain an infinite number of WebSphere Managed Nodes.

Resource Allocation Templates

Another element of the topology model are Resource Allocation Templates. They are used to describe the relationship between a Topology Node and IT resources and is therefore only needed for nodes that have direct hardware requirements. For example a WebSphere Node, including its WebSphere Application Server requires a server with enough memory and an operating system, to be installed on. A WebSphere Cell in contrast does not have direct IT resource requirements. A Resource Allocation Template might contain requirements on hardware components like minimum memory and CPU requirements as well as requirements on software that is installed, like their versions.

When a service is instantiated, the Resource Allocation Templates are used to find fitting resources, the applications can be deployed onto. Topology nodes can have different Resource Allocation Templates for different environments. For example different operating systems. For each operating system requirements can be defined(Figure 4.4)

4.2.2 Operational Service Model

The Operational Service Model contains the processes necessary to modify a service instance. This could be instantiating the whole service or adding and removing components. For each modification task it contains a Management Plan.

These Management Plans describe the steps necessary to fulfill the management goal. These steps are implemented with Topology Node Operations. A Topology Node Operation relates to a Topology Node like a Management Plan to a Service Definition. It modifies the state of a Topology Node like installing or configuring software components. Figure 4.5 displays the relation between topology nodes (purple), Topology Node Operations (green) and a Management Plan (white).

Like the attributes of a Topology Node its operations are defined within its Maximo Classification. The implementation of an operation is done with Maximo Job Plans. There are some default Job Plans for example to invoke TPM workflows or custom scripts.

Mapping of Input and Output

Another important part of the Management Plans is the mapping of input and output data. A Management Plan Task has a specified interface defining which input and output parameter it has. But it is not defined where the data comes from and where it has to be stored after the execution. To define this data flow Input Mappings and Output Mappings have to be created. Input Mappings are executed immediately before a Management Plan Task to collect and provide the data needed. Output Mappings are executed immediately after a task execution and manage the storing of output data.

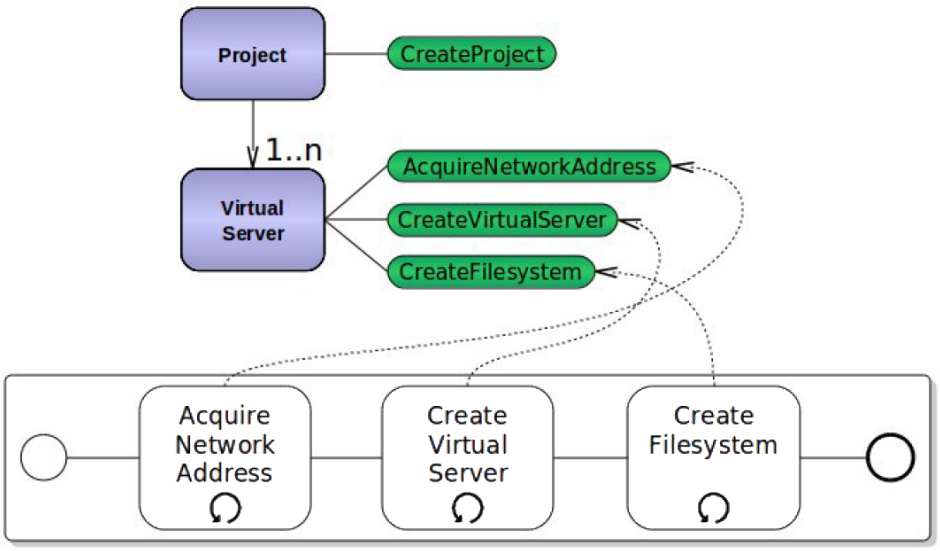

Figure 4.5: Service Definition - Management Plan [SIKD+10]

The most common way to manage the input and output data is to store them in topology node attributes. Other options are constants, user input requests through a GUI or parameters that have been passed to the Service Request.

Figure 4.6 shows how input and output mapping works. The Aquire Network Address tasks retrieve a hostname and an IP address from a pool. That data is then stored within the Topology Node of the corresponding Virtual Server. The Create Virtual Server tasks then retrieve this data and data about memory and CPU size of the virtual machine. This data is then used to deploy a virtual machine with a TPM workflow.

Preparation Workflow

The processing of a Management Plan always has two phases. A preparation phase and an execution phase. In the preparation phase a Maximo Workflow is executed, that gathers all relevant data, prepares the Topology Model and requests all required approvals.

The Management Plan is analyzed for required input data. For instance, if it has to be executed on a selected Topology Node, the user that requested the Management Plan execution will have to select it.

When the execution of the Management Plan results in new components being installed, the preparation workflow will create the required Topology Nodes. A preparation workflow for the management plan shown in Figure 4.5 will have to perform the following steps.

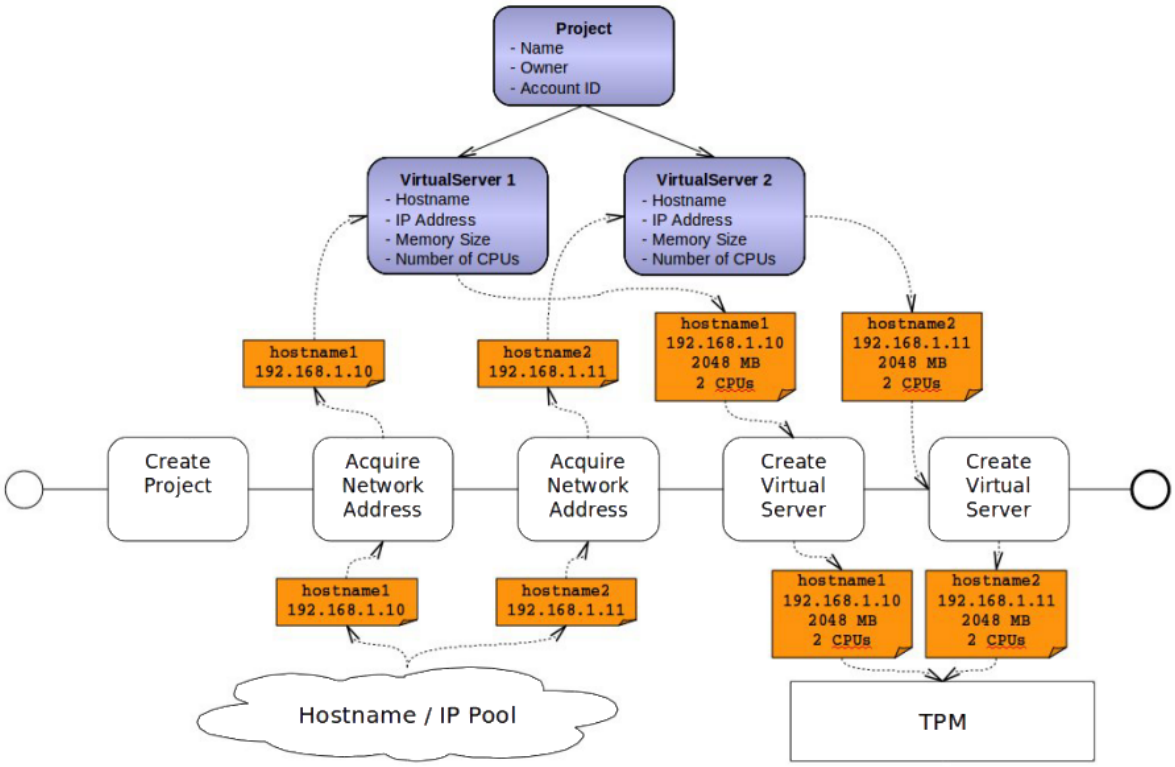

Figure 4.6: Service Definition - Input/Output Mapping [SIKD+10]

Having a cardinality of 1..n for the Virtual Server node, the requester has to select the number of servers he wants to create. Also it might be necessary to provide more data about the server configuration like the amount of memory and CPUs, if they have not been predetermined within the Topology Model. After retrieving all information about the resource requirements of the Virtual Servers the resources can be reserved on physical machines. At this point the jobplan is ready for execution, but depending on the configuration its execution has to be approved by an administrator first. If so, an approval request for the administrator is created. When approved the Management Plan will be executed.

4.3 Service Offerings

The Service Definitions described above are only abstract templates of a software service. To be used they have to be instantiated first. There are two ways of instantiating a Service Definition. The first way is to use the TSAM interface directly. This however is deprecated because it bypasses the common TPAe way of using Service Requests. Also using TSAM directly would require the end-user to understand the technical details of it. To simplify requesting of a service Service Offerings are used.

Service Offerings are put on top of a Service Definition and only expose their external functionality provided by the Management Plans. Service Offerings are implemented using the Tivoli Service Request Manager. With it Service Offerings can be defined and provided using Offering Catalogs. An Offering Catalog is a set of Service Offerings that can be provided to certain user groups. For example there could be a group of software testers. For this group there could be an Offering Catalog that offers Service Offerings to deploy, update or undeploy a testing system.

Whenever an end-user requests a service from the Offering Catalog (Catalog Request) a Service Request is created. To request a service a form has to be filled out with all required parameters. The created Service Request consists of a Fulfillment Manager Approval Workflow, which contains all necessary approval steps and the Management Plan it should execute. When the execution of the Management Plan is approved, it is sent to TSAM for its fulfillment.

To be executed by TSAM the Service Request contains the ID of the Management Plan to be executed as well as parameters for the name of the new service instance as well as the ID of the service Definition and its revision or the ID of an existing service instance.

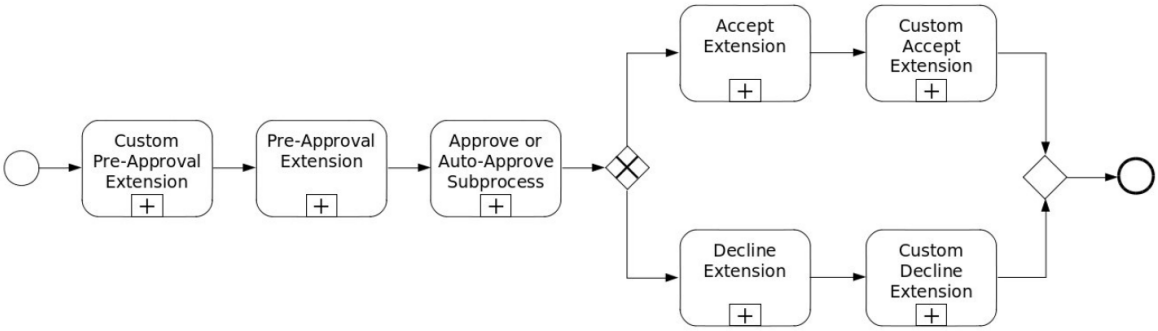

Figure 4.7: Service Request Processing Workflow [SIKD+10]

The Service Request Workflow (Figure 4.7) consists of several sub workflows of which the custom extensions are optional:

Custom Pre-Approval Extension This extension allows for customer specific preparation steps, like validation the resources for a request.

Pre-Approval Extension This extension is for Service Definition and Management Plan related preparation. For example a Service Definition Instance might be created and the needed resources reserved. Because the Service Request might be declined, the created Service Definition instance is referred to as Draft Service Instance.

Approve or Auto-Approve Subprocess This process manages the approval of a Service Request. Depending on the system property *pmrdp.enable.autoapproval* the request will be automatically approved or has to be approved by the TSAM Cloud Administrator role.

Accept Extension This extension is executed when the Service Request has been approved. It executes further preparations to continue with the TSAM processing.

Custom Accept Extension This extension allows for customer specific preparations to continue with the TSAM processing after the approval.

Decline Extension This extension is executed when the Service Request has been declined. It is used to clean up the Draft Service Instance and resource reservations.

Custom Decline Extension This extension allows for customer specific cleanup operations to be executed when the Service Request is declined.

4.4 Modeling Concepts

As described in the previous section [SIKD+10] uses different graphical modeling languages for Service Definitions. Since there is no tool for the modeling I will adapt the concepts of the modeling into the common modeling languages Unified Modeling Language (UML) and Business Process Modeling Notation (BPMN). Both languages are assumed to be known and not further described.

The following sections introduce the modeling languages that are used for the structural and operational description of a Service Definition.

4.4.1 Structural Modeling

To model the hierarchy of Topology nodes I decided to use UML Class Diagrams. To describe the modeling concept I remodeled the example from Figure 4.4 within introduced language (see Figure 4.8).

To represent a Topology Node I use a UML Class element without its operation and attribute fields, because they are not relevant for their structural hierarchy. The Directed Association describes the strong consists of (contains) relation, from every Topology Node only one of these relations can originate.

The Dependency describes the weaker relationship between Topology Nodes. In my model Topology Nodes that contains another Topology Node will always be placed under its contained Topology Node. This way all arrows will always point upwards. To make the graphics better to read the constraint, that arrows will always originate from the top of a Class box and end on the bottom of a Class box, has been added.

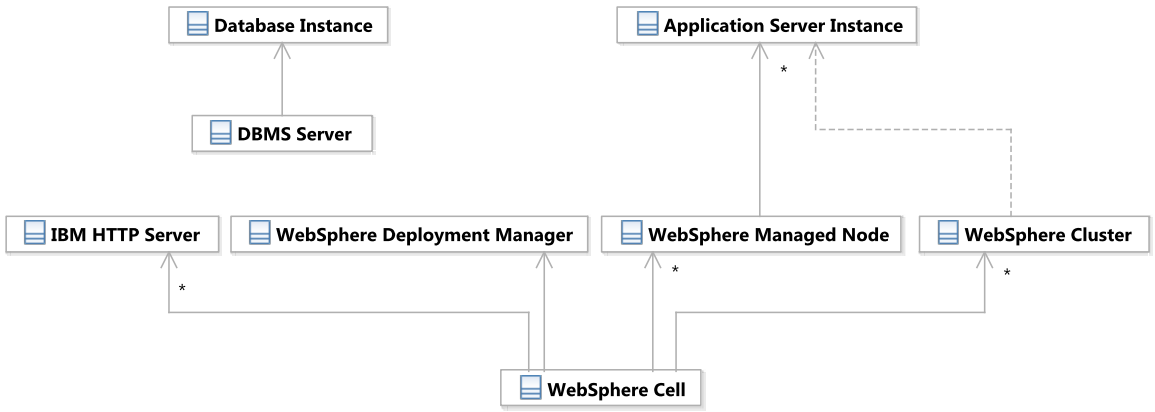

Figure 4.8: Structural Model Example

Cardinality is depicted using multiplicity of Directed Associations containing the amount of instances of Topology Nodes. A WebSphere Cell can contain multiple WebSphere Clusters, WebSphere Managed Nodes and HTTP Servers. A WebSphere Managed Node can contain multiple Application Server Instances.

4.4.2 Operational Modeling

To model the Operational Model BPMN Models and UML Class Diagrams are being used. The BPMN Model describes the steps of a Management Plan and a UML Class Diagram describes the input and output mapping of operations.

Management Plans

In Figure 4.9 my representation of the Management Plan in Figure 4.6 is depicted. The relationship between a Management Plan Task and a Topology Node Operation is not modeled within this diagram and will be considered in the Data Flow Diagram.

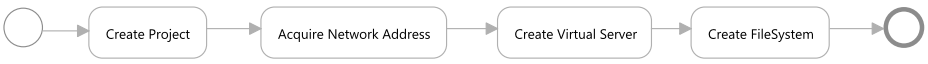

Figure 4.9: Operational Model Example

Data Flow

The diagram in Figure 4.10 shows the modeling of the data input and output of the Topology Node Operations. To model a Topology Node the notation of a UML Class is used. Arrows indicate the input and output data of operations. The beginning of an arrow indicates where data originates, the head of the arrow indicates where the data goes.

In the case shown in Figure 4.10 the acquireNetworkAddress operation puts data in the Hostname and IP Address attribute of the Virtual Servers. The data itself is generated within the operation. The createVirtualServer operation uses all four attributes of the Virutal Server Topology Node Instance to physically deploy a virtual server in the environment. The data for Memory Size and Number of CPUs is set when instantiating the Topology Model in the preparation phase of the Management Plan (see chapter 4.2.2).

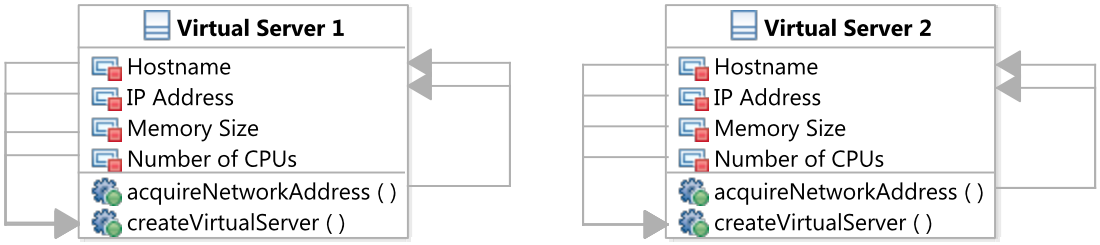

Figure 4.10: DataFlow Model Example

5 Implementation

In this chapter the modeling of the Archive Cloud Service for TSAM will be described. Due to time issues of the TSAM installation only the segregation into topology nodes and the implementation of automation steps will be presented. An integration of this model within TSAM will be done after this thesis.

5.1 Archive Cloud Service Software Stack

The Archive Cloud Service as described in chapter 3 is provided by several applications. The central part is the FileNet Content Engine. It provides a Repository for storing and managing documents. This Content Engine is extended by the FileNet Records Manager to declare documents as records. The Workplace XT application is the web frontend of FileNet. It can be used to manage FileNet.

The FileNet P8 Content Search Engine is used to create full text indices of the stored documents. These full text indices are used by the eDiscovery Manager to search for documents.

All these applications run within a WebSphere Application Server environment. For user and group management the Tivoli Directory Server is used. The IBM DB2 database management system is used for the Databases. All data is stored on a GPFS file system.

5.2 Analysis of the software components

To segregate the software stack into manageable resources, the applications have to be separated into simple disjoint black boxes that are reduced to their exposed attributes and functionalities. These black boxes are not only applications but also logical structures within applications, e.g. data bases within a DBMS. They will be implemented as topology nodes within TSAM.

In the following sections all layers of the software stack described in chapter 3 are deconstructed into components. These components are analyzed on their multitenancy capabilities. The description of the components is divided into three parts a basic description of the

component, an analyzation on the multitenancy capabilities and a structural description to model the component for TSAM.

After the description of the structural model of the components the process of deploying a new tenant is described. Using the operational modeling concept described in chapter 4.4.

5.2.1 Storage

Description

The most basic component in our stack is the storage component. All our data is stored on a cluster file system (GPFS). In this component model only storage for data is considered. Application data like binaries and configurations are stored on the cluster as well. Most file systems support files, folders and file system objects such as soft and hard links. GPFS additionally supports file system objects called FileSets. FileSets behave like file systems but can be mounted within the file system and therefore allow administrative operations at a finer granularity.

Multitenancy

Multitenancy is supported in this layer. As described in chapter 2.1 tenants could share folders, file systems or hard drives. Assuming that all of these possibilities would have the same usage within the system, because all of these concepts will have a path to their storage location, in the end only security and operational issues have to be considered. The concept of FileSets is not a part of the four tiers described in chapter 2.1, but concerning isolation it shares the level of a file system. We assume that the security of a FileSet is sufficient enough for our use case and decide to let tenants share the hard drives the FileSets are stored on.

Structure

Our application needs three separate locations to store data. We will use one FileSet per storage location.

WAS_FileSet In this storage area WebSphere data will be stored. This contains configurations of the applications as well as the data of FileNet Content Engine's Object Stores.

DB2Data_FileSet In this FileSet the data files for the databases will be stored.

DB2Log_FileSet This FileSet stores the transaction logs of DB2.

Figure 5.1 shows the topology nodes needed for storage element of the Archive Cloud Service. The GPFS Cluster component is a logical representation of everything needed to provide a GPFS Cluster system. This GPFS Cluster hosts the three FileSets. According to the modeling concept in chapter 4.4 the relationship is modeled with a directed association.

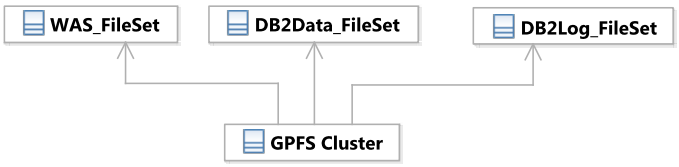

Figure 5.1: Storage topology

5.2.2 Database Management System

Description

For the DBMS a DB2 HADR (High Availability Disaster Recovery) solution is used (see Figure 5.2). This configuration consists of two sites. The primary site consists of two servers that run in an active/passive pattern to achieve high availability. Both servers share the necessary resources like their IP address, mount point and database. In case of a failure of the active server, the passive server can take over immediately.

The second site contains a standby server for disaster recovery. While the primary site communicates with the client application, the standby site is only keeping itself in sync with the primary site by applying the transactions from its log buffer to the own database. If the primary site fails, the standby site can take over the operation within half a minute.

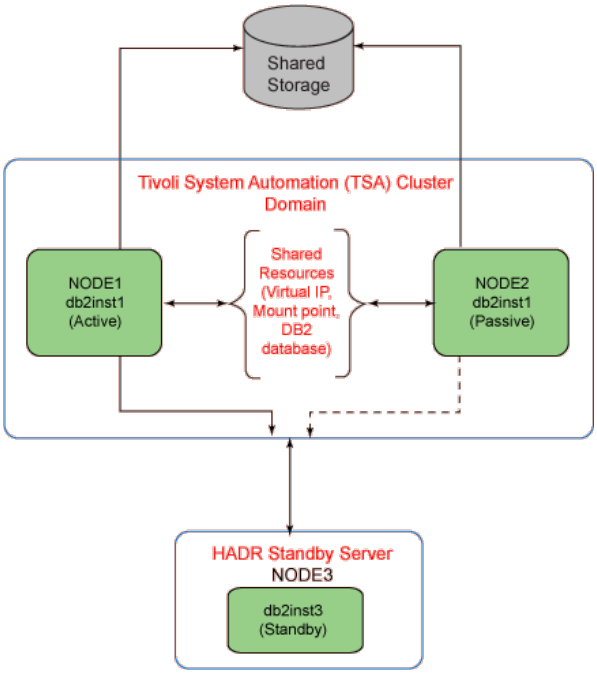

Figure 5.2: IBM DB2 HADR Configuration [Suno7]

This configuration also allows rolling updates by updating one of the servers at a time and therefore stay operating all the time.

Multitenancy

As described in chapter 2.1 there are 4 tiers of database sharing ranging from shared tables to separate database servers. Using legacy software that requires multiple separate databases, we decided to use one shared instance of a database management system.

Structure

Figure 5.3 displays the Topology Nodes and their dependencies needed for the database element of the service. On the lower left hand side you can see the DB2 HADR Cluster. For simplification reasons the cluster is not modeled in full detail, containing different sites and servers. The DB2 HADR Cluster is *needed by* the DB2 Instance. The DB2 instance is a logical component, that is able to accept database connections and hosts databases.

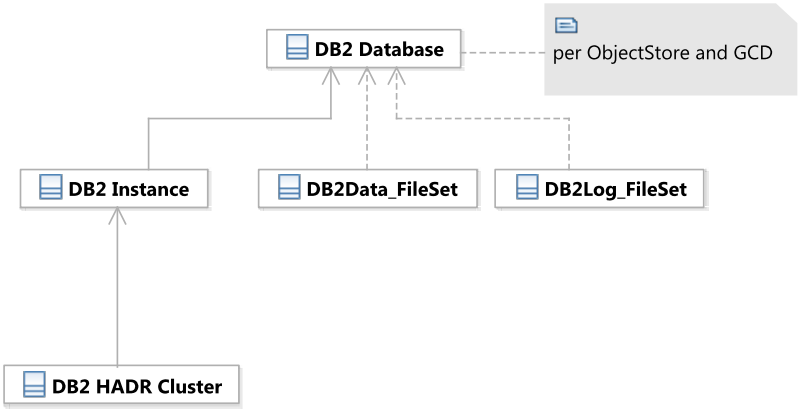

Figure 5.3: Database Management System topology

According to chapter 4.4 UML Dependencies are used to describe weaker relationships than with Directed Association.

The cardinality of the DB2 Database is described in the UML Note. In our environment we need multiple databases. There is one database needed for every ObjectStore within FileNet CE. Another database called Global Configuration Database is needed by FileNet CE for it's configuration.

A DB2 Database *needs* a DB2 Instance to be executed in and storage areas to store both the data and the transaction logs.

5.2.3 Application Server

Description

As application server for our environment we use a WebSphere Application Server Network Deployment (WebSphere ND). Application servers are runtime environments for Java Enterprise Edition (Java EE) applications. WebSphere ND is an application server system that allows to span cluster over multiple servers.

Figure 5.4 shows the components of a WebSphere ND deployment. The Application Server is the primary runtime component. This is where the Java EE Application is actually executed. A WebSphere Node is an administrative grouping of Application Servers within one operating system instance. There can be multiple nodes within one operating system but a node can not be spanned over multiple operating systems. Nodes consist of a Node Agent and one or more Application Servers. The Node Agent manages the configuration of the Application Servers and deploys applications on them.

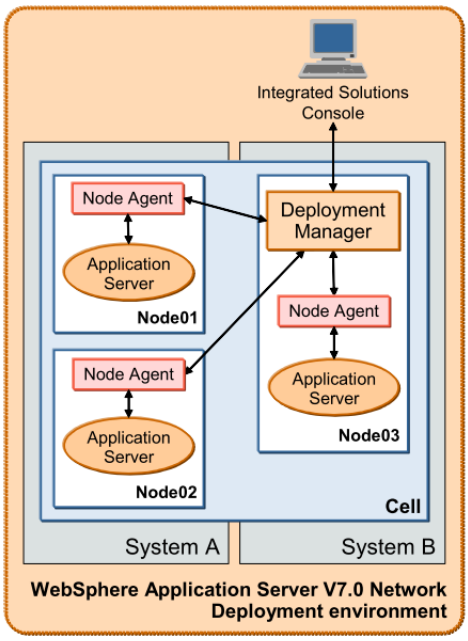

Figure 5.4: WebSphere ND concept [AHP+09]

WebSphere Cells group multiple WebSphere Nodes into a single administrative domain. WebSphere Cells are administered with the WebSphere Deployment Manager. The Deployment Manager communicates with the Node Agents to manage the application servers within the respective WebSphere Node. The Deployment Manager can manage multiple WebSphere Nodes but a WebSphere Node can only be administered by one Deployment Manager. A master configuration of all WebSphere Nodes is stored within the Deployment Manager and is replicated onto all Application Servers when changed.

The Deployment Manager can not only administer Application Servers but also HTTP Servers. HTTP Servers are not part of the actual WebSphere deployment, but they can also be configured. All requests to WebSphere applications are first directed to HTTP Servers. If the request does not require dynamic content, it can be responded by the HTTP Server directly. If not the request is forwarded to a WebSphere Application Server. To configure this request forwarding HTTP Server plugin configurations are used. The Deployment Manager is capable of automatically generating and propagating these plugins for its managed applications.

WebSphere ND compared to a normal WebSphere deployment allows to span clusters (not shown on Figure 5.4) over multiple WebSphere Nodes. Using these, a Java EE application can be deployed over multiple virtual or physical computers. That leads to an automatic load balancing and failover capabilities. Which leads to a scalable and high available runtime environment.

Multitenancy

The WebSphere ND environment allows different levels of Multitenancy. Applications of tenants could be placed on the same within the same WebSphere Cluster, on separate WebSphere Clusters but within the same WebSphere Cell or the applications can be deployed on completely disjoint WebSphere ND deployments. In the case of disjoint WebSphere Clusters in the same WebSphere Cell there is also the option to either share WebSphere Nodes between Clusters or not.

For our solution we decided to give each tenant their dedicated WebSphere Clusters. These clusters however will be spanned over shared WebSphere Nodes.

Structure

In Figure 5.5 the structure of our WebSphere environment is depicted. As described above there is a WebSphere Cell with its Deployment Manager, multiple HTTP Servers and multiple WebSphere Managed Nodes with their WebSphere Application Servers.

For our software Stack we need three WebSphere Clusters. How applications are distributed among the clusters is described later in this chapter. All three clusters are part of a WebSphere Security Domain. This Security Domain configures the security settings for the applications within the clusters.

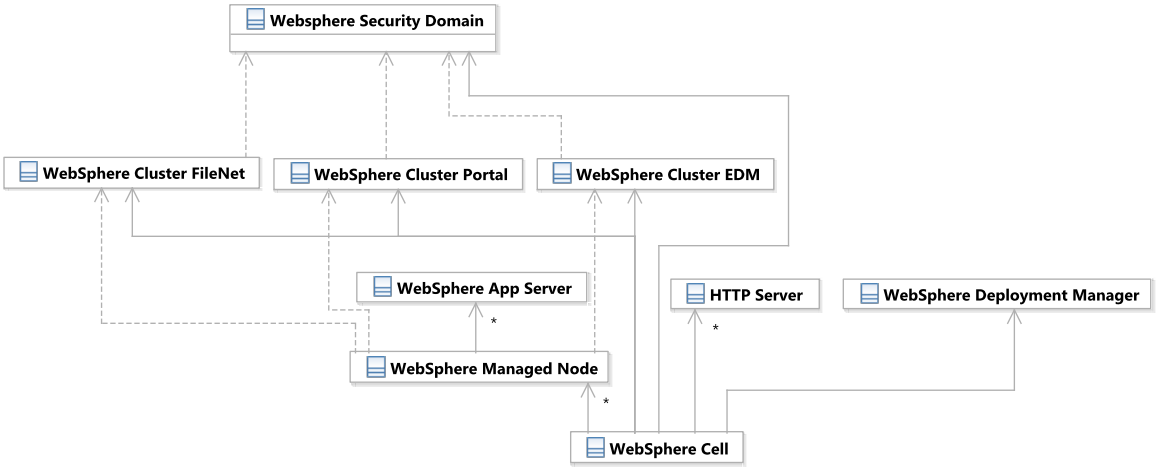

Figure 5.5: Application Server topology

To enable the FileNet Content Engine to communicate with its databases, JDBC DataSources are needed (see Figure 5.6). Because FileNet CE is the only application with access to the

databases the DataSources are added only to the scope of the WebSphere Cluster for the FileNet application. The DataSources use a JDBC Provider to connect to the DB2 Database.

For every ObjectStore and the Global Configuration Database of FileNet CE a pair of DB2 Database and DataSource is created. DataSource and JDBC Provider as modeled here are logical elements that represent both a transactional and non-transactional DataSource and JDBC Provider each.

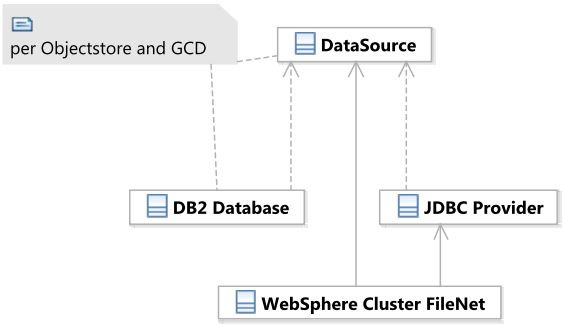

Figure 5.6: WebSphere DataSources topology

5.2.4 Directory Server

Description

The directory server is the central component for user and group management for all deployed applications. In this setup the Tivoli Directory Server is used. Within this directory server an organizational unit is created. It contains technical users for example a user, applications use to connect to the Directory Server as well as customer created users.

Multitenancy

Tivoli Directory Server supports multitenancy by using Organizational Units with separate Access Control Lists (ACL). This way only users from an Organizational Unit are able to access the data within it.

Structure

Figure 5.7 displays the directory server components. It contains a Topology Node for the Tivoli Directory Server which represents all hard and software resources needed to provide

a directory service just like the DB2 HADR solution this could contain multiple servers. This server is needed by the Organizational Unit(OU) that is created within it.

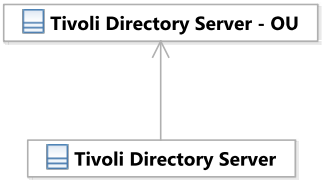

Figure 5.7: Directory Server Topologies

5.2.5 FileNet CE

Description

FileNet Content Engine is the core application of the Archive Cloud service. It provides the object repositories used by all other applications. These object repositories are called ObjectStores in the FileNet terminology. Object Stores reside within the P8 Domain, the basic element of FileNet CE. The domain contains the configuration of the Directory Server and a mapping between roles within FileNet and groups within TDS.

Multitenancy

FileNet Content Engine could support Multitenancy by using one FileNet P8 Domain per tenant. Having concerns about security and maintenance issues with this approach however it was not further investigated.

It was decided to give every customer his own instance of the application. Since the FileNet Content Engine is needed by all components described from now on, Multitenancy will not be addressed anymore.

Structure

As depicted on Figure 5.8 The FileNet Content Engine is deployed into the WebSphere Cluster FileNet. It uses DataSources to connect to the necessary databases. FileNet itself needs access to the GCD and its ObjectStores need access to their databases. Besides the databases the ObjectStores need access to the WAS_FileSet to store their data in.

The ObjectStores reside within the FileNet P8 Domain. This domain needs the Organizational Unit on the Tivoli Directory Server for for users and group management.

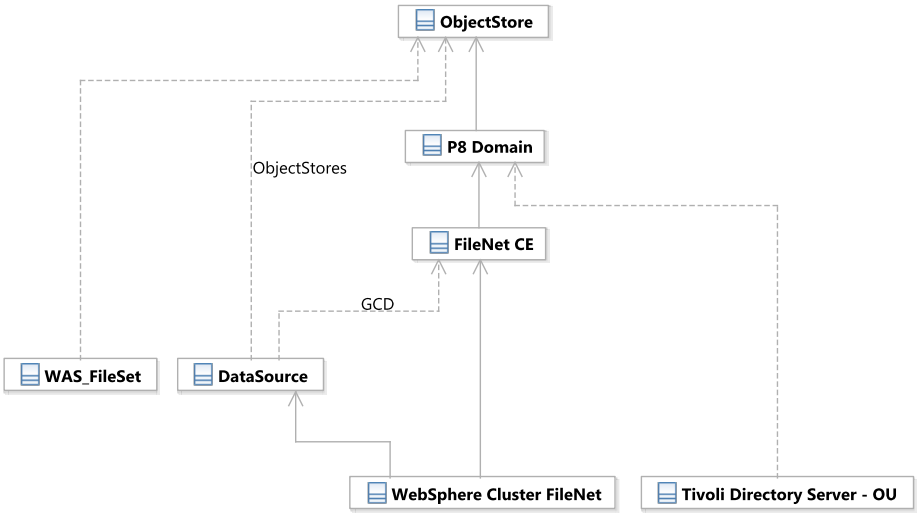

Figure 5.8: FileNet CE topology

5.2.6 FileNet addons

Description

In FileNet CE addons can be installed to extend its functionality. In our deployment we install the Workplace XT extension and the Records Manager extension. On top of that Content Search Engine and eDiscovery Manager are installed.

Workplace XT provides a web interface to the features of FileNet CE. The Records Manager extension adds records management support (see 1.2.2) The Content Search Engine can create full text indices of documents and the eDiscovery Manager is used to discover documents for litigation cases.

Structure

All four applications are Java EE applications that are installed into the WebSphere cluster of FileNet. The Records Management capability of Records Manager is installed into the P8 Domain and can be used by ObjectStores later on. The Content Search Engine is installed separately and then configured to be used by the P8 Domain. The eDiscovery Manager discovers documents and records within FileNet CE and Records Manager.

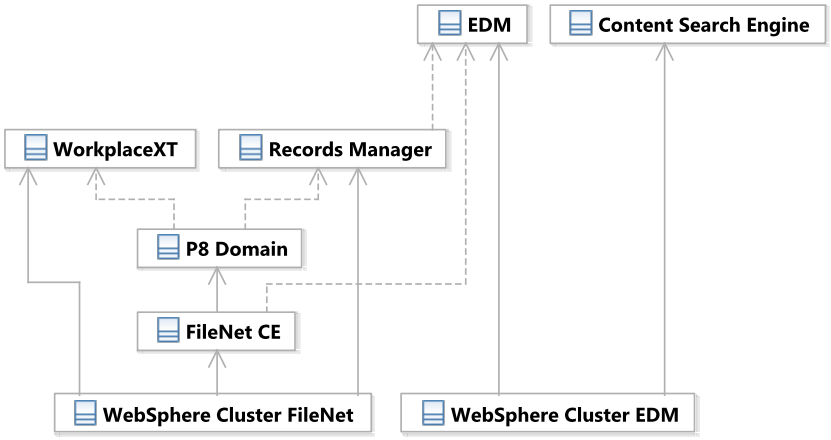

Figure 5.9: FileNet addon topology

5.2.7 Portal

Description

The Archive Cloud Portal is a Java EE application. It provides a subset of the FileNet and Records Manager functionality and therefore needs access to both applications.

Structure

The Archive Cloud Portal is deployed into its own WebSphere Application Cluster. It accesses FileNet CE and Records Manager over their APIs to work with the ObjectStores.

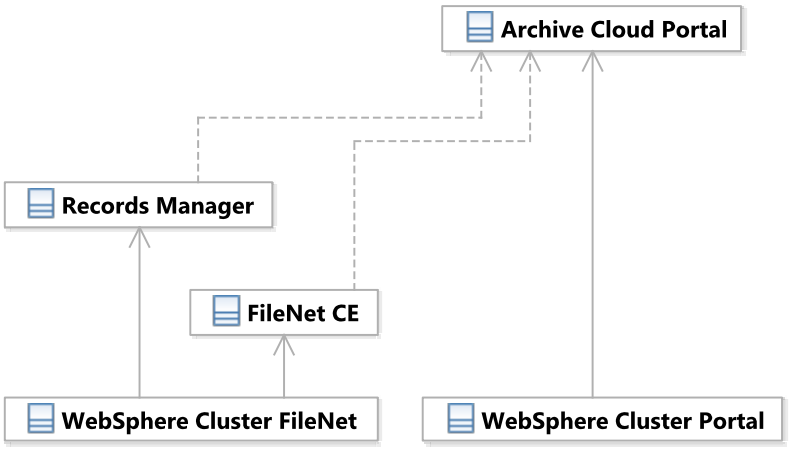

Figure 5.10: Archive Cloud Portal topology

5.2.8 Complete topology

Considering all the dependencies described in the previous sections a complete topology model can be created (see Figure 5.11). It can be separated in a tenant specific and a multitenant part. The lower part of Figure 5.11, labeled with the rectangle, is the multitenant part.

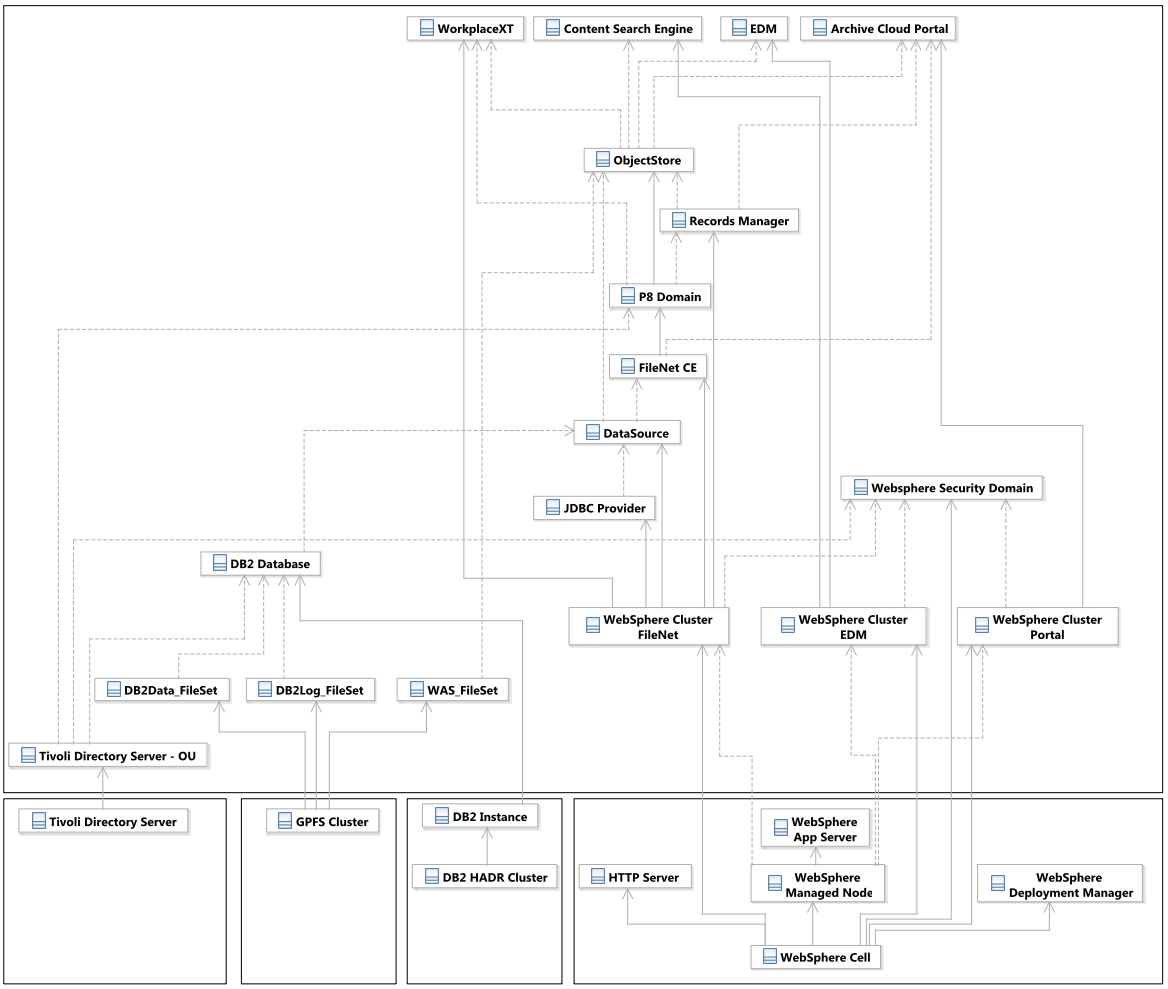

Figure 5.11: Complete topology separated in 5 services

I decided to separate the topology into five separate managed services. Four of these services are static and provide the necessary middleware. These services are Tivoli Directory Server, GPFS Cluster, DB2 HADR Cluster and WebSphere ND. Each of these services aim to provide an automatically scaling middleware service. The fifth service is the Archive Cloud Service that is provisioned for every tenant.

5.3 Provisioning Workflow

This section describes steps necessary to provision the Archive Cloud Service for a tenant as well as their data dependencies.

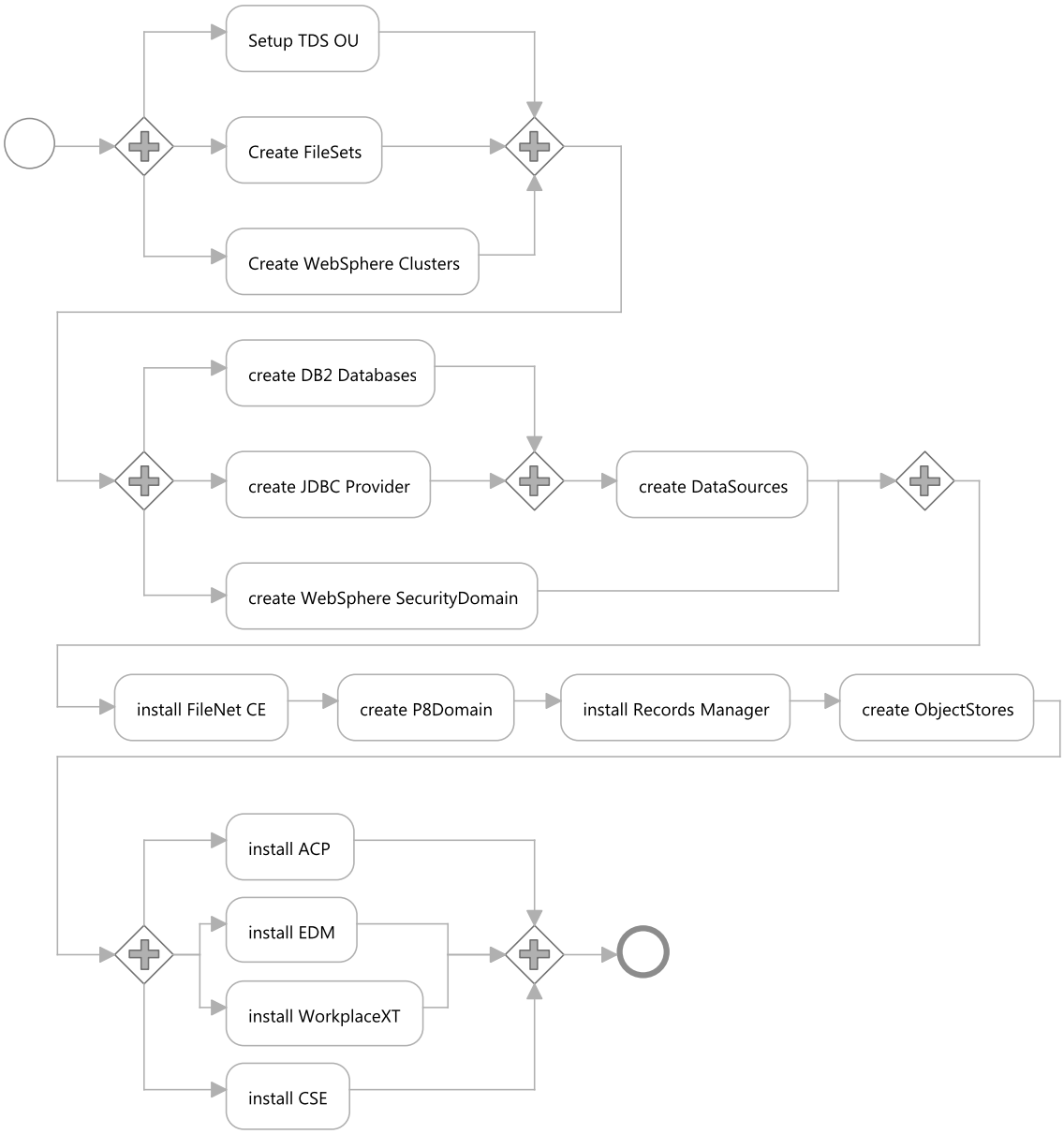

Figure 5.12: Deployment Flow

Based on the dependencies depicted in Figure 5.11 a deployment process can be modeled. The deployment process will be modeled as a build plan for TSAM. TSAM itself does not define a graphical modeling language for deployment processes. To model the process I will use the Business Process Modeling Notation(BPMN).

The deployment process can be separated into 4 parts (see Figure 5.12). The first part sets up basic components for the middleware. The second part configures the middleware to make it ready for application deployment. In the third part the FileNet and Records Manager are deployed and configured to provide the repositories. The last part deploys all applications working with the FileNet CE.

All these parts of the deployment are described in detail within the next sections. The Input and Output mapping will be modeled according to chapter 4.4. This section does not describe all parameters required for each step, but only the ones that are important to understand the dependencies.

As described in chapter 4.2.2 the Topology Model will be created in a preparation workflow. This workflow will generate most of the information in the Topology Nodes. This information is then used to create a representation of the Topology Node within the infrastructure.

The following sections will be separated into the four horizontal sections within the complete deployment flow depicted in Figure 5.12.

5.3.1 Middleware basics

This part of the deployment consists of 3 parallel steps as depicted in Figure 5.12.

One step is creating all necessary elements within the Tivoli Directory Server. First an Organizational Unit has to be created. In this Organizational Unit multiple users and groups are created. There are several administrative and user groups for FileNet CE, Records Manager, Archive Cloud Service and EDM. Additionally administrative users are created for configuring the applications later on.

To create this Organizational Unit an operation on the Tivoli Directory Server is executed. This operation will need information about the Distinguished Names (DN) of the base, users and groups, as well as a user to connect to it and some technical users that are required by other applications later on.

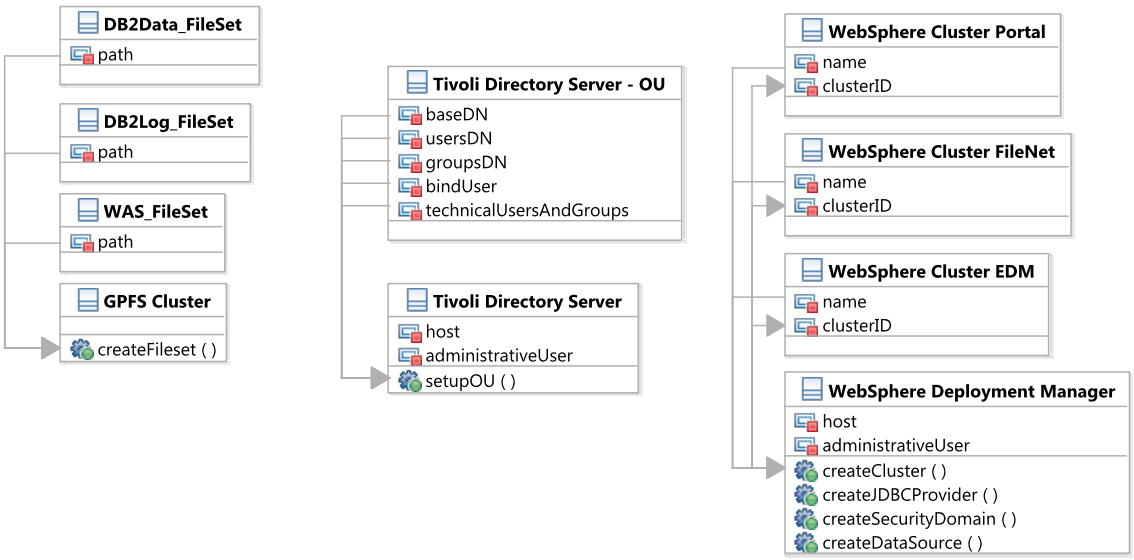

Figure 5.13: Data Mapping: FileSets, Organizational Unit, WebSphere Cluster

The createFileSets task creates the FileSets WAS_FileSet, DB2Data_FileSet, DB2Log_FileSet described in 5.2.1. It also sets their system users and groups and creates the folder structure needed in later deployment steps. To create the FileSets an operation on the GPFS Cluster is called. This operation needs the path of each FileSet, where it should be created.

The createWebSphereClusters task creates WebSphere Clusters and adds member nodes to it. After that the Object Cache Instance is configured for each cluster. A Data Replication Domain is defined for all clusters and an administrative WebSphere user is created. To perform this operation on the WebSphere Deployment Manager the names of the clusters have to be provided. The selection of the nodes that are added to the cluster are considered to be part of the operational logic within the WebSphere ND Service and are not further considered here. When a cluster is created, its ID within WebSphere is stored into the clusters Topology Node to reference it again later.

After these basic preparation steps the middleware can be configured for the application deployment.

5.3.2 Middleware configuration

This part of the deployment contains four tasks (see Figure 5.13 and Figure 5.14). Three of these tasks can be executed in parallel.

Create DB2 Databases creates all necessary databases for the Object Stores and the Global Configuration Database of FileNet. Creating the databases includes tuning their parameters,

creating their buffer pools and table spaces and configuring the access control. The operation to create a database is executed on the DB2 Instance node. To create a database it needs the name of the Database and the storage locations to store the database files and the location to store the database transaction logs at.

The next step that can be executed parallel to the previous one is to create the JDBC Provider. The JDBC Provider in this model as already mentioned in the beginning of this chapter actually consists of two JDBC Providers. one for a transactional database connection and one for a non-transactional database connection. The JDBC Provider is created by the WebSphere Deployment Manager. To create it it requires the ID of the WebSphere Cluster the provider should be created in and the name of the provider. Details about the driver to be used for the provider are not modeled here. When the JDBC Provider is created its ID within WebSphere is saved for further reference into the Topology Node.

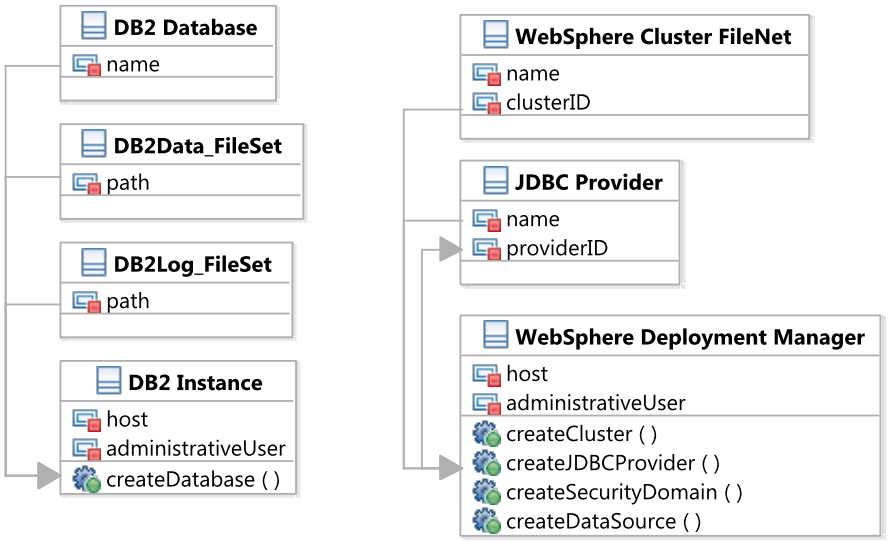

Figure 5.14: Data Mapping: Databases, JDBC Provider

After setting up the databases and the JDBC providers the DataSources can be created. Again the DataSource Topology Node represents both the transactional and non-transactional DataSources required to connect to the database. The DataSources are created by the WebSphere Deployment Manager. To create a DataSource it needs ID of the cluster in which the DataSource should be created. Additionally it needs information about how to access the database. This information consists of the host of the DB2 Instance, the JDBC provider to use and the name of the database.

In parallel to the other tasks a WebSphere Security Domain is created. It sets the security settings for all three WebSphere Clusters and sets up the Tivoli Directory Server Organizational Unit as the user realm. Also an administrative user within WebSphere is created. The

Security Domain is again created by the Deployment Manager. To do so it requires the IDs of the WebSphere Clusters. To access the Organizational Unit it requires the bind user and the host of the directory server. The baseDN of the Organizational Unit defines the realm of the Security Domain. It also needs information about the administrative user to create and the name of the Security Domain.

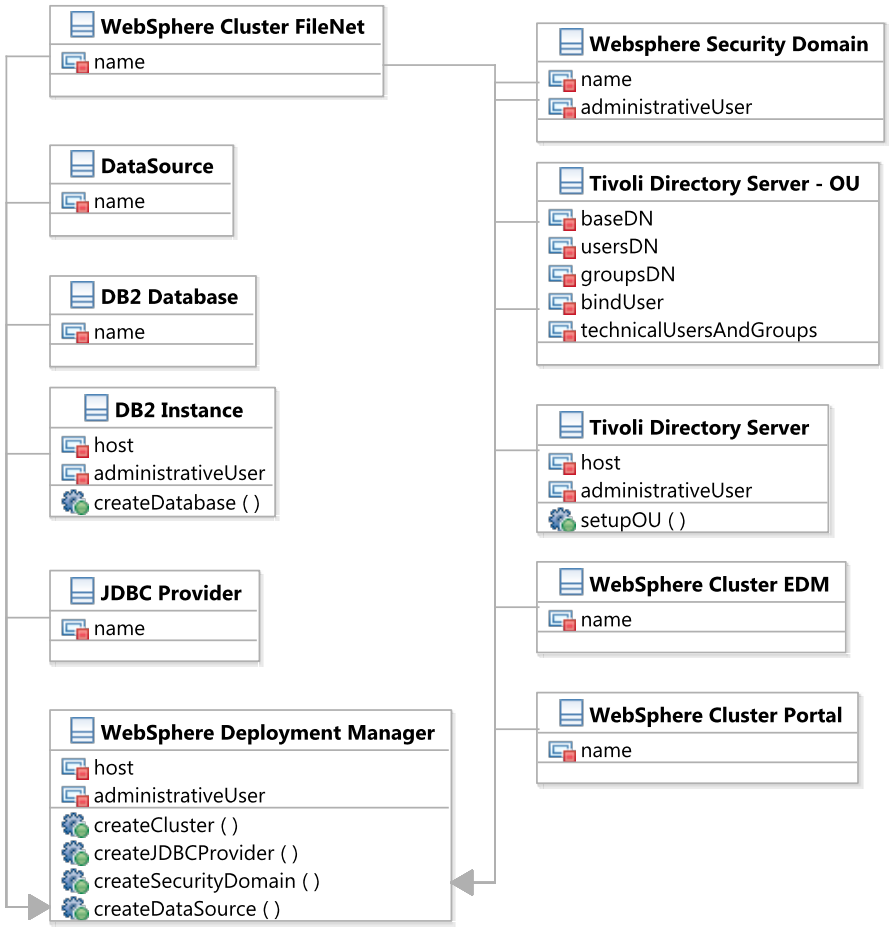

Figure 5.15: Data Mapping: DataSources, Security Domain

5.3.3 FileNet CE and RM

This part of the development sequentially deploys the FileNet Content Engine, creates a FileNet P8 Domain, installs the Records Manager into the Domain and creates Object Stores.

Before the FileNet Content Engine can be deployed it has to be configured. To do so, configuration files have to be put into the WAS_FileSet, the installation therefore needs its path. The configuration needs the DataSource for the Global Configuration Database.

The logic of installing a FileNet Content Engine is provided within the FileNet CE Topology Node. To deploy the application access to the WebSphere Deployment Manager is needed and the ID of the Cluster to deploy the application in.

After deploying and starting the application the FileNet P8 Domain can be created. So far this is still a manual task, but upgrading to a newer version should enable automatization of this task. To create the domain one has to connect to the FileNet Content Engine and input the configuration of the Tivoli Directory Server. This contains access to the server and configuration of the users and groups.

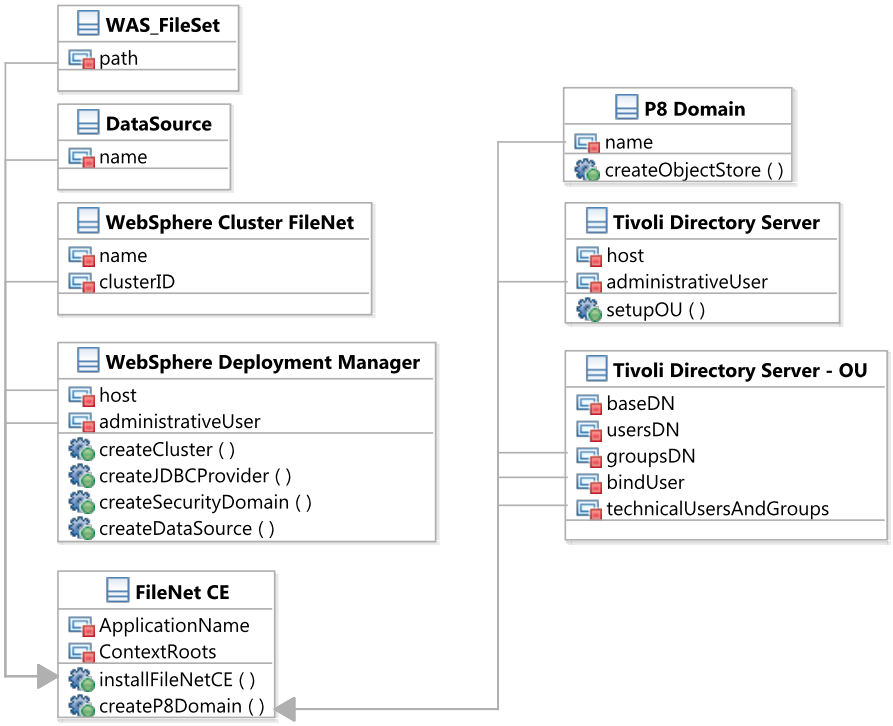

Figure 5.16: Data Mapping: FileNet Content Engine, FileNet P8 Domain

When the FileNet P8 Domain is ready to be used, the Records Manager can be installed. Just like the FileNet Content Engine some configuration files have to be created and put into the WAS_FileSet. These configuration files define the access to the FileNet Content Engine. When the configuration is generated, the Records Manager application can be deployed into

the WebSphere Cluster. After starting the application the Records Manager addon has to be installed into the FileNet P8 Domain, to make it usable within FileNet Content Engine.

Eventually the FileNet ObjectStores can be created and configured. Just like the FileNet P8 Domain this step has to be done manually for now. For every ObjectStore to be created the DataSources have to be provided to access the necessary databases. Since content is not only stored in databases, a file system path has to be provided, where files can be stored. When the storage is configured, access rights have to be configured using the groups that were created within the Organizational Unit in the Tivoli Directory Server.

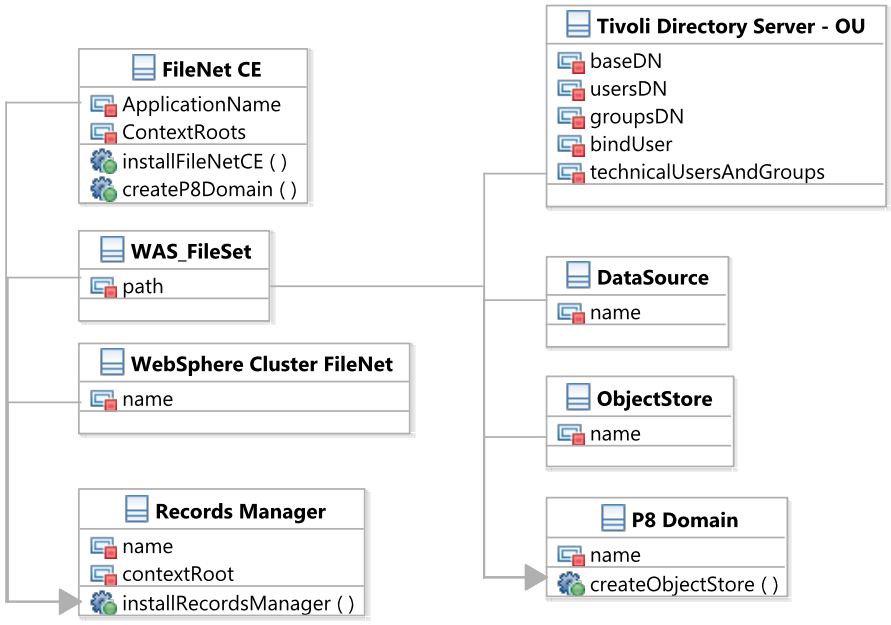

Figure 5.17: Data Mapping: Records Manager, ObjectStores

5.3.4 End user applications

When all ObjectStores are ready to be used the end user applications can be installed. All applications can be installed in parallel. Figure 5.18 and Figure 5.19 show the required resources to install them.

The Archive Cloud Portal is installed into its own WebSphere Cluster using the WebSphere Deployment Manager (see Figure 5.18). The configuration files are stored in the WAS_FileSet. It requires the context roots of FileNet CE to connect to it and and own ObjectStore to store further configuration in.

The Workplace XT application is installed into the same WebSphere Cluster as the FileNet Content Engine (see Figure 5.18). It also stores its configuration into the WAS_FileSet.

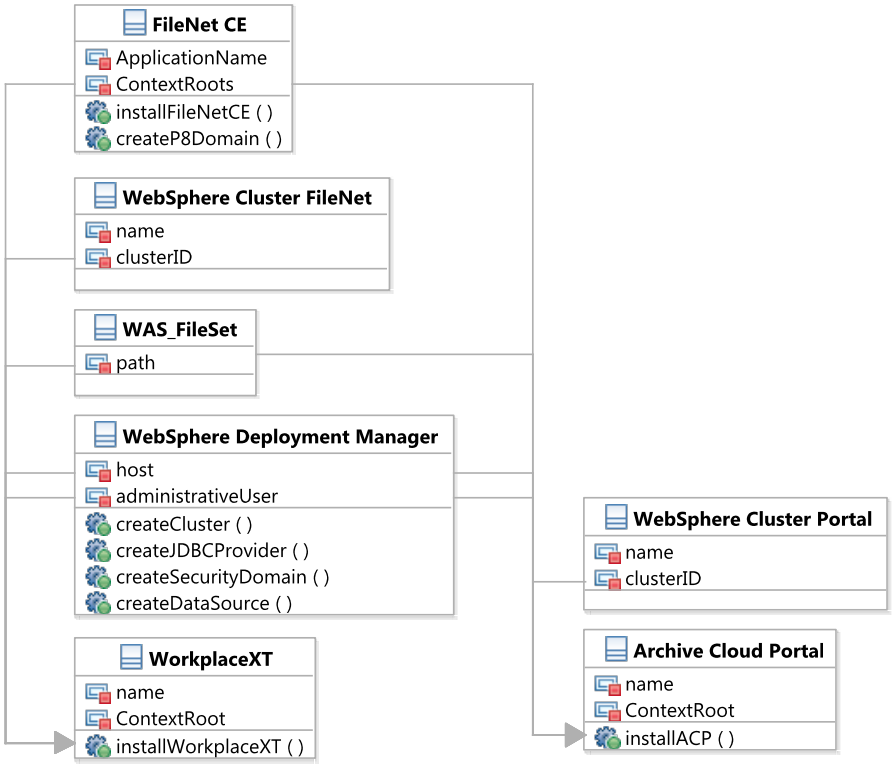

Figure 5.18: Data Mapping: Workplace XT, Archive Cloud Portal

The Content Search Engine is installed into the WebSphere Cluster EDM using the WebSphere Deployment Manager (see Figure 5.19). Its configuration is stored in the WAS_FileSet. When the application is installed the P8 Domain has to be configured to use it as search engine. After that index areas have to be set up for the ObjectStores that shall be indexed.

The eDiscovery Manager is installed into the WebSphere Cluster EDM using the WebSphere Deployment Manager (see Figure 5.19). Its configuration resides in the WAS_FileSet. The configuration contains the information necessary to connect to the FileNet Content Engine.

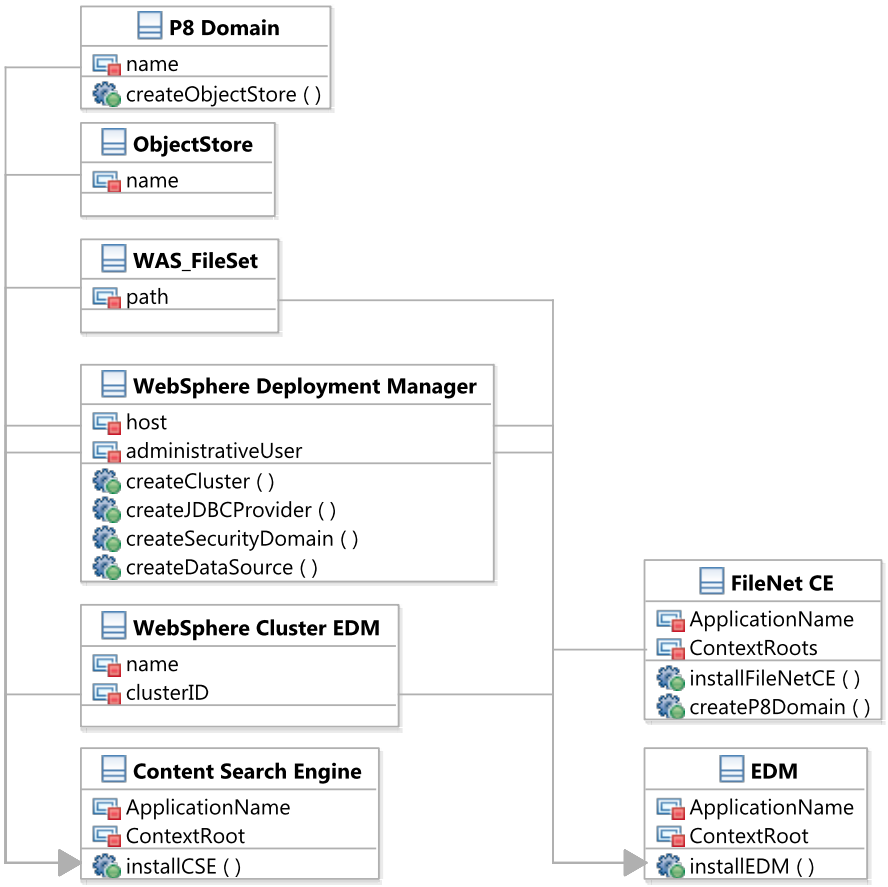

Figure 5.19: Data Mapping: Content Search Engine, eDiscovery Manager

6 Conclusion and Outlook

In this thesis an Electronic Archive Management system has been analyzed and decomposed into managable resources to enable automatic provisioning. Before addressing the decomposition the basics of Cloud Computing have been presented and an insight into Electronic Content Management has been given. The topics Document Management and Records Management have been discussed in more detail and based on that Electronic Archiving Management has been defined.

After providing the fundamental concepts, the concept of multitenancy has been examined. Doing so, layers of tenant isolation have been introduced and concerns about shared resources have been addressed.

Based on these informations the Archive Cloud Service project of IBM has been introduced and its software components have been presented.

To address the automatic provisioning task the Tivoli Service Automation Manager and its concepts of Service Definitions have been introduced. These Service Definitions based on their operational and structural model of software services has been taken as foundation to introduce a modeling concept for our Archive Cloud Service.

Using the introduced modeling concept the Archive Cloud Service's software components have been analyzed and decomposed into Topology Nodes. Then the multitenancy capabilities of these Topology Nodes have been examined. Based on the multitenancy capabilities the service has been devided into five separate services. Four of these services provide the necessary middleware and are shared between all tenants. These services cope with tasks within the scope of High Availability, Desaster Recovery and Scalability. The fifth service is the Archiving Cloud Service itself which is provisioned per tenant. This service deals with provisioning, deprovisioning, configuring and updating of software components.

Having not been able to implement the presented architecture due to complications in the system setup, no results on the resource utilization side can be provided. Having implemented most of the operational part of the service using scripts, the time needed to provision a tenant could be reduced from several days or weeks below two hours.

The segregation of the software stack into multiple services could not only provide the middleware for the Archive Cloud Service but could be used for all kinds of enterprise applications.

Based on the results of this thesis other concerns of the Archive Cloud Service will be discussed in other thesises. The automated scaling of the middleware services based on performance metrics will be presented in a thesis working in the area of monitoring and metering. Techniques of data replication will be reviewed in another thesis.

Bibliography

[AFG+09] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, M. Zaharia. Above the Clouds: A Berkeley View of Cloud Computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, 2009. URL http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html. (Cited on page 15)

[AHP+09] A. Agopyan, H. Huebler, T. Puah, T. Schulze, D. S. Vilageliu, M. Keen. *WebSphere Application Server V7.0 - Concepts, Planning and Design*. IBM, 2009. (Cited on page 50)

[AII] AIIM. What is Enterprise Content Management (ECM)? URL http://www.aiim.org/What-is-ECM-Enterprise-Content-Management. (Cited on page 17)

[Bal07] D. Balovich. Sarbanes-Oxley Document Retention And Best Practices, 2007. URL http://www.creditworthy.com/3jm/articles/cw90507.html. (Cited on page 20)

[BSWS+10] G. Breiter, B. Schmidt-Wesche, B. Snitzer, G. Widmayer, J. Whitmore, J. Villareal, M. Behrendt, R. Caponigro, R. Chang, S. Pappe, T. Weinmann, X. Chotteau. IBM Cloud Computing & Common Cloud Management Platform Reference Architecture (CC & CCMP RA) 1.0, 2010. URL http://www.iaas.uni-stuttgart.de/lehre/vorlesung/2010_ws/vorlesungen/smcc/materialien/Gerd%20Breiter%20CCMPforCloudCourse201011.pdf. (Cited on pages 33 and 36)

[CCW06] F. Chong, G. Carraro, R. Wolter. Multi-Tenant Data Architecture, 2006. URL http://msdn.microsoft.com/en-us/library/aa479086.aspx. (Cited on page 25)

[Dir07] B. Dirking. Lowering E-Discovery Costs Through Enterprise Records and Retention Management, 2007. (Cited on page 11)

[Foc07] FocalPoint Securities, LLC. eDiscovery: The Ongoing Shift to Fortune 500 Clients - A Look at Trends and Major Players in the Legal Vertical, 2007. (Cited on page 11)

[Ful10] Fulbright & Jaworski L.L.P. 7th Annual Litigation Trends Survey Report, 2010. (Cited on pages 3 and 11)

[Goo] Google App Engine - Billing and Budgeting Resources. URL http://code.google.com/intl/en-US/appengine/docs/billing.html. (Cited on page 15)

[ISO01] ISO15489 - Information and documentation - Records Management, 2001. (Cited on page 19)

[Kam06] D. U. Kampffmeyer. *ECM*. Project Consult Unternehmensberatung Kampffmeyer, 2006. (Cited on pages 17 and 18)

[MG09] P. Mell, T. Grance. The NIST Definition of Cloud Computing v15, 2009. (Cited on pages 12 and 13)

[MKW+09] C. Mega, K. Krebs, F. Wagner, N. Ritter, B. Mitschang. CMaaS - Content Management as a Service, 2009. (Cited on pages 20, 24 and 28)

[MMLP09] R. Mietzner, A. Metzger, F. Leymann, K. Pohl. Variability modeling to support customization and deployment of multi-tenant-aware Software as a Service applications. In *Proceedings of the 2009 ICSE Workshop on Principles of Engineering Service Oriented Systems*, PESOS '09, pp. 18–25. 2009. URL http://dx.doi.org/10.1109/PESOS.2009.5068815. (Cited on pages 24 and 27)

[Pro10a] Project Consult. Content Management, 2010. URL http://www.project-consult.de/ecm/wissen/themen/cm. (Cited on page 16)

[Pro10b] Project Consult. Document Management, 2010. URL http://www.project-consult.de/ecm/wissen/themen/dm. (Cited on page 19)

[Roe10] D. Roe. 6 Ways Document Management and Records Management Differ, 2010. URL http://www.cmswire.com/cms/document-management/6-ways-document-management-and-records-management-differ-006454.php. (Cited on page 19)

[SIKD+10] T. Spatzier, M. Illgner-Kurz, H. Daur, F. Triebel, C. Bachhuber-Haller, S. Rodet, R. Schulze, A. Janta, H. Eissler. *Tivoli Service Automation Manager - Solution Guide v1.99*, 2010. (Cited on pages 35, 36, 37, 39, 40, 41 and 42)

[SSW10] K. Stanoevska-Slabeva, T. Wozniak. Cloud Basics - An Introduction to Cloud Computing. In *Grid and Cloud Computing*, pp. 47–61. Springer Berlin Heidelberg, 2010. URL http://dx.doi.org/10.1007/978-3-642-05193-7_4. (Cited on page 13)

[Sun07] G. Sundaram. Implement DB2 high availability disaster recovery in a Tivoli System Automation cluster domain, 2007. URL http://www.ibm.com/developerworks/data/library/techarticle/dm-0704sundaram/index.html. (Cited on page 48)

[ZAB+09] W.-D. Zhu, R. Aitchison, E. Bonner, H. C. Mendez, R. Rathgeber, A. Yadav, H. Yessayan. Understanding IBM FileNet Records Manager, 2009. (Cited on page 30)

[ZCF+09] W.-D. Zhu, K. Cole, A. Fowler, M. Kirchner, B. J. Mcdowell, C. Snow, M. Winter, M. Worel. FileNet P8 Platform and Architecture, 2009. (Cited on page 31)

All links were last followed on March 15, 2011.

Declaration

All the work contained within this thesis, except where otherwise acknowledged, was solely the effort of the author. At no stage was any collaboration entered into with any other party.

(Florian Fritz)